A Modal Analysis of Acoustic Propagation in the Changing Arctic Environment

by

Thomas Howe

B.S. Mechanical Engineering
Boston University, 2013

Submitted to the Department of Mechanical Engineering in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY September 2015

© Massachusetts Institute of Technology 2015. All rights reserved.

Signature redacted

Author ............... Department of Mechanical Engineering
August 19, 2015

Signature redacted

Certified by............. Henrik Schmidt
Professor of Mechanical and Ocean Engineering
Thesis Supervisor

Accepted by ............... David E. Hardt
Chairman, Department Committee on Graduate Theses
A Modal Analysis of Acoustic Propagation in the Changing Arctic Environment

by

Thomas Howe

Submitted to the Department of Mechanical Engineering
on August 19, 2015, in partial fulfillment of the requirements for the degree of Master of Science in Mechanical Engineering

Abstract

This work takes an in depth look at acoustic propagation through double duct sound speed profiles in the Arctic. While the traditional Arctic sound speed profile has a single surface duct, some portions of the Arctic have a sound speed profile which includes a second, lower duct. These double duct systems are seen throughout the Beaufort Sea, dating back to 2004, in data made available by the Ice-Tethered Profiler program at the Woods Hole Oceanographic Institutes. The acoustic propagation through the double duct system is analyzed using normal mode analysis, through the Kraken normal mode code. A simulated lower duct is introduced in order to isolate only those modes which travel within the lower duct. Propagation through the lower duct is compared to propagation in traditional Arctic sound speed profiles, and for certain ducts distinct increases in propagation strength are shown.

Thesis Supervisor: Henrik Schmidt
Title: Professor of Mechanical and Ocean Engineering
Contents

1 Introduction ........................................... 9
  1.1 Motivation ........................................ 9
  1.2 Objectives ........................................ 9

2 Sound Speed Profiles in the Arctic ..................... 11
  2.1 Classic Arctic sound speed profile .................. 12
  2.2 Double duct Arctic sound speed profile .............. 13
    2.2.1 Strength of Double Duct ...................... 13
  2.3 Ice-Tethered Profiler ............................... 14
  2.4 Double Duct profiles from ITP data ................ 15
    2.4.1 Spatial Extent of Double Duct ................ 17
    2.4.2 Temporal Extent of Double Duct .............. 20

3 Normal Mode Analysis .................................. 23
  3.1 Normal Mode Theory .................................. 23
  3.2 Implementation of Modal Theory ..................... 26
  3.3 Coherent and Incoherent Transmission Loss ........... 30
  3.4 Group and Phase Velocities .......................... 30
  3.5 Waveguide Invariant ................................ 31

4 Methods ................................................. 33
  4.1 Kraken ............................................. 33
  4.2 Key Assumptions and Simplifications ................. 33
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-1</td>
<td>Typical sound speed profiles in polar and non-polar regions [5]</td>
<td>12</td>
</tr>
<tr>
<td>2-2</td>
<td>Ray traces for a typical Arctic sound speed profile, highlighting the numerous ice cover interactions [5]</td>
<td>13</td>
</tr>
<tr>
<td>2-3</td>
<td>Representative double duct Arctic sound speed profile</td>
<td>14</td>
</tr>
<tr>
<td>2-4</td>
<td>ITP schematic [7]</td>
<td>15</td>
</tr>
<tr>
<td>2-5</td>
<td>Location of analyzed ITPs, with locations of double duct profiles marked by squares</td>
<td>18</td>
</tr>
<tr>
<td>2-6</td>
<td>Detail of the locations of ITPs with double duct profiles</td>
<td>19</td>
</tr>
<tr>
<td>2-7</td>
<td>Mean sound speed profile from each ITP which exhibits double duct</td>
<td>20</td>
</tr>
<tr>
<td>2-8</td>
<td>Sound speed profiles from ITP 84, active 2014-2015</td>
<td>21</td>
</tr>
<tr>
<td>2-9</td>
<td>Sound speed profiles from ITP 2, active in 2004</td>
<td>21</td>
</tr>
<tr>
<td>2-10</td>
<td>Drift Locations for ITP 2 and 84</td>
<td>22</td>
</tr>
<tr>
<td>4-1</td>
<td>Full and modeled Arctic environment</td>
<td>34</td>
</tr>
<tr>
<td>4-2</td>
<td>Sound speed profile for full profile and simulated ducts of 1.5 and 4 m/s strength</td>
<td>35</td>
</tr>
<tr>
<td>4-3</td>
<td>Height and Strength of simulated lower duct</td>
<td>36</td>
</tr>
<tr>
<td>5-1</td>
<td>Number of trapped modes in strong and weak full double ducts and in simulated lower ducts</td>
<td>42</td>
</tr>
<tr>
<td>5-2</td>
<td>Mode shapes in full double duct and simulated lower duct at 450 Hz</td>
<td>42</td>
</tr>
<tr>
<td>5-3</td>
<td>Phase and group velocities, in terms of frequency, for the first 5 modes in full and simulated ducts</td>
<td>44</td>
</tr>
</tbody>
</table>
5-4 Group velocity in terms of phase velocity for all modes in the full and simulated profiles, at 900 Hz ........................................... 44
5-5 Mode shapes for the first 12 modes in the full and simulated profiles at 900 Hz .......................................................... 45
5-6 Mode shapes for the highest 3 modes in the simulated duct at 900 Hz 45
5-7 Transmission loss at source depth across frequencies for the full profile 49
5-8 Transmission loss at source depth across frequencies for simulated lower duct with 4.24 m/s strength ................................. 49
5-9 Detection distance in terms of frequency and simulated duct strength 53
Chapter 1

Introduction

1.1 Motivation

The Arctic is an area of increased scrutiny, study, and activity. As global temperatures rise and ice cover in the Arctic reduces, a thorough understanding of the Arctic environment has become even more critical. Autonomous Underwater Vehicles (AUVs) are important tools in the study of the Arctic, allowing for relatively low-cost sensor networks to look at everything from water temperatures and ice health to tracking biologic populations and acoustic targets. Many AUV applications make heavy use of acoustic sensing. By understanding the intricacies of the acoustics in double duct Arctic sound speed profiles, AUVs will be able to operate more intelligently in these regions.

1.2 Objectives

This work presents an in-depth picture of the double duct Arctic sound speed profile, as well as arguments for its importance and utility in Arctic AUV operations. Additionally, it strives to provide a foundational understanding of acoustic propagation within double duct systems, aiding in comparisons between contemporary and historic Arctic acoustic data.
Chapter 2

Sound Speed Profiles in the Arctic

The Arctic ocean provides a special case in ocean acoustics, due to the presence of ice cover [9]. The ice cover means that the temperatures in the water column follow a different pattern than is seen in the open ocean, thereby impacting the sound speed profile. The speed of sound in water depends on temperature, salinity, and pressure. In this work, a simplified version of this dependence is used, namely

\[ c = 1449.2 + 4.6T - 0.055T^2 + 0.00029T^3 + (1.34 - 0.01T)(S - 35) + 0.016z \]  

(2.1)

where the sound speed \( c \) is given in meters per second, temperature \( T \) in degrees centigrade, salinity \( S \) in parts per thousand, and depth \( z \), in meters, brings hydrostatic pressure into the relation [5]. Near the surface, where pressure is low, temperature effects dominate. At deeper depths ocean temperature is relatively stable around the globe, and pressure effects dominate the sound speed [9]. Because temperature effects dominate sound speed near the surface it is clear that the cold ice on the surface of the Arctic will produce a drastically different sound speed profile than is observed in non-polar regions. The difference in typical Arctic and non-polar sound speed profiles is highlighted in figure 2-1.
2.1 Classic Arctic sound speed profile

The key feature of the classic Arctic sound speed profile is that the minimum sound speed resides at the ice-water interface. As acoustic waves travel through ocean environments with a sound speed which varies in depth the waves refract towards depths with lower sound speeds [5][10]. All of the acoustic energy in the classic Arctic environment will interact with the ice cover. Because of the roughness of the ice cover, acoustic interactions with the ice cause significant scattering and therefore significant losses [4].

Because sound in the classic arctic profile is always refracted upwards, it is said to be caught in a surface duct [10]. There is no path the sound can take which will result in it not returning to the surface.
2.2 Double duct Arctic sound speed profile

Parts of the Arctic Ocean have a sound speed profile with a local minimum on the order of 200 meters deep. This unique feature causes some acoustic energy to get stuck around this local minimum, creating another duct in addition to the surface duct discussed in section 2.1, and is referred to as a double duct sound speed profile. Acoustic energy trapped within the lower duct will not interact with the ice, meaning it will not be subjected to losses due to ice scatter. The reduced losses mean that the lower duct has potential for much longer range propagation than the traditional Arctic sound speed profile.

2.2.1 Strength of Double Duct

Figure 2-3 shows an example of a double duct sound speed profile. The local minimum responsible for the lower duct is seen at approximately 185 meters depth. The duct is bounded on the upper side by a local maximum in sound speed, in this case at about 69 meters depth. The lower bound of the duct is the depth below the local minimum depth at which the sound speed is the same as at the upper bound of the duct.

This definition of the bounds of the lower duct presents two primary metrics with which to quantify the duct, namely the height of the duct, defined as the difference in depth between the upper and lower bounds of the duct, and the strength of the
duct, defined as the difference in sound speed between the upper bound of the duct and the minimum sound speed within the duct.

### 2.3 Ice-Tethered Profiler

Evidence of double duct Arctic sound speed profiles is seen in data collected in the Arctic in the past decade. The Woods Hole Oceanographic Institution has been collecting key oceanographic data in the Arctic through their Ice-Tethered Profiler (ITP) program. The ITP program, which first collected data in 2004, consists of a sparse network of profilers throughout the Arctic. Each ITP consists of a surface buoy sitting on the ice. A wire rope is suspended through the ice to a depth of 500 to 800 meters. A profiler with a range of oceanographic sensors travels up and down the wire. Data are transmitted from the profiler to the surface buoy, and then on to shore. Data are available as soon as a few hours after collection. A schematic of an ITP is shown in figure 2-4.

The Ice-Tethered Profiler data were collected and made available by the Ice-Tethered Profiler Program (Toole et al., 2011; Krishfield et al., 2008) based at the...
Woods Hole Oceanographic Institution (http://www.whoi.edu/itp).

Data from the ITPs are released at three levels of processing, raw data, real time data, and archive data. Raw data are released without any smoothing or processing. Real time data are released as they are received, but with some minimal processing. The processing includes interpolating location data for each profile, and averaging the collected data into 2 dB bins. Archive data are released at the conclusion of each ITP’s mission, once the sensor has been recovered. The archive data includes more substantial processing, including secondary sensor calibration and regional conductivity adjustments.

Analysis of the real time data is presented here. The real time data was chosen because a larger number of profilers have this data available. At the time of writing there are at least 20 profilers which are still active, meaning the archive data are not yet available.

2.4 Double Duct profiles from ITP data

Analysis of the available ice-tethered profiler data reveals several instances of double duct sound speed profiles throughout the Arctic. The compressed and minimally processed real time ITP data were downloaded for each profiler, and the salinity, temperature and pressure was read from each file into MATLAB. Sound speed is calculated at each depth using equation (2.1). Over 30,000 total sound speed profiles
from 75 ITPs were calculated from the available data. The mean sound speed profile for each ITP is calculated and examined for a local minimum corresponding with the lower duct. The ITPs are each deployed for several months, and can travel a few hundred kilometers in that time [7]. The lengthy deployments and travel of the ITPs means that some variations can be expected in the data. Analyzing the mean sound speed profile for each ITP does not account for these variations, but the presence of a double duct mean profile shows that particular ITP was sampling a double duct profile for most of its deployment.

2.4.1 Spatial Extent of Double Duct

Double duct profiles are exhibited predominantly in the Beaufort Sea. Figure 2-5 shows the location of each analyzed ITP, highlighting the locations where a double duct sound speed profile is observed. Because the ITPs are mounted directly on the ice, they move with the Arctic ice. For figures 2-5 and 2-6 the average location of each ITP is plotted.

The double duct profiles are not limited to the southern portions of the Arctic, and are seen at locations up to 81 degrees north. Figure 2-6 shows a subset of figure 2-5, displaying only the area of the Arctic where the double ducts exist. At lower latitudes all of the available ITPs in the Beaufort have a double duct. At the more northern latitudes there are a few ITPs in the region which do not have the double duct profile, but the majority of the ITPs still show a double duct.

Figure 2-7 shows the mean sound speed profile for the ITPs with double ducts, corresponding to the mean locations shown in figure 2-6. The double duct profiles vary significantly, with some having much stronger lower ducts than others. There is not a pure correlation between ITP location and double duct strength. ITPs have not been deployed at a wide range of longitudes at latitudes below 80 degrees north, meaning little can currently be said about the east-west locations of double duct profiles.
Figure 2-5: Location of analyzed ITPs, with locations of double duct profiles marked by squares
Figure 2-6: Detail of the locations of ITPs with double duct profiles
Figure 2-7: Mean sound speed profile from each ITP which exhibits double duct

2.4.2 Temporal Extent of Double Duct

In addition to being observed in many locations within the Arctic, the double duct profile is observed throughout the history of the ITP program. Figure 2-8 shows several double duct profiles from ITP 84, collected between October 2014 and January of 2015. Figure 2-9 shows several double duct profiles from ITP 2, collected nearly ten years prior, in August and September of 2004.

The profiles collected from ITPs 2 and 84 are very similar, despite being collected ten years apart. There is very little variation on each ITP through the several months of deployment. This shows that the double duct is a persistent phenomenon.
Figure 2-8: Sound speed profiles from ITP 84, active 2014-2015

Figure 2-9: Sound speed profiles from ITP 2, active in 2004
Figure 2-10: Drift Locations for ITP 2 and 84
Chapter 3

Normal Mode Analysis

Having shown the existence and prevalence of the double duct sound profile in the Arctic, the next step is to detail how it affects acoustic propagation in the Arctic. Specifically, what acoustic energy gets trapped in the lower duct, and how does this trapped energy behave. There are several techniques used to think about and model ocean acoustics, such as ray-tracing, wavenumber integration, parabolic equations, and normal modes. Normal mode theory allows for neat discretization of acoustic energy propagating in the ocean, and was therefore chosen as the primary analysis method in this work. Normal modes also provide a simple method to separate out the lower duct and carefully analyze only the sound which gets trapped there.

3.1 Normal Mode Theory

Normal mode theory is a method of solving the acoustic wave equation by breaking the problem down into a set of propagating modes. These modes are similar to the shapes taken by a plucked string. Continuing the analogy of a vibrating string the length of the string corresponds to water depth, the thickness of the string to sound speed, and the plucking of the string to an acoustic source [6]. A complete acoustic field can be calculated by summing the full set of weighted modes.

For a range-independent acoustic medium (where the sound speed varies with
depth only) the acoustic wave equation governs the field solution:

\[ \nabla \left( \frac{1}{\rho} \nabla P \right) - \frac{1}{\rho c^2(z)} \frac{\partial^2 P}{\partial t^2} = -s(t) \frac{\delta(z - z_s)\delta(r)}{2\pi r} \]  

where \( s \) is a point source strength, \( c \) is sound speed, \( \rho \) is water density, and \( P \) is acoustic pressure in terms of depth \( (z) \), range \( (r) \), and time \( (t) \) [6].

Several boundary conditions and assumptions are needed to transform this problem into a workable form. First, assume a pressure release boundary at the top of the water column (depth \( 0 \)) and a perfectly rigid boundary at the bottom (depth \( D \)).

\[ P(r, 0, t) = 0 \quad \frac{\partial P}{\partial z}(r, D, t) = 0 \]  

Next, assume the source is placed in an infinite medium, so there are no reflected signals traveling radially inwards toward the source. Finally, simplify the point source to be a continuous, single frequency \( (\omega) \) source [9].

\[ s(t) = e^{-i\omega t} \]  

A single frequency source means that the pressure field will be harmonic with the same frequency as the source.

\[ P(r, z, t) = p(r, z)e^{-i\omega t} \]  

Putting these conditions back into equation (3.1) yields the Helmholtz equation [6].

\[ \frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial p}{\partial r} \right) + \rho(z) \frac{\partial}{\partial z} \left( \frac{1}{\rho(z)} \frac{\partial p}{\partial z} \right) + \frac{\omega^2}{c^2(z)} p = -\frac{\delta(z - z_s)\delta(r)}{2\pi r} \]  

The Helmholtz equation, as stated in (3.5) includes a source term on the right hand side. The solution to the unforced Helmholtz equation must be, through separation of variables, of the form

\[ p(r, z) = \Phi(r)\Psi(z) \]
which allows the unforced Helmholtz equation to be written as

\[
\frac{1}{\Phi} \left[ \frac{1}{r} \frac{d}{dr} \left( \frac{d\Phi}{dr} \right) \right] + \frac{1}{\Psi} \left[ \frac{\rho(z)}{d} \frac{d}{dz} \left( \frac{1}{\rho(z)} \frac{d\Psi}{dz} \right) + \frac{\omega^2}{c^2(z)} \Psi \right] = 0 \tag{3.7}
\]

where the two bracketed expressions are functions of only \( r \) or \( z \) [5]. The only way for equation (3.7) to be satisfied is if each bracketed expression is a constant. Using separation of variables, with a separation constant of \( k_{rm} \) yields the modal equation

\[
\rho(z) \frac{d}{dz} \left[ \frac{1}{\rho} \frac{d\Psi_m(z)}{dz} \right] + \left[ \frac{\omega^2}{c^2(z)} - k_{rm}^2 \right] \Psi_m(z) = 0 \tag{3.8}
\]

where \( \Psi_m(z) \) is the particular mode function corresponding with each separation constant \( k_{rm} \) [9]. The boundary conditions from (3.2) can be re-written in terms of the modal equation as

\[
\Psi(0) = 0 \quad \frac{d\Psi}{dz} \bigg|_{z=D} = 0 \tag{3.9}
\]

Equation (3.8) with boundary conditions (3.9) form a classic Sturm-Liouville problem. The Sturm-Liouville problem is well understood, and has several important properties [2]. From these properties we know that the modal equation has an infinite number of solutions, with each solution corresponding to a mode, analogous to the shape of a vibrating string. Each mode can be described by a mode shape function \( (\Psi_m(z)) \) which is an eigenfunction, and a corresponding horizontal propagation constant \( (k_{rm}) \) which is an eigenvalue [5]. The modes of the Sturm-Liouville problem form a complete set, are normalized, and are orthogonal [6].

\[
p(r, z) = \sum_{m=1}^{\infty} \Phi_m(r) \Psi_m(z) \tag{3.10}
\]

\[
\int_0^D \frac{\Psi_m^2(z)}{\rho(z)} dz = 1 \tag{3.11}
\]

\[
\int_0^D \frac{\Psi_m(z)\Psi_n(z)}{\rho(z)} = 0 \quad \text{for} \quad m \neq n \tag{3.12}
\]
Combining these three properties, equation (3.5) can be rewritten as

\[
\frac{1}{r} \frac{d}{dr} \left[ r \frac{d\Phi_n(r)}{dr} \right] + k_{mn}^2 \Phi_n(r) = -\frac{\delta(r)\Psi(z)}{2\pi \rho(z_s)} \tag{3.13}
\]

where \(z_s\) represents the depth of the omnidirectional source [5]. The solution to equation (3.13) is known, and is given in terms of a Hankel function as

\[
\Phi_n(r) = \frac{i}{4\rho(z_s)} \Psi(z_s) H_0^{(1,2)}(k_{mn}r) \tag{3.14}
\]

A Hankel function of the first kind is chosen to meet the condition that energy is radiating outward from the source only [9]. Using the asymptotic Hankel approximation, the total pressure field can be given in terms of normal modes as

\[
p(r, z) \approx \frac{i}{\rho(z_s)\sqrt{8\pi r}} \sum_{m=1}^{\infty} \Psi_m(z_s) \Psi_m(z) \frac{e^{i k_m r_m}}{\sqrt{k_m}} \tag{3.15}
\]

Equation (3.15) does not account for losses due to material absorption. There are several methods for expressing this volume attenuation, with several choices of units, including dB per wavelength, dB per meter, or dB per meter hertz [5]. Thorpe plane wave attenuation provides a standard form of frequency dependent (with \(f\) in kHz) attenuation in dB per meter [6].

\[
\alpha = \frac{40 f^2}{4100 + f^2} + \frac{0.1 f^2}{1 + f^2} \tag{3.16}
\]

### 3.2 Implementation of Modal Theory

Equation (3.15) gives an expression for acoustic pressure as a function of range and depth, in terms of solutions to modal Sturm-Liouville problem, as stated in equation (3.8). In order to effectively use this expression, computational methods must be selected which allow for efficient computation of the modal function.

The first step is to limit the scope of the problem by only considering the modes of interest. In most cases the logical place to limit the horizontal wave number of the calculated modes is at the maximum and minimum sound speeds in the modeled
In equation (3.17), \( c_b \) refers to the speed of sound in the material under the water column, which in most cases is much higher than the speed of sound in the water. If the bottom boundary condition is a vacuum or rigid boundary, or has a sound speed lower than the maximum water sound speed, \( c_b \) should be replaced by the maximum sound speed [9].

Next, a discretization of the water column must be selected. As a rule of thumb, there should be at least 10 points per wavelength for the highest mode [9].

A Sturm sequence is used to determine the total number of modes within the range of interest. This is done by constructing a matrix \( C \), which is a tri-diagonal matrix containing solutions to equation (3.15) at a specific \( k_r \) value [5].

\[
C = \begin{bmatrix}
d_0 & e_1 & & & \\
e_1 & d_1 & e_2 & & \\
& \ddots & \ddots & \ddots & \\
e_{N-1} & d_{N-1} & e_N & \\
e_N & d_N
\end{bmatrix}
\]

\[
e_j = \frac{1}{h\rho} \quad j = 1...N
\]

\[
d_j = \frac{-2 + h^2 \left[ \frac{\omega^2}{c^2(z_j)} - k_r^2 \right]}{h\rho} \quad j = 1...N - 1
\]

\[
d_0 = \frac{-2 + h^2 \left[ \frac{\omega^2}{c^2(z_j)} - k_r^2 \right]}{2h\rho} + \frac{f^T(k_r^2)}{g^T(k_r^2)}
\]

\[
d_N = \frac{-2 + h^2 \left[ \frac{\omega^2}{c^2(z_j)} - k_r^2 \right]}{2h\rho} + \frac{f^B(k_r^2)}{g^B(k_r^2)}
\]
In equations (3.20) through (3.22) \( j \) is the index through points in depth, \( h \) is the distance between the equally spaced discrete points in depth, and functions \( g \) and \( f \) define the boundary conditions at the top and bottom of the water column.

The matrix \( C \) can be re-written in terms of \( \lambda \), where \( \lambda = k_r^2 \), so that it contains terms \( a_k(\lambda) \), where \( a_k(\lambda) = d_k(\lambda) + \lambda \) [5].

\[
C(\lambda) = \begin{bmatrix}
\lambda - a_1(\lambda) & -e_1 \\
-e_1 & \lambda - a_1 & -e_2 \\
& \ddots & \ddots \\
& & -e_{N-1} & \lambda - a_{N-1} & -e_N \\
& & & -e_N & \lambda - a_N(\lambda)
\end{bmatrix}
\]  

(3.23)

The Sturm sequence is defined, using components of the matrix \( C(\lambda) \), as

\[
p_k(\lambda) = [\lambda - a_{k-1}(\lambda)] p_{k-1}(\lambda) - e_{k-1}^2 p_{k-2}(\lambda) \quad k = 1...N + 1
\]

(3.24)

with starting values \( p_{-1} = 0 \) and \( p_0 = 1 \) [5]. The number of real eigenvalues larger than \( \lambda \) is defined as the number of zero crossings in the Sturm sequence \( p_0(\lambda)...p_{n+1}(\lambda) \).

The first calculation of the Sturm sequence gives the total number of modes in the range of \( k_r \) values of interest. By bisecting this range and re-calculating the Sturm sequence for a variety of values of \( \lambda \), the ranges of \( k_r \) values which contain only one mode is calculated [9].

Once the range of \( k_r \) values containing only one mode are found, an estimate of the actual \( k_r \) value associated with each mode is calculated. This is done using Newton’s method or other root finding scheme, in association with the Sturm sequence and an inverse iteration [5].

Starting with some estimate \( \kappa \) of the eigenvalue associated with the \( n \)th mode, which is some small value \( \epsilon \) away from the actual eigenvalue, such that \( \kappa = \lambda_m - \epsilon \), the associated eigenvector \( (\Psi_m) \) by definition satisfies the equation

\[
[A(\lambda_m) - \lambda_m I] \Psi_m = 0
\]

(3.25)
where the matrix $A(\lambda_m)$ is the a components of the matrix $C(\lambda)$ [9].

The inverse iteration

$$ w_k = [A(\lambda_m) - \kappa I]^{-1} w_{k-1} \quad k = 1, 2 \ldots \infty $$

(3.26)

is then applied. Because the inverse matrix $(A - \kappa I)^{-1}$ has the same eigenvectors as $A$, the estimated eigenvector $w_k$ will converge to $\Psi_m$ as $k$ goes to infinity [5]. The inverse iteration refines the estimate of the eigenvalue, as

$$ \kappa + \frac{(w_r)_{k-1}}{(w_r)_k} \rightarrow \lambda_m $$

(3.27)

as $k$ goes towards infinity [5].

Upon refining the estimate of the eigenvalue, an estimate for the error is calculated using the Richardson extrapolation. The Richardson extrapolation also refines the estimate of each eigenvalue. The calculated eigenvalues are a function of $h$, the increment between mesh points, with $k_0$ being the true (continuous) eigenvalue [5].

$$ k_0^2(h) = k_0^2 + b_2 h^2 + b_4 h^4 + ... $$

(3.28)

Fitting a curve to equation (3.28) across a range of mesh increments provides a refined estimate of the eigenvalue, and an estimate of the error of the previous estimate of the eigenvalue. While the error estimate is unacceptably high, the previous steps are all repeated with a finer vertical mesh.

Once suitable estimates of the eigenvalues are found, and the modes (corresponding eigenvectors) are found, the modes are normalized [5].

$$ \int_0^D \frac{\Psi_m^2(z)}{\rho(z)} dz = 1 $$

(3.29)

This discrete computation is performed using the trapezoidal rule.

Finally the modes are summed using equation (3.15) to yield the full field acoustic pressure.
3.3 Coherent and Incoherent Transmission Loss

In order to usefully display meaningful information about the complex pressure field from equation (3.15) the transmission loss is calculated. Transmission loss is the difference, in dB, between the pressure from some acoustic source at a reference distance (usually one meter) and the pressure from the same source at an arbitrary point \((r, z)\) [9]. There are two types of transmission loss, coherent and incoherent, shown in equations (3.30) and (3.31), respectively [5].

\[
TL(r, z) \approx -20 \log \left| \frac{1}{\rho(z_s)} \sqrt{\frac{2\pi}{r}} \sum_{m=1}^{\infty} \frac{\Psi_m(z_s) \Psi_m(z) e^{ik_{rm}r}}{r \sqrt{k_{rm}}} \right| \quad (3.30)
\]

\[
TL_{inc}(r, z) \approx -20 \log \frac{1}{\rho(z_s)} \sqrt{\frac{2\pi}{r}} \sqrt{\left| \sum_{m=1}^{\infty} \frac{\Psi_m(z_s) \Psi_m(z) e^{ik_{rm}r}}{r \sqrt{k_{rm}}} \right|^2} \quad (3.31)
\]

Coherent transmission loss shows detailed modal interference patterns, while incoherent transmission loss is smoothed, and does not show these patterns. This makes incoherent transmission loss more appropriate in cases where environmental properties are not precisely known, or when models are being compared to data which has been averaged over frequency [5].

3.4 Group and Phase Velocities

Each mode has associated phase and group velocities. These velocities are in the horizontal direction, away from the acoustic source. The phase velocity \(v\) comes directly from the wavenumber definition [5].

\[
v_m = \frac{\omega}{k_{rm}} \quad (3.32)
\]

The group velocity \(u\) depends on the change in source frequency as

\[
u_m(\omega) = \frac{d\omega}{dk_{rm}} \quad (3.33)
\]
A simple finite difference formula is used to compute the group velocity [6].

\[ u_m \approx \frac{(\omega + \Delta \omega) - \omega}{k_{rm}(\omega + \Delta \omega) - k_{rm}(\omega)} \]  

(3.34)

The phase velocity ranges from the speed of sound in the material to infinity. It represents the horizontal speed of a peak in the plane-wave approximation of a mode [5]. Phase velocity increases towards infinity for steeper launch angles of the propagating wave. The group velocity is the speed of energy transport of a mode.

### 3.5 Waveguide Invariant

In waveguides (alternatively referred to as ducts) plots of transmission loss in terms of range and frequency exhibit distinct striations. These striations can be used to characterize the waveguide through a single number, or waveguide invariant [5]. The waveguide invariant can be used to estimate the range to a broadband acoustic source using only a single mobile acoustic receiver [3].

The waveguide invariant \( \beta \) is defined as

\[ \frac{1}{\beta} = -\left( \frac{v}{u} \right)^2 \frac{du}{dv} \]  

(3.35)

in terms of modal phase and group velocities [5].

In practice, the waveguide invariant is calculated by plotting the group and phase velocities for several modes, and then grouping a few modes with similar characteristics, and then calculating a best fit slope and plugging that back into equation (3.35) along with the mean phase and group velocities of the selected modes [9].

The waveguide invariant can also be expressed as

\[ \beta = \frac{r}{\omega} \frac{d\omega}{dr} \]  

(3.36)

where \( \frac{d\omega}{dr} \) corresponds to the slope along the striations of transmission loss [5].
Chapter 4

Methods

4.1 Kraken

The Kraken normal mode code is a powerful tool written by Mike Porter, and released with the Acoustics Toolbox. It can handle a wide range of complex acoustic environments, including many interfaces between acoustic and elastic media, range-dependent environments, and ocean surface scattering [6].

The Kraken code is run through MATLAB. It takes a range of number of input files including environment files which specify the sound speed and boundary conditions, and field files which specify some key parameters for the calculation of the total pressure field.

4.2 Key Assumptions and Simplifications

To explore the modes trapped in the lower duct of the Arctic double duct profile it is not necessary to model the full Arctic environment. Several complex phenomenon such as reflections and scattering off the ice do not need to be included in the model. This is because modes which are fully trapped within the lower duct will not reach the ice surface or the bottom [9]. Because these sounds do not reach these interfaces, the boundary conditions are irrelevant. For simplicity, a vacuum boundary is used at both the top and bottom surface.
There are sounds emanating from the modeled source which will leave the lower duct, interact with the surface or bottom, or refract upwards, and then re-enter the lower duct. A normal mode approach examining the modes within the lower duct will not account for these elements of the pressure field. However, at sufficient distances all of the sounds which leave the lower duct will be sufficiently decayed, through ice interactions bottom loss and volume absorption, to no longer have a meaningful contribution to the total field [9].

4.2.1 Simulated Lower Duct

The modes trapped within the lower duct are isolated by limiting the phase speeds of interest. Modes trapped within the lower duct must have a phase speed which is between the limiting sound speeds of the lower duct [9]. Unfortunately the sound speeds within the lower duct are not found there exclusively. The same sound speed can exist at multiple depths in the sound speed profile. Specifically, in the Arctic profiles examined here, the sound speeds of the lower duct also exist closer to the surface. This means that limiting the phase speeds of interest will include some modes not within the lower duct.

To get around this, a simulated lower duct is created. The simulated duct consists of real ITP data in the lower duct, with a constant sound speed, equal to the maximum...
sound speed in the lower duct, for all depths outside of the lower duct.

The isovelocity profile outside of the duct means that limiting the phase speeds to the minimum and maximum sound speeds of the lower duct will successfully eliminate modes outside of the lower duct.

This method of creating simulated lower ducts allows ducts of various duct strengths to be easily examined. By simply adjusting the speed of the isovelocity portion of the simulated duct the strength of the duct changes. Figure 4-2 shows the full sound velocity profile from ITP 84, as well as simulated ducts of various strengths.

Using real data as the base for simulated lower ducts means that the height of the simulated ducts change with the strength of the ducts. Figure 4-3 shows the relationship between the height and strengths of the simulated lower ducts, superimposed on the height and strength of the ducts observed in the ITP data. The unevenness in the relationship between height and strength of the simulated ducts is due to the somewhat rough nature of the full sound speed profile. The height and strength of the simulated ducts fit the data from observed ducts. This means that it is not necessary
to adjust the height and strength of the simulated ducts separately. Adjusting only
the strength provides simulated ducts which mimic the properties of the observed
ducts well.

4.3 Creating Environment Files

In order to use Kraken, environment files are generated specifying the sound speed,
boundary conditions, attenuation, number of mesh points, and other important en-
vironmental parameters.

A separate environment file is needed for each frequency. In order to do sufficiently
broadband modeling kraken is run using frequencies between 5 and 1,000 Hz. A
maximum frequency of 1 kHz is used because at higher frequencies the evanescent
tails of modes trapped in the lower duct are significant enough that they interact with
the ice cover.

Kraken allows for modeling of layered bottom media, which is not needed for this
work. Only one acoustic media is used, representing the water column. The number
Table 4.1: Values used in simulation of the Arctic double duct sound speed profile

The 'options' field allows for selecting the interpolation of sound speeds, top boundary condition, and attenuation units. N2-linear interpolation is used for the sound speeds. This interpolation choice is more than sufficient, and avoids potential problems that some interpolation schemes have with sharp curves in sound speed [6]. A vacuum boundary is used at the top boundary. As discussed in section 4.2 the boundaries are sufficiently far from the edge of the lower duct at the frequencies examined, and therefore have little impact on the overall solution. The vacuum boundary condition is computationally simple, but there is no practical difference in this case between the vacuum or rigid boundary condition [9]. The Thorpe attenuation formula is used to calculate loss due to material absorption. While the Thorpe formula does not provide good loss estimates in sediments, it provides a frequency dependent loss estimate in the water column, which is all that is required in this work [6].

The number of mesh points depends on the frequency being used as

\[ \text{nmesh} \geq \frac{10fZ}{c_{low}} \]  \hspace{1cm} (4.1)
where $f$ is the frequency in Hz, $Z$ is the total depth in meters, and $c_{low}$ is the lowest sound speed in the area of interest, in meters per second. The $c_{low}$ value changes depending on the sound speed profile being used, but always corresponds to the local minimum sound speed in the lower duct. This ensures that the starting mesh includes at least the required 10 points per wavelength.

An interface roughness of 0, implying a perfectly smooth interface, is used. This is chosen for computational simplicity. Again, this is the easy choice, as the sound trapped within the lower duct never reaches the boundary interfaces.

A water depth of 721 meters is used. This is plenty of depth to allow for sufficient depth below the lower duct. While the Arctic is much deeper, the sound trapped in the lower duct does not penetrate to these depths, and there is no need to calculate acoustic pressures at lower depths.

The choice of bottom boundary condition follows the same logic as the top boundary, so a vacuum is again chosen.

Setting the upper and lower phase speed limits ensures that only the modes trapped within the lower duct are calculated. The lower phase speed limit is the sound speed in the local minimum in the lower duct. The upper phase speed limit is the sound speed in the local maximum at the top boundary of the lower duct.

One source, at 180 meters depth is used. This depth is within the lower duct for all of the analyzed ITPs. 750 receiver depths, between 18 and 700 meters deep, are used. There is no added computational cost, aside from storage, to find the field at several depths once the modes have been calculated [6]. Looking at so many receiver depths ensures that several will be available within the lower duct.

### 4.4 Detection Distance

In order to provide a metric for the usefulness of the double duct profile, a detection distance is calculated. This distance is the furthest distance at which a single acoustic
receiver can detect a simulated acoustic source. The sonar equation states

\[ S(r, z) = SL - TL(r, z) \]  \tag{4.2} \]

where \( S(r, z) \) is the signal level at some point, \( SL \) is the source level, and \( TL(r, z) \) is the transmission loss between the source and receiver, with all levels in dB [5].

Looking along a single depth (the source depth) and using the incoherent transmission loss to remove tight dependence on specific modal interactions, it becomes easy to calculate the signal level at any range along the source depth. The range at which the signal level is equal to the ambient noise level, providing a signal to noise ratio of 0 dB, is said to be the detection distance. In this calculation a source level of 125 dB and an ambient noise level of 50 dB are used.
Chapter 5

Results

Results are presented here for a full double duct profile and for simulated lower ducts. The full profile is the mean sound speed profile from ITP 84, which has a lower duct strength of 4.24 m/s. Some results are also presented for the mean sound speed profile from ITP 21, which is a much weaker double duct with a strength of only 0.21 m/s. The simulated ducts are created using the methods described in section 4.2.1, using the mean sound speed profile from ITP 84 as a base.

5.1 Trapped Modes

Figure 5-1 shows the number of modes trapped in the full and simulated ducts as a function on frequency. The red lines show the full profiles, while the blue lines show the number of modes trapped in the simulated profiles of various strengths. As expected, the number of modes trapped in the duct increases with duct strength. When looking only at the lower duct (through the simulated profiles) the number of trapped modes only increases with frequency. However, when looking at the full profile there are a few instances where the number of trapped modes decreases with an increase in frequency. These jumps can be attributed to modes in the full profile which propagate outside of the lower duct.

Figure 5-2 shows the shapes of all the trapped modes in the strong full profile and a simulated profile of the same strength at 450 Hz. The horizontal lines mark the
Figure 5-1: Number of trapped modes in strong and weak full double ducts and in simulated lower ducts

Figure 5-2: Mode shapes in full double duct and simulated lower duct at 450 Hz
boundaries of the lower duct. The full profile has 9 trapped modes, while the simulated profile has 7. The mode numbers in figure 5-2 correspond the mode numbers for the full profile. The mode numbers for the simulated profile have been shifted so that they are plotted with the corresponding mode shape from the full profile. Modes 1 through 4 and 6 through 8 exhibit close agreement between the full and simulated profiles. These modes exist primarily in the lower duct. Mode 5 exists only above the lower duct, and therefore does not have a similarly shaped mode from the simulated profile. Mode 9 has significant amplitude both inside and above the lower duct, so there can not be a corresponding mode from the simulated profile.

5.1.1 Phase and Group Speeds

The phase and group velocities for the first 5 modes at 450 Hz, in both a full double duct profile and simulated profile of the same strength, are shown in figure 5-3. The full profile is shown in red, and the simulated duct is shown in blue. The simulated profile phase speeds are smooth, trending towards the material sound speed at the source depth. The phase speeds for the full profile follow similar trajectories as the simulated profile, but are jumping between the curves for various modes. These jumps are caused by the mismatch in mode number between the full and simulated profiles, due to the modes in the full profile which are outside the lower duct.

The group speeds exhibit larger differences between the full and simulated profiles. As frequency increases, the full and simulated profiles trend together more. There are frequencies at which the group speed for the full profile dips to a much slower value than is seen in the simulated profiles. These frequencies correspond the frequencies at which the mode numbers are changing, and the full profile phase curves are jumping between the different simulated profile phase curves. The bottoms of the dips in the full profile group speed appear to form coherent curves, which correspond to the group speeds of the modes which propagate above the lower duct, such as mode 5 from figure 5-2.

Figure 5-4 shows the relationship between group and phase velocity for the trapped modes in the full profile and simulated duct, at 900 Hz. For modes 1 through 15,
Figure 5-3: Phase and group velocities, in terms of frequency, for the first 5 modes in full and simulated ducts

Figure 5-4: Group velocity in terms of phase velocity for all modes in the full and simulated profiles, at 900 Hz
Figure 5-5: Mode shapes for the first 12 modes in the full and simulated profiles at 900 Hz

Figure 5-6: Mode shapes for the highest 3 modes in the simulated duct at 900 Hz
with the exception of modes 4 and 11, the full profile and simulated duct of the same strength match very closely. As seen in figure 5-5, modes 4 and 11 are the two modes in the full profile which exist above the lower duct. Seeing these low group speeds from the modes above the lower duct support that these surface channel modes are responsible for the dips in group speeds seen in figure 5-3.

The rest of the mode shapes, through the 12 shown, match very closely between the full and simulated profiles. The relationship between group speed and phase speed is used to calculate and describe the waveguide invariant, as described in section 3.5. The close agreement seen between the full and simulated profiles for many of the modes in figure 5-4 implies that these modes can be used to calculate a waveguide invariant for the lower duct. Because the behavior of these modes match for the simulated lower duct and the full sound speed profile, it is clear that the behavior is coming exclusively from the lower duct.

The mode shapes for the highest three modes in the simulated duct are shown in figure 5-6. For these modes, the group speed starts to trend upwards above the expected values. The mode shapes show significant evanescent tails above and below the bounds of the lower duct. For these high modes, the surface boundary is too close, meaning that the modes begin to behave differently than modes purely trapped within the lower duct, and deviate from the constant slope relationship expected from the waveguide invariant calculation.

<table>
<thead>
<tr>
<th>Profile</th>
<th>Strength (m/s)</th>
<th>Waveguide Invariant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>4.24</td>
<td>4.586</td>
</tr>
<tr>
<td>Simulated</td>
<td>4.24</td>
<td>4.683</td>
</tr>
<tr>
<td>Simulated</td>
<td>3</td>
<td>4.193</td>
</tr>
<tr>
<td>Simulated</td>
<td>2</td>
<td>3.985</td>
</tr>
</tbody>
</table>

Table 5.1: Waveguide invariants for full profile and simulated ducts of various strengths

Table 5.1 shows the calculated waveguide invariants for various profiles. They were calculated from the modes trapped at 900Hz. At strengths below 2 m/s, even at 900 Hz, there are not enough trapped modes to make a meaningful waveguide
5.2 Transmission Loss within Lower Duct

Figures 5-7 and 5-8 show the transmission loss at the source depth (180 meters) as a function of range and frequency for the full profile and simulated lower duct of the same strength. In both figures, clear striations are visible. Overlaid on these plots are white lines showing the slope calculated using equation (3.36) and the corresponding waveguide invariant.

For the full profile, the striations match the expected slope from the waveguide invariant well. The alignment is not as strong for the simulated waveguide, particularly at low frequencies. Note that the overlaid lines are not supposed to correspond with the peaks of the transmission loss striations, but should simply run parallel to them. The striations in the full profile transmission loss exhibit some fuzziness at certain frequencies. In figure 5-7 this is particularly evident around 280 Hz and 400 Hz. These correspond to frequencies where new modes, particularly modes which are not within the lower duct, begin to propagate.

The fact that the waveguide invariant matches the modal interference pattern in the lower duct supports the idea that the lower duct exists as a waveguide within the full Arctic profile, and that it can be harnessed for long range, low loss propagation.
Figure 5-7: Transmission loss at source depth across frequencies for the full profile

Figure 5-8: Transmission loss at source depth across frequencies for simulated lower duct with 4.24 m/s strength
5.2.1 Detection Distance

Figure 5-9 shows the distance at which a single hydrophone can detect a modeled acoustic source, in terms of simulated duct strength and source frequency. The detection distance is calculated as described in section 4.4. The white contour line represents a threshold detection distance of 24 km from a traditional Arctic profile. This threshold is not a particularly accurate one. It is calculated by using Kraken to calculate the transmission loss at the source depth using sound speed profile data from ITP 29, which does not exhibit a double duct. For this more traditional Arctic profile ice interactions become a critical loss mechanism. The ice scatter is modeled using Soft-boss Twersky scatter, with a bump density of 0.092 ridges per kilometer, a first principal radius of 8.2 meters, and a second principal radius of 5.1 meters, the default values suggested for modeling generic ice scatter [6].

Propagation through the double duct performs best at high duct strength, as expected. This is because higher strength ducts are able to trap more modes. The most notable feature in figure 5-9 is the drop off in performance between 200 and 300 Hz. The best performance comes at around 550 Hz. The fluctuations in performance are due to the interplay between the number of modes trapped in the lower duct and the volume attenuation of the modes which are trapped. More trapped modes means less transmission loss in the lower duct. The higher the frequency, the more modes get trapped in the lower duct, all else held constant, as seen in figure 5-1. However, higher frequencies experience more losses due to volume attenuation while traveling through sea water [9]. Tracing up frequency at a single strength, performances increases when another mode becomes trapped in the lower duct, and then levels off or decreases as frequency increases, until an additional mode becomes trapped.
The threshold value shown in figure 5-9 compares a full traditional Arctic profile to a simulated lower duct profile, which is limited to isovelocity outside of the lower duct. The number of paths sound can take from the source to receiver in the simulated profile is much lower than the number of paths possible for the full profile. These factors combine to create a threshold which is inappropriately high, and transmission through the lower duct outperforms a traditional Arctic profile more than is shown in figure 5-9. In the limiting case, a double duct profile with a strength of zero approaches a traditional Arctic profile.
Chapter 6

Conclusions

The analysis and understanding of the Arctic double duct sound speed profile system is the first step in more effective autonomous vehicle operations in the Arctic. The double duct profile exists in a large portion of the Arctic, and is a persistent phenomenon.

Calculating acoustic pressure from modes propagating within the lower duct allows the acoustic properties of the sound speed feature to be studied and understood. The double duct system exhibits characteristics similar to well known and often studied duct systems, and can be adequately summarized by a waveguide invariant.

At its simplest, modal analysis does not allow for the complete isolation of modes which propagate only within the lower duct. This is addressed by constructing an artificial sound speed profile, which is an isovelocity profile with the addition of a local minimum of the constructed lower duct. This simulated lower duct successfully isolates the modes propagating within the lower duct of the double duct system.

For receiver depths within the lower duct, the sound field is dominated by modes which are trapped in the lower duct. This feature of the double duct system highlights the importance of understanding and leveraging the lower duct while operating in and studying the Arctic.
6.1 Future Work

6.1.1 Ice Scatter Modeling

Autonomous underwater vehicles rely heavily on acoustics for communications. These communications generally happen in the 5-10 kHz range [1]. These frequencies are outside the frequencies studied in this work. Accurate modeling of these higher frequencies requires detailed ice scatter models to be included. These models are required because the lower duct in Arctic double duct profiles is relatively close to the surface, and the evanescent tails from higher frequency modes in the lower duct extend out far enough to interact non-negligibly with the ice cover. This phenomenon is seen even as low as 1kHz.

The addition of ice scatter modeling should also include range dependent modeling. This will allow for partial ice cover to be added, for situations when the area of interest is near the edge of the Arctic ice cover. From the nature of the instruments used, the ITPs only provide sound speed data under ice. The inclusion of range dependent modeling goes hand in hand with additional data collection of sound speed data near the ice edge to look at the development of the double duct profile under and on either side of the marginal ice zone.

6.1.2 Array Processing

The detection distance presented in this work is based off a simple single receiver system. By using an array of multiple sensors an additional array gain can be leveraged, increasing the detection distance, thereby aiding in target tracking exercises. Knowledge of the spatial extent of the double duct, particularly in depth, will be very helpful in determining the limiting cases for leveraging array gain in the double duct system, and in determining ideal array positioning.
6.1.3 ICEX16

The Laboratory for Autonomous Marine Sensing Systems (LAMSS) will be participating in ICEX16, a US Navy Arctic experiment exercise. We will be operating a AUV towing an acoustic array under the ice. While the exact location of the experiment is not yet known, it will be in the Beaufort Sea, in the same region as the double duct profiles analyzed here. This provides a very exciting opportunity to collect specific acoustic data in a double duct system, provide experiment validation for this work, and test yet to be developed methods for leveraging the acoustic properties of the lower duct for target tracking applications.

6.1.4 Ambient Ice Noise

One of the key portions of the ICEX16 experiment is using a towed array to sample ambient noise from the ice cover. The understanding of how sound behaves around double duct sound speed profiles will be key in the autonomous sampling and processing of this ambient ice noise data. Additionally, LAMSS has acoustic data from the Sea Ice Mechanics Initiative experiments of the early 1990’s. A comparison of these data sets will hopefully yield important information not only about the current state of the Arctic, but a more thorough understanding of the changes the Arctic is undergoing.
Bibliography


Appendix A

Code for Analysis of ITP Data

clear all
close all

%% ITP.reader.allCasts.m
% Load data from WHOIs ITP, and calculate the resulting sound speed
% profile. Working with level 2 data, where WHOI researchers have already
% interpolated for salinity.
%
% Check if each file exists, in order to read all available data. Gives
% ability to plot all SVP for each ITP, and take average SVP, location,
% etc..
%
% Plot multiple sound velocity profiles on the same plot, each labeled
% with their record date.
%
% Thomas Howe  thomhowe@mit.edu  (413) 320-2050

%% Declare Extent of Search

% NOTE: The search phase of this code need only be run once to find all of
% the valid data files over the search parameter. Once it has been run once
% and the data has been read in, simply save a .mat file with the
% appropriate fields and load that, while commenting out the following
% section

% Comment from here for search

allITPs = [1:31 33:87];
allCasts = 1:800;

% Initialize Cell Arrays
allFiles = cell(length(allITPs),length(allCasts));
offset = 12;
goodITPs = [];
goodCasts = [];

% Create array of all possible file names
for j = 1:length(allITPs)
    for c = 1:length(allCasts)
        allFiles{j,c} = sprintf('tp%igrd%04i.dat',allITPs(j),allCasts(c));
    end
end

% Check if file exists, if so add appropriate numbers to 'good' values.
% Read in Data & Header, calculate sound speed.
for j = 1:length(allITPs)
    for c = 1:length(allCasts)
        if exist(allFiles{j,c},'file') == 2
            % Check if ITP number has been added, if not, add it
            added = find(goodITPs == allITPs(j));
            [P, T, S] = importfile(allFiles{j,c});
            if isempty(added) && length(P) >= 350
                goodITPs = [goodITPs allITPs(j)];
                itpIndex = find(goodITPs == allITPs(j));
                goodCasts{itpIndex} = [];
                SoundSpeed{itpIndex} = [];
                disp(['Searching Through ITP ',num2str(allITPs(j))])
            end
        end
    end
end
if length(P) >= 350 \% Condition for cast to be deep enough to be usable
% Add cast number to good cast
goodCasts{itpIndex} = [goodCasts{itpIndex} allCasts(c)];
castIndex = find(goodCasts{itpIndex} == allCasts(c));

% Truncate Values. Some of the casts have fewer points,
P = P(1:350);
T = T(1:350);
S = S(1:350);

% Calculate Sound Speed
SoundSpeed{itpIndex}(:,castIndex) = ...
calcSoundSpeed(T, S, max(P*1.019716))';

% Calculate Depth
di{itpIndex,castIndex} = P*1.019716;

% Read in Header
itpHeader(itpIndex,castIndex) = Level2.Header.Reader(allFiles{j,c});

end
end
end
end

% Comment to here for search

% Once search has been completed, simply add the following line:
load('ITPdata') \%where the filename is changed as appropriate

offset = 12; \% Offset value for plotting SVPs

\% Loop over ITPs to process sound speed for each.
for j = 1:length(goodITPs)
% Calculate Mean Sound Speed
meanSoundSpeed(:, j) = mean(SoundSpeed{j}, 2);
descr{j} = ['ITP ', num2str(goodITPs(j))];

% Find Local Minimum in SVP
SVP = -meanSoundSpeed(50:200, j);
[~, min.locs] = findpeaks(SVP, 'SortStr', 'descend', 'NPeaks', 1);
if ~isempty(min.locs)
    mid.min.z{j} = di{l, 1}(min.locs + 50);
end

end

% Plot mean sound speeds for each ITP
plot_mult_svp(meanSoundSpeed(:, 1:35), di{l, 1}, offset, descr(1:35), ...
    'date.range', goodITPs(1:35), 'Mean SVP first half of ITPs')
for j = 1:35
    if ~isempty(mid.min.z{j})
        plot(meanSoundSpeed(di{l, 1}==mid.min.z{j}) + (j-1)*offset, mid.min.z{j}, ...
            ',s', 'MarkerSize', 14)
    end
end

plot_mult_svp(meanSoundSpeed(:, 36:end), di{l, 1}, offset, descr(36:end), ...
    'date.range', goodITPs(36:end), 'Mean SVP second half of ITPs')
for j = 36:length(goodITPs)
    if ~isempty(mid.min.z{j})
        plot(meanSoundSpeed(di{l, 1}==mid.min.z{j}) + (j-1)*offset - ...
            offset*35, mid.min.z{j}, ',s', 'MarkerSize', 14)
    end
end

% Separate ITPs with Double Ducts
% As a result of looking at plots of mean SVFs
ddITPs = [1 2 3 5 6 18 21 30 33 34 35 41 42 43 53 54 55 62 64 65 68 69 ... 70 77 78 79 80 81 84 85 86 87];
ddIndex = [];
%ddLabel = cell(length(ddITPs));
for i = 1:length(ddITPs)
    ddIndex = [ddIndex find(goodITPs == ddITPs(i))];
ddLabel{i} = num2str(ddITPs(i));
end

plot_multsvp(meanSoundSpeed(:,ddIndex),di{1,1},offset,descip(ddIndex),
'date_range',goodITPs(ddIndex),'Mean SVP from ITPs with Double Duct')

% Plot several profiles from selected ITP
singIndex = find(goodITPs == 2);
numCasts = 30;
singCasts = floor(linspace(1,length(goodCasts{singIndex}),numCasts));
castIndex = [];
for i = 1:numCasts
    singLabel{i} = itpHeader(singIndex,singCasts(i)).monthday;
end

plot_multsvp(SoundSpeed{singIndex}(:,singCasts),di{1,1},...
    offset,singLabel,'date_range',84,'SVPs from ITP 84')

% Plot strong (84) and weak (21) SVPs, with annotation
offset = 0;
stIdx = find(goodITPs == 84);
wkIdx = find(goodITPs == 21);

stwk(:,1) = meanSoundSpeed(:,wkIdx);
stwk(:,2) = meanSoundSpeed(:,stIdx);
mean_SVP = [di{1,1} meanSoundSpeed(:,stIdx)];
[wkPks, wkLocP] = findpeaks(stwk(:,1));
[wkMns, wkLocM] = findpeaks(-stwk(:,1));
wkPks = wkPks(1);
wkLocP = wkLocP(1);
wkMns = -wkMns(1);
wkLocM = wkLocM(1);

diff = abs(stwk(wkLocM:end,1) - wkPks);
wkLow = wkLocM + find(diff==min(diff));

[stPks, stLocP] = findpeaks(stwk(:,2));
[stMns, stLocM] = findpeaks(-stwk(:,2));
stPks = stPks(2);
stLocP = stLocP(2);
stMns = -stMns(3);
stLocM = stLocM(3);

diff = abs(stwk(stLocM:end,2) - stPks);
stLow = stLocM + find(diff==min(diff));

% Plot SVPs, Source Depth
plot_mult_svp(stwk,di{1,1},offset,{'ITP 21', 'ITP 94'},'',84,'')
ax1 = gca;
plot([1400 1500],[130 130],':','LineWidth',2)
text(1460,130,'Source Depth 130 m', 'FontSize',16)

% Plot Weak SVP Upper/Lower Bounds
plot(wkPks, di{1,1}(wkLocP),'v','MarkerFaceColor','red','
'MarkerEdgeColor','red','MarkerSize',6)
text(wkPks, di{1,1}(wkLocP),[num2str(di{1,1}(wkLocP)),' m',
num2str(wkPks),'/m/s'], 'FontSize',15)
plot(wkPks, di{1,1}(wkLow),'v','MarkerFaceColor','red','
'MarkerEdgeColor','red','MarkerSize',6)
text(wkPks, di{1,1}(wkLow),[num2str(di{1,1}(wkLow)),' m',
num2str(wkPks),'/m/s'], 'FontSize',15)
% Plot Strong SVP Upper/Lower Bounds
plot(stPks+offset, di{1,1}(stLocP), 'v', 'MarkerFaceColor', 'red', ...
'MarkerEdgeColor', 'red', 'MarkerSize', 6)
text(stPks+offset, di{1,1}(stLocP), [num2str(di{1,1}(stLocP)), ' m', ...
num2str(stPks), ' m/s'], 'FontSize', 15)
plot(stPks+offset, di{1,1}(stLocP), 'v', 'MarkerFaceColor', 'red', ...
'MarkerEdgeColor', 'red', 'MarkerSize', 6)
text(stPks+offset, di{1,1}(stLocP), [num2str(di{1,1}(stLocP)), ' m', ...
num2str(stPks), ' m/s'], 'FontSize', 15)

% Plot Local Min in both SVPs
plot(wkMns, di{1,1}(wkLocM), '>', 'MarkerFaceColor', 'red', ...
'MarkerEdgeColor', 'red', 'MarkerSize', 6)
text(wkMns, di{1,1}(wkLocM), [num2str(di{1,1}(wkLocM)), ' m, c=', ...
num2str(wkMns), ' m/s'], 'FontSize', 15)
plot(stMns+offset, di{1,1}(stLocM), '>', 'MarkerFaceColor', 'red', ...
'MarkerEdgeColor', 'red', 'MarkerSize', 6)
text(stMns+offset, di{1,1}(stLocM), [num2str(di{1,1}(stLocM)), ' m, c=', ...
num2str(stMns), ' m/s'], 'FontSize', 15)

ax1.XTickMode = 'auto';
ax1.XLim = [1445 1480];

% Plot ITP Locations

for j = 1:length(goodITPs)
allLats = [];
allLons = [];
for c = 1:length(allCasts)
if ~isempty(itpHeader(j,c).lat)
allLats = [allLats itpHeader(j,c).lat];
end
if ~isempty(itpHeader(j,c).long)
allLons = [allLons itpHeader(j,c).long];
end
end
```matlab
d.lat(j) = mean(allLats);
d.lon(j) = mean(allLons);
clear allLats allLons
end

% ALL Itp Map

NewPolarMap
plotm(d.lat, d.lon, 'Marker', '+', 'MarkerEdgeColor', 'red', 'MarkerSize', 6,...
'LineStyle', 'none')
plotm(d.lat(ddIndex), d.lon(ddIndex), 'Marker', 's', 'MarkerFaceColor', 'red',...
'MarkerEdgeColor', 'red', 'MarkerSize', 8, 'LineStyle', 'none')
% Add dates to each location
dlat = -.018;
dlong = .018;
%title('Locations of ITPs showing Double Duct SVP', 'FontSize', 25)

% Just double ducts map

NewBeaufortMap
plotm(d.lat, d.lon, 'Marker', '+', 'MarkerEdgeColor', 'red', 'MarkerSize', 6,...
'LineStyle', 'none')
plotm(d.lat(ddIndex), d.lon(ddIndex), 'Marker', 's', 'MarkerFaceColor', 'red',...
'MarkerEdgeColor', 'red', 'MarkerSize', 8, 'LineStyle', 'none')
textm(d.lat(ddIndex), d.lon(ddIndex), ddLabel, 'FontSize', 15)

% Just ITPs 2 & 84 with time

singIndex = find(goodITPs == 2);
numCasts = length(goodCasts{singIndex});
singCasts = floor(linspace(1, length(goodCasts{singIndex}), numCasts));
for i = 1:numCasts
    singLabel{i} = itpHeader(singIndex, singCasts(i)).monthday;
```

68
singLat(i) = itpHeader(singIndex, singCasts(i)).lat;
singLon(i) = itpHeader(singIndex, singCasts(i)).long;
end

NewBeaufortMap
plotm(singLat, singLon, 'Marker', '.', 'MarkerSize', 6, 'LineWidth', 2)
textm(singLat(1), singLon(1), 'ITP 2', 'FontSize', 15)
textm(singLat(1), singLon(1), ['...

itpHeader(singIndex, singCasts(1)).monthday', ', ','...
itpHeader(singIndex, singCasts(1)).year]

plotm(singLat(1), singLon(1), 'Marker', '^', 'MarkerSize', 8, 'MarkerFaceColor', 'b')
plotm(singLat(end), singLon(end), 'Marker', 'v', 'MarkerSize', 8, 'MarkerFaceColor', 'b')

clear singLabel singLat singLon numCasts

singIndex = find(goodITPs == 84);
numCasts = length(goodCasts{singIndex});
singCasts = floor(linspace(1, length(goodCasts{singIndex}), numCasts));

for i = 1:numCasts
    singLabel{i} = itpHeader(singIndex, singCasts(i)).monthday;
singLat(i) = itpHeader(singIndex, singCasts(i)).lat;
singLon(i) = itpHeader(singIndex, singCasts(i)).long;
end

plotm(singLat, singLon, 'Marker', '.', 'MarkerSize', 6, 'LineWidth', 2)
textm(singLat(1), singLon(1), 'ITP 84', 'FontSize', 15)
textm(singLat(1), singLon(1), ['...
itpHeader(singIndex, singCasts(1)).monthday', ', ',...
clear all
close all

% ITP.calcStrength.m
% Calculate the strength of each ITP, along with the locations bounding the
% lower duct.
% 
% Thomas Howe  thomhowe@mit.edu  (413) 320-2050

load('ITPdata') %where the filename is changed as appropriate

offset = 12;  %Offset value for plotting SVPs

% Loop over ITPs to process sound speed for each.
for j = 1:length(goodITPs)
    % Calculate Mean Sound Speed
    meanSoundSpeed(:,j) = mean(SoundSpeed{j},2);
    descrip{j} = ['ITP ',num2str(goodITPs(j))];

    % Find Local Minimum in SVP
    c_neg = -meanSoundSpeed(50:200,j);
    [~, min_locs] = findpeaks(c_neg,'SortStr','descend','NPeaks',1);
    if ~isempty(min_locs)
mid_min_z{j} = di{1,1}(min_locs+50);
end

end

%% Separate ITPs with Double Ducts
% As a result of looking at plots of mean SVPs
ddITPs = [1 2 3 5 6 18 21 30 33 34 35 41 42 43 53 54 ... 55 62 64 65 68 69 70 77 78 80 81 82 84 85 86 87];
ddIndex = [];
%ddLabel = cell(length(ddITPs));
for i = 1:length(ddITPs)
    ddIndex = [ddIndex find(goodITPs == ddITPs(i))];
    ddLabel{i} = num2str(ddITPs(i));
end

plot_mult_svp(meanSoundSpeed(:,ddIndex),di{1,1},offset,...
    descrip(ddIndex),'date_range',goodITPs(ddIndex),...
    'Mean SVP from ITPs with Double Duct')

%% Plot several profiles from selected ITP
singIndex = find(goodITPs == 2);
numCasts = 30;
singCasts = floor(linspace(1,length(goodCasts{singIndex}),numCasts));
castIndex = [];

for i = 1:numCasts
    singLabel{i} = itpHeader(singIndex,singCasts(i)).monthday;
end

plot_mult_svp(SoundSpeed{singIndex}(:,singCasts),di{1,1}...
    ,offset,singLabel,'date_range',84,'SVPs from ITP 84')
%% Declare Max & Min Location Vectors
itpPks = ones(1,length(ddIndex));
itpMns = itpPks;
FullStrength = itpPks;

% Certain ITPs have local max & min which are clearly not the double duct.
% These values correct for these, and are found through trial and error.

itpPks(30) = 2;
itpMns(30) = 3;
itpMns(28:29) = 2;
itpMns(16) = 2;
itpPks(16) = 2;
itpMns(13) = 2;
itpMns(12) = 3;
itpMns(8) = 10;
itpPks(8) = 7;
itpMns(2) = 2;

for i = 1:length(ddIndex)
    ss = meanSoundSpeed(:,ddIndex(i));
    dd = di{1,1};

    [Pks, LocP] = findpeaks(ss);
    [Mns, LocM] = findpeaks(-ss);
    Pks = Pks(itpPks(i));
    LocP = LocP(itpPks(i));
    Mns = -Mns(itpMns(i));
    LocM = LocM(itpMns(i));
    diff = abs(ss(LocM:end) - Pks);
    Low = LocM + find(diff == min(diff));

    if 0
        figure
hold on
plot(ss, dd)
plot(Pks, dd(LocP), 'v', 'MarkerSize', 6, 'MarkerFaceColor', 'red', ... 
'MarkerEdgeColor', 'red')
plot(Mns, dd(LocM), '>', 'MarkerSize', 6, 'MarkerFaceColor', 'red', ... 
'MarkerEdgeColor', 'red')
plot(Pks, dd(Low), '>', 'MarkerSize', 6, 'MarkerFaceColor', 'red', ... 
'MarkerEdgeColor', 'red')
ax = gca;
ax.YDir = 'reverse';
title(['ITP Number ', num2str(ddITPs(i)), ', Index ', num2str(i)])
end

FullStrength(i) = Pks - Mns;
FullHeight(i) = abs(dd(LocP) - dd(Low));
end

%% Compare to Limited ITP84 Profile
load('ITP84MeanSVP.mat')
c_dd_min = 1451.1;
strength = linspace(0.25, FullStrength(ddITPs == 84), 25);
for j = 1:length(strength)
[LIMmean_SVP, minPos, maxPos] = limitSVP(mean_SVP, c_dd_min, strength(j));
if 0
figure
hold on
plot(mean_SVP(:, 2), dd)
plot(mean_SVP(maxPos(1), 2), dd(maxPos(1)), 'v', 'MarkerSize', 6, ... 
'MarkerFaceColor', 'red', 'MarkerEdgeColor', 'red')
plot(mean_SVP(minPos(1), 2), dd(minPos), '>', 'MarkerSize', 6, ... 
'MarkerFaceColor', 'red', 'MarkerEdgeColor', 'red')
plot(mean_SVP(maxPos(2), 2), dd(maxPos(2)), 'v', 'MarkerSize', 6, ... 
'MarkerFaceColor', 'red', 'MarkerEdgeColor', 'red')
ax = gca;

73
ax.YDir = 'reverse';
title(['ITP Number ',num2str(ddITPs(i))...', Index ',num2str(i)]
end

LimStrength(j) = abs(mean_SVP(maxPos(1),2) - mean_SVP(minPos,2));
LimHeight(j) = abs(dd(maxPos(1)) - dd(maxPos(2)));
end

figure
hold on
plot(FullStrength,FullHeight,'o','MarkerSize',8,...
'MarkerFaceColor','red','MarkerEdgeColor','red')
plot(LimStrength,LimHeight,'b--','LineWidth',2)
xlabel('Strength (m/s)')
ylabel('Height (m)')
legend('ITPs','Simulated Lower Ducts')
ax = gca;
ax.FontSize = 16;
ax.LineWidth = 2;

%% Plot ITP Locations

for j = 1:length(goodITPs)
allLats = [];
allLons = [];
for c = 1:length(allCasts)
if ~isempty(itpHeader(j,c).lat)
allLats = [allLats itpHeader(j,c).lat];
end
if ~isempty(itpHeader(j,c).long)
allLons = [allLons itpHeader(j,c).long];
end
end
d.lat(j) = mean(allLats);
d.lon(j) = mean(allLons);
clear allLats allLons
end

% ALL Itp Map

NewPolarMap

plotm(d.lat, d.lon, 'Marker','+', 'MarkerEdgeColor',...
'red', 'MarkerSize', 6, 'LineStyle', 'none')
plotm(d.lat(ddIndex), d.lon(ddIndex), 'Marker','s',...
'MarkerFaceColor', 'red', 'MarkerEdgeColor', 'red',...
'MarkerSize', 8, 'LineStyle', 'none')

% Add dates to each location
dlats = -.018;
dlongs = .018;
%title('Locations of ITPs showing Double Duct SVP', 'FontSize', 25)

% Just double ducts map

NewBeaufortMap

plotm(d.lat, d.lon, 'Marker','o', 'MarkerEdgeColor',...
'red', 'MarkerSize', 2, 'LineStyle', 'none')
for i = 1:length(ddIndex)
plotm(d.lat(ddIndex(i)), d.lon(ddIndex(i)), 'Marker','o',...
'MarkerFaceColor', 'red', 'MarkerEdgeColor', 'red', 'MarkerSize',...
8*Fullstrength(i)/mean(Fullstrength)+2, 'LineStyle', 'none')
textm(d.lat(ddIndex(i)), d.lon(ddIndex(i)), ddLabel{i}, 'FontSize', 15)
end
Appendix B

Code for using KRAKEN and analyzing results

B.1 Generating Environmental Files

function krakenENVgen(depths, soundspeeds, ...
freq, outFileName, clow, chigh, depth)
%% krakenENVgen.m
%% Generates a .env environment file, for use with KRAKEN, from the
%% acoustics toolbox. All of the options for the .env file can be changed
%% in this function. They are currently selected for use with an average
%% double duct arctic profile, as generated from WHOI's ITP data.
%%
%% The depth and sound speeds should be vectors, shallowest values first.
%% The outFileName should be a string, excluding the .env extention.
%% The freq should be a double, the test frequency in Hz.
%%
%% More information about the .env format can be found at
%% http://oalib.hlsresearch.com/Modes/kraken.pdf
%%
%% Thomas Howe  thomhowe@mit.edu  (413) 320-2050
if length(depths) == length(soundspeeds)
    disp('SSP Lengths Match');
else
    disp('ERROR: SSP Length Missmatch')
end

np = length(depths);
outFileName = [outFileName,'.env'];

% Assign Header Options
freq = freq; % frequency, in Hz
nmedia = 1; % number of layers in profile, excluding halfspaces
options = 'NV'; % options (N- N2 linear interpolation,
% V - vacuum top boundary condition, M- dB/m)
%bunden = 0.0092; % Value from .env example
%eta = 8.2; % Value from .env example
%xi = 5.1; % Value from .env example
sigma = 0; % Value from .env example
Z = floor(depths(end)); % deepest depth
cs = 0; % Value from .env example
rho = 1.03; % Value from .env example
botopt = 'V'; % vacuum bottom boundary
%clow = 1500; % Lower phase speed limit (m/s)
%chig = 1504; % Upper phase speed limit (m/s)
rmax = 200; % Max range (km)
NSD = 1; % Number of source depths
NRD = 750; % Number of receiver depths
SD = depth; % Source depth (m)
RD = '18 700 /'; % Receiver depth (m)

nmesh = ceil(freq*10*Z/clow); % Number of mesh points
disp(['Working on ',num2str(freq),'Hz, using ',num2str(nmesh),...' mesh points'])

%% Open File
fid = fopen(outFileName,'w');

%% Write header options to file
fprintf(fid,['''',outFileName(1:end-4),''' ']);  
fprintf(fid,'%d  ! FREQ (Hz)
', freq);
fprintf(fid,'%d  ! NMEDIA
', nmedia);
fprintf(fid,'%d  ! OPTIONS
', options);
fprintf(fid,'%d  ! BUMDEN (l/m) ETA (m) XI
', bumden,eta,xi);
fprintf(fid,'%d  ! NMESH SIGMA (m) Z
', nmesh,sigma,Z);

%% Write first line of SSP
fprintf(fid,'%.lf %.lf %.lf %.2f /
', floor(depths(1)), soundspeeds(1), cs, rho);

%% Write the bulk of the SSP
for i=[2:5:np np]
    fprintf(fid,'%.1f %.lf /
', floor(depths(i)), soundspeeds(i));
end

%% Write the tail options
fprintf(fid,['''',botopt,''' ']);
fprintf(fid,'%.1f  ! BOTOPT SIGMA (m) 
', sigma);
fprintf(fid,'%.1f %.1f  ! CLOW CHIGH (m/s) 
', clow, chigh);
fprintf(fid,'%.1f  ! RMAX (km) 
', rmax);
fprintf(fid,'%d  ! NSD 
', NSD);
fprintf(fid,'%d  ! SD (m)
', SD);
fprintf(fid,'%d  ! NRD 
', NRD);
fprintf(fid,'%d  ! RD (m)
', RD);
fprintf(fid, [RD, '
']);
fclose(fid);

B.2 Running KRAKEN

clear all
close all
set(0,'DefaultFigureColormap', (flipud(parula)));

%% Multiple Kraken
% Write a .env file and run kraken on a range of profile (several
% frequencies) to generate a transmission loss plot
%
% Thomas Howe thomhowe@mit.edu (413) 320-2050

%% Load Sound Speed
load('ITP21MeanSVP.mat') % 1451.05 to 1451.31
load('ITP84MeanSVP.mat') % 1451.1 to 1455.4
% loads mean.SVP, a matrix with depths in column 1, speeds in column 2.

%% Declare Extent of Double Duct
% For ITP84
c_dd_min = 1451.106608618971;
c_dd_max = 1455.4;
maxLoc = [30 126];

% For ITP21
%c_dd_min = 1451.05;
%c_dd_max = 1451.31;

% These numbers are based on the local min and max of the double duct
% profile.
%%% Set Problem Parameters
freqs = 100:5:1000;
%freqs = 900;
strengthNum = 100; % Number of Strengths to look at
incTL = true; % Set Coherent vs Incoherent TL
simProfile = true;
betaFreq = 900;
betaStrength = 1; %(Strength Index)

%%% Set Desired Plots
plotTLvRF = false;
plotNumModefreq = false;
plotNumModestrength = true;
plotGPvF = false;
plotGvP = false;
calcBeta = true; %must also plotGvP
plotDetectDist = true;

strength = linspace(c.dd.min, c.dd.max-0.05,strengthNum+1)-c.dd.min;
strength = strength(2:end); % get rid of first strength, which is zero
%strength = 2
for j = 1:length(strength)
disp(['Working on Strength ',num2str(j),'] of ',... num2str(length(strength))])
if simProfile
c_dd_max = c_dd_min + strength(j);
end
[LIMmean.SVP, minLoc, maxLoc] = limitSVP(mean.SVP,...
c.dd_min, strength(j));
else
LIMmean.SVP = mean.SVP;
end

% Find Sound Speed at Source Depth
sd = 180; %source depth at 130 meters
diff = abs(LIMmean.SVP(:,1) - sd);
cSD = LIMmean.SVP(diff == min(diff),2);

for i = 1:length(freqs)
% Write Environment & Field Files
filename = ['dd',num2str(freqs(i)),' Uz '];
krakenENVgen(LIMmean.SVP(:,1),LIMmean.SVP(:,2),freqs(i),...
filename,c.dd_min,c.dd_max,130)
krakenFLPgen(filename,incTL)

% Run Kraken
kraken(filename)

if plotTLvRF || plotDetectDist
% Plot Transmission Loss
fig = figure(999);
set(fig,'Visible','off')
plotTLr([filename,'.shd'],sd)

% Get Data from Plot
ax = fig.CurrentAxes;
D = get(ax,'Children');
ranges{j}(i,:) = get(D,'XData');
TL{j}(i,:) = get(D,'YData');
close(fig)
end

% Get Number of Modes from .prt
MODES{j} = readPRTmodes(filename,i,j);
numModes(j,i) = MODES{j}.num;
if numModes(j,i) > 0
for n = 1:numModes(j,i)
    phaseSpeed{i,j,n} = MODES{j}.phaseSPEEDS(n);
groupSpeed{i,j,n} = MODES{j}.groupSPEEDS(n);
end

%% Calculate Launch Angle of Highest Mode
kr = MODES{j}.kr(numModes(j,i));
k = 2*pi*freqs(i)/cSD;
maxAngle{j}(i) = acos(kr/k);
end

%% Plot TL at Receiver Depth vs Freq and Range
if plotTLvRF
    figure(666)
    ax = gca;
    imagesc(ranges{j}(1,:),freqs,TL{j})
xlabel('Range (km)')
ylabel('Frequency (Hz)')
title(['Transmission Loss vs Range and Frequency at Receiver Depth, for strength '...\n    ,num2str(strength(j))])
%legend(num2str(strength(j)));
ax.YDir = 'normal';
ax.FontSize = 16;
ax.LineWidth = 2;
c = colorbar;
c.Label.String = 'Transmission Loss (dB)';
c.Label.FontSize = 16;
end
end %Strength Loop
%% Plot Number of Modes vs Freq
if plotNumModeFreq
if length(freqs) > 1
figure
clear label
hold on
ax = gca;
for j = 1:length(strength)
plot(freqs, numModes(j,:), 'LineWidth', 2)
label{j} = num2str(strength(j));
end
xlabel('Frequency (Hz)')
ylabel('Number of trapped modes')
htLegend = legend(label,'Location','best');
ht = text(...
'Parent', hLegend.DecorationContainer, ...
'String', 'Duct Strength (m/s)', ...
'HorizontalAlignment', 'center',...
'VerticalAlignment', 'bottom',...
'Position', [0.5, 1.05, 0], ...
'Units', 'normalized',...
'FontSize',14);
ax.YDir = 'normal';
ax.FontSize = 16;
ax.LineWidth = 2;
end
end

%% Plot Number of Modes vs Strength
if plotNumModeStrength
if length(strength) > 1
figure
clear label
hold on
ax = gca;
for i = 1:length(freqs)
plot(strength,numModes(:,i),'LineWidth',2)
label{i} = num2str(freqs(i));
end
xlabel('Strength (m/s)')
ylabel('Number of trapped modes')
hLegend = legend(label,'Location','best');
hlt = text(...
'Parent', hLegend.DecorationContainer, ...)
'String', 'Frequency (Hz)', ...
'HorizontalAlignment', 'center', ...
'VerticalAlignment', 'bottom', ...
'Position', [0.5, 1.05, 0], ...
'Units', 'normalized',...
'FontSize',14);
ax.YDir = 'normal';
ax.FontSize = 16;
ax.LineWidth = 2;
end
end

%% Plot Phase & Group Velocity vs Freq
for j = 1:length(strength)
for i = 1:length(freqs)
for m = 1:numModes(j,end)
if ~isempty(phaseSpeed{i,j,m})
phSpeed{j,m}(i) = phaseSpeed{i,j,m};
gpSpeed{j,m}(i) = groupSpeed{i,j,m};
else
phSpeed{j,m}(i) = 0;
gpSpeed{j,m}(i) = 0;
end
end
end
end

85
if plotGPvF
    if length(freqs) > 1
        figure
        hold on
        ax = gca;
        for m = 1:numModes(1,end)
            plot(freqs(length(freqs)-length(find(phSpeed{1,m}))+1:end),...
                 phSpeed{1,m}(find(phSpeed{1,m})),':r','LineWidth',2)
            text(100,1453,['Mode ',num2str (m)],'FontSize',14)
        end
        plot(freqs(length(freqs)-length(find(gpSpeed{1,m}))+1:end),...
             gpSpeed{1,m}(find(gpSpeed{1,m})),'-r','LineWidth',2)
        end
        xlabel('Frequency (Hz)')
        ylabel('Velocity (m/s)')
        legend('Phase Speed','Group Speed')
        ax.FontSize = 16;
        ax.LineWidth = 2;
    end
end

if plotGvP
    i = betaStrength; %strength index
    i = find(freqs == betaFreq);
    if ~isempty(i)
        figure
        hold on
        ax = gca;
        % Find max & min group speeds, to separate upper and lower modes
        % (Only for Full profile)
        if simProfile
            PC = 'b';
        else
            PC = 'r';
        end
end
end
for m = 1:numModes(1,end)
    if phSpeed{j,m}(i) ~= 0 && gpSpeed{j,m}(i) ~= 0
        gs(m) = gpSpeed{j,m}(i);
        ps(m) = phSpeed{j,m}(i);
        text(ps(m),gs(m), ['Mode ',num2str(m)],'FontSize',14)
    end
end
plot(ps,gs,[PC,'o'],[MarkerSize',8,'MarkerFaceColor',PC])

% Fit line to subset of modes
if calcBeta
    if simProfile
        fitModes = 2:5;
    else
        fitModes = [1:3 5:10];
    end
    p = polyfit(ps(fitModes),gs(fitModes),1);
    beta = (-(mean(ps(fitModes))/mean(gs(fitModes)))^2+p(1))^-1;
end
end
end

%%% Calculate and Plot Detection Distance
if plotDetectDist
    nf = 50;  % ambient noise level
    sl = 125;  % source level
    snrT = 1;  % SNR threshold
    tradD = 26;  % detection distance for traditional Arctic profile.
    % Found by running kraken and following same routine on ITP 2
betaFreq = freqs;
betaI = find(freqs == betaFreq);
if ~isempty(betaI)
    TLcut = sl - nf*snrT;
    for i = 1:length(betaI); %select frequency
        for j = 1:length(strength)
            %select
            TLut = sin(maxAngle{j}(i)) - nf*snrT;
            thisTL(j,:) = TL(j)(betaI(i),:); %TL{strength}{freq,range} at rec depth
            thisRange = ranges{j}(betaI(i),:);
            xx = find(thisTL(j,:) >= TLcut);
            if ~isempty(xx)
                detecDist(i,j) = thisRange(xx(1));
            else
                detecDist(i,j) = 100;
            end
        end
    end
if 0
    figure
    hold on
    for i = 1:length(betaI)
        plot(strength, smooth(detecDist(i,:)), 'LineWidth', 2)
        leg{i} = num2str(freqs(i));
    end
    XL = xlim;
    plot(XL, [tradD tradD], 'b:', 'LineWidth', 2)
    text(strength(1), tradD, 'Traditional Arctic Profile', 'FontSize', 14)
    xlabel('Strength (m/s)')
    ylabel('Detection Distance (km)')
    ax = gca;
    ax.FontSize = 16;
    ax.LineWidth = 2;
    hLegend = legend(leg, 'Location', 'best');
    hlt = text(...
'Parent', hLegend.DecorationContainer, ...
'String', 'Frequency (Hz)', ...
'HorizontalAlignment', 'center', ...
'VerticalAlignment', 'bottom', ...
'Position', [0.5, 1.05, 0], ...
'Units', 'normalized', ...
'FontSize', 14);
end

figure
hold on
imagesc(strength,freqs,detecDist)
ax = gca;
ax.FontSize = 16;
ax.LineWidth = 2;
xlabel('Strength (m/s)')
ylabel('Frequency (Hz)')
c = colorbar;
colormap jet
c.Label.String = 'Detection Distance (km)';
c.Label.FontSize = 16;

contour(strength,freqs,detecDist,[tradD tradD],'LineColor','w',...
'LineWidth',2)

if 0
figure
clear leg
hold on
xlabel('Range (m)')
ylabel('Transmission Loss (dB)')
ax = gca;
ax.FontSize = 16;
ax.LineWidth = 2;
for j = 1:length(strength)
plot(thisRange, thisTL(j,:))
leg{j} = num2str(strength(j));
end
plot([0 100], [TLcut TLcut], ':')
legend(leg)
end
end
end

B.3 Creating Simulated Lower Duct

function [limit_SVP, minPos, maxPos] = ...
    limitSVP (mean_SVP, c_dd_min, strength)

% Limits SVP to only the lower duct, with constant sound speed above and
% below the lower duct.

% If the strength desired is higher than the strength of the mean_SVP, the
% limit_SVP will be a constant at c_dd_min + strength

% mean_SVP and limit_SVP are both matrices with depths in the first
% column, sound speeds in the second column.

% Finely Interpolate mean_SVP
%dq = linspace(mean_SVP(1,1),mean_SVP(end,1),1000);
%intp_mean_SVP(:,1) = dq;
%intp_mean_SVP(:,2) = interp1(mean_SVP(:,1),mean_SVP(:,2),dq,'spline');

intp_mean_SVP = mean_SVP;

% Find Local Min & Max, Currently setup for interpolation of ITP84
[Mns, LocM] = findpeaks(-intp_mean_SVP(:,2));
Mns = -Mns(4);
LocM = LocM(4);
[Pks, LocP] = findpeaks(intp_mean_SVP(:,2));
LocP = LocP(3);
Pks = Pks(3);

limit_SVP = intp_mean_SVP;
c_max = c_dd_min + strength;

% Locate Minimum (Redundant when above is uncommented)
MinDV = abs(c_dd_min - intp_mean_SVP(:,2));
minPos = find(MinDV == min(MinDV));

% Find Where c_max is, which positions are closest to minPos
MaxDV = abs(intp_mean_SVP(:,2) - c_max);
 [~,I] = sort(MaxDV);
keyboard
posDV = abs(I(1:4) - minPos);
 [~,J] = sort(posDV);
maxPos(1) = I(J(1));
for i = 2:4
  if length(maxPos)==1
    if abs(I(J(i))-maxPos(1)) > 25
      maxPos(2) = I(J(i));
    end
  end
end
end

limit_SVP(1:min(maxPos),2) = c_max;
limit_SVP(max(maxPos):end,2) = c_max;