Target Tracking Onboard an Autonomous Underwater Vehicle: Determining Optimal Towed Array Heading in an Anisotropic Noise Field

by

Maria Alejandra Parra-Orlandoni

B.S. Systems Engineering
United States Naval Academy, 2005

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
AND
WOODS HOLE OCEANOGRAPHIC INSTITUTION

SEPTEMBER 2007

Author…………………………………………………………………………………………
Joint Program in Oceanography/Applied Ocean Science and Engineering
Massachusetts Institute of Technology and Woods Hole Oceanographic Institution
September, 2007

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

Accepted by……………………………………………………….......................
Henrik Schmidt
Chair, Joint Committee in Applied Ocean Science and Engineering
Massachusetts Institute of Technology and Woods Hole Oceanographic Institution

Accepted by……………………………………………………….......................
Lallit Anand
Chair, Committee on Graduate Students, Department of Mechanical Engineering
Massachusetts Institute of Technology

REPORT DOCUMENTATION PAGE

1. REPORT DATE *(DD-MM-YYYY)* XX-09-2007	2. REPORT TYPE Master's Thesis	3. DATES COVERED *(From - To)* JUN-SEP 2007

4. TITLE AND SUBTITLE Target Tracking Onboard an Autonomous Underwater Vehicle: Determining Optimal Towed Array Heading in an Anisotropic Noise Field	5a. CONTRACT NUMBER N62271-97-G-0026
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Maria Alejandra Parra-Orlandoni	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Massachusetts Institute of Technology	8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, Ca 93943	10. SPONSOR/MONITOR'S ACRONYM(S) NPS
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
1. DISTRIBUTI9ON STATEMENT A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
In order to overcome the challenges that an anisotropic noise field poses for underwater target tracking, we conduct an onboard estimation of the horizontal noise directionality in the real-time processing suite of an autonomous underwater vehicle (AUV) towing a horizontal line array. The estimation of the noise directionality is a precursor to another adaptive behavior: optimizing tracking capability of a towed array by choosing a particular heading that minimizes the detection level in the target's direction. In each distinct simulated anisotropic noise field, the AUV successfully calculates the optimal towed array headings based on the real-time estimation of the horizontal noise directionality. The findings reveal a clear advantage over the conventional broadside beam tracking method, with some limitations due predominantly to the noise field itself.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Sean Tibbitts, Educational Technician
a. REPORT	b. ABSTRACT	c. THIS PAGE	UU	109	19b. TELEPHONE NUMBER *(include area code)* (831) 656-2319 civins@nps.edu

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Target Tracking Onboard an Autonomous Underwater Vehicle: Determining Optimal Towed Array Heading in an Anisotropic Noise Field

by

Maria Alejandra Parra-Orlandoni

Abstract

In order to overcome the challenges that an anisotropic noise field poses for underwater target tracking, we conduct an onboard estimation of the horizontal noise directionality in the real-time processing suite of an autonomous underwater vehicle (AUV) towing a horizontal line array. The estimation of the noise directionality is a precursor to another adaptive behavior: optimizing tracking capability of a towed array by choosing a particular heading that minimizes the detection level in the target's direction. In each distinct simulated anisotropic noise field, the AUV successfully calculates the optimal towed array headings based on the real-time estimation of the horizontal noise directionality. The findings reveal a clear advantage over the conventional broadside beam tracking method, with some limitations due predominantly to the noise field itself.

Thesis Supervisor: Henrik Schmidt

Title: Professor of Mechanical and Ocean Engineering

Acknowledgements

Firstly, I would like to formally thank my advisor, Prof. Henrik Schmidt, without whom I might have never had the opportunity to attend MIT. His sheer enthusiasm for his work is contagious, and it is a constant source of motivation for all who come in contact with him. Prof. Schmidt has provided the kind of guidance and leadership that I have come to hold as exemplary.

I would like to thank Prof. Arthur Baggeroer, who always shows greatest sincerity towards his students. I would also like to thank Geoff Fox and Pete Beaulieu, who unfailingly manage to keep everything organized. Without them, I would have been quite lost on more than one occasion.

A sincere thank you goes to Dr. Mike Benjamin and Dr. Arjuna Balasuriya, who have both offered a helping hand to me many times. I also thank Andrew Poulsen, who generously spent much time explaining concepts and methods to me.

I would like to thank my friends all over the world who have continued to support me. I am thankful to new friends and organizations, including the Kickboxing Club, which made me feel at home in Cambridge.

I would like to thank Prof. Shan-Yuan Ho, known as "Hoho" to me; she is an amazing woman with an inspiring life's story and from whom much can be learned.

I would like to affectionately thank Alexis Dumortier, Kevin Cockrell, Costas Pelekanakis, and the rest of the LAMSS crew. We have had some great laughs in 5-223—just check out the quote book if you have forgotten! I will dearly miss you and hope that our paths cross again in the future.

Finally, a heartfelt thank you to my family: my dad, Ivan J. Parra, who always encourages me to "go for it" and do my best; my brother, Ivan Roberto, who I always hope to inspire and does not know that he himself inspires me; and my mom, Margarita Orlandoni, who consistently gives me the strength to face life and all its challenges. Most lovingly, I thank Mitch Fury, my husband, who is my best friend and allows me to be me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Anti-submarine warfare (ASW) has been a necessary military focus since the introduction of U-boats in the 1940's. As with all military capabilities and strategies, ASW must reflect the advances made in the field in order to remain relevant. Namely, improvements in underwater warfare technology and strategy, in addition to the transition of small unmanned vehicles into the mainstream, call for a more capable defense. In light of such advancements, it is clear that the development of underwater warfare techniques has most certainly outpaced the development of ASW. With terrorist related small-scale attacks in littoral areas being more prevalent in present day, it is important to develop a reliable method with which to defend against such attacks.

In this thesis we attempt to address such concerns by adding improved target tracking capabilities to the behavior library of an autonomous underwater vehicle (AUV) towing a horizontal sonar line array.

1.1 Progression of ASW

It is worthwhile to review the progression of ASW in order to fully appreciate its evolution over the past several decades. Analyzing ASW's progress—or sometimes its lack thereof—reveals the necessity for continued growth and advancement in the field.

1.1.1 Brief history of ASW

The history of American ASW has a rollercoaster-like pattern. Figure 1-1 depicts the cyclic nature of the United States Navy's readiness with respect to ASW from World War II (WWII) until present day [1].
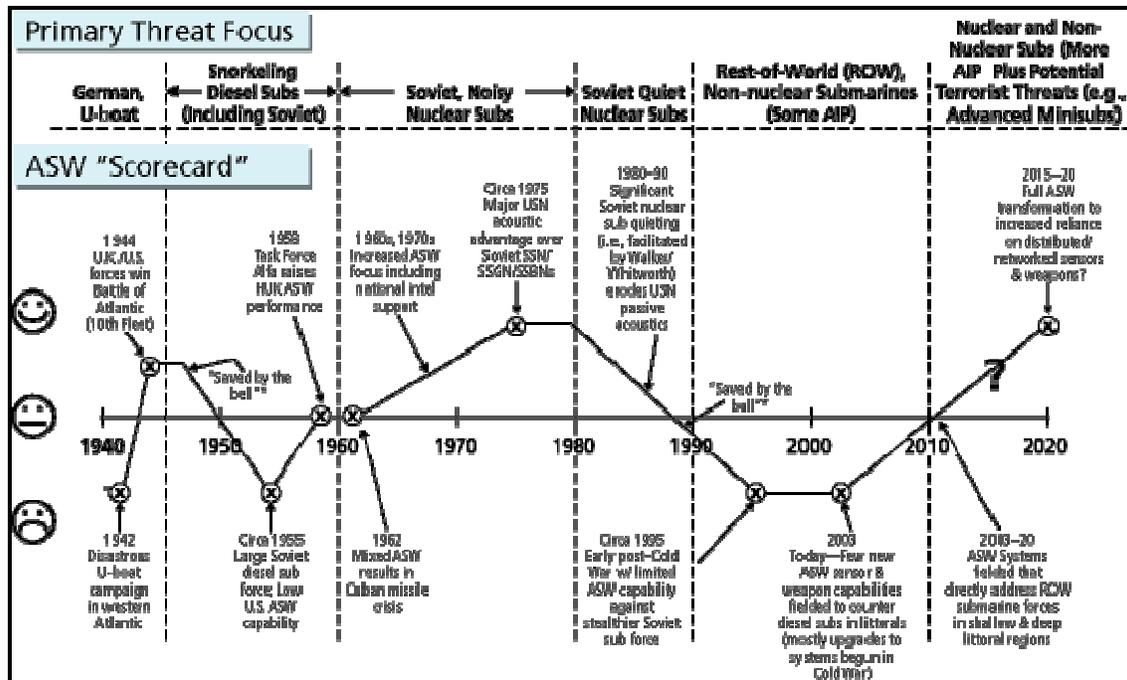
Figure 1-1: Anti-submarine warfare historical performance [1]. This shows the rollercoaster-like successes and failures of American ASW.

When the United States entered WWII in 1941, they made a significant effort in sonar research and development, which greatly improved the performance of active sonar systems, as well as the understanding of underwater acoustic propagation and detection theory [2]. In 1945, however, the Allies faced submarines equipped with snorkels. Radar trials against a snorkel-equipped submarine established "a 0.06 probability of detection [3]," and clearly, such weak defenses were not acceptable. Unexpectedly, the ever-changing arena of underwater warfare brought about a new threat—the Soviet diesel submarine—which took center stage in ASW.

Soon after, a technological revolution occurred at the end of the 1960s: the introduction of digital signal processing. This led to the dramatic increase in capabilities and versatility of sonar systems, particularly as computer performance quickly evolved [2]. Such improvements, in addition to other research and experimentation, contributed to the partial success achieved during the Cuban missile crisis in 1962. The crisis provides one of the best operational examples of the United States Navy combined-force ASW capabilities that emerged from 1950s experimentation and development [3].

In response to the United States Navy's acoustic superiority during the 1960s and 1970s, the Soviets made noteworthy improvements such as a gradual reduction in submarine noise of 35 dB. By the mid-1990s, a significant portion of the (by then Russian) submarines was much quieter than it had been in the 1970s. Luckily, the end of the Cold War approached, and the fleet of quiet Russian submarines lost their imminent threat [3]. Nevertheless, conflicts at the end of the twentieth century with a maritime component such as the attack submarines of the Falklands War in 1982 or the mines of the Gulf War confirmed the importance of mastering sonar techniques to counter threats [2].

Today, more than forty countries excluding the United States collectively have between three and four hundred submarines, including minisubs (under three hundred tons). Nearly 75% of these submarines are relatively modern (post-1970s) designs. Some of the vessels have weapons such as wake-homing antiship torpedoes and submerged-launch antiship cruise missiles, with ranges of over 200 km and speeds up to Mach 3. Such capabilities present a formidable threat to American and allied surface units [3].

On a final unsettling note, minisubs, manned submersibles, and AUVs are becoming common vessels for threatening littoral zones. Some countries known to have such vehicles include Colombia, Iran, North Korea (the largest minisub force in the world), Pakistan, and South Korea. Drug cartels have used manned submersibles and minisubs to smuggle cocaine from ports in Colombia to ships at sea. North Korea has used these types of vehicles to insert agents into the South. The Tamil "Sea Tigers," a terrorist group, has attempted twice to build minisubs. These are merely samples of the ever-present underwater dangers that threaten littoral zones [3].

The next generation of ASW surveillance systems—for the years beyond 2010—has yet to be established, and based on the threats of today, has a demanding mission to fulfill. One promising approach to the problem is the distributed sensor field [3].

1.1.2 A future for ASW: PLUSNet

A current, large scale research project incorporates numerous concepts for a powerful and successful ASW strategy. The project is entitled *Persistent Littoral Undersea Surveillance Network*—or PLUSNet—and encompasses the ideals of a distributed sensor network. PLUSNet's primary objective is to "use a network of AUVs and other mobile and fixed sensors to adaptively and cooperatively detect and track moving underwater targets in a distributed littoral surveillance system (Figure 1-2) [4]." PLUSNet runs under the sponsorship of Tom Curtin of the Office of Naval Research, and principle investigators include Henrik Schmidt of MIT and William Kuperman of Scripps' Marine Physics Laboratory. The program manager is Mitchell Shipley from Penn State, and the director is Marc Stewart.

PLUSNet's "big picture" can be depicted as a grid of multiple semi-autonomous controlled networks consisting of bottom mounted and mobile nodes. The system's environmentally and tactically adaptive processing enhances detection, classification, localization, and tracking of quiet diesel electric submarines operating in shallow water. The network is able to autonomously detect a high level target source, track the source, and forward contact and track information between nodes and to the host ship or shore-based component. Theoretically, enough networks placed in a grid and working collaboratively could provide ASW surveillance capability over an area on the order of 10^4 square kilometers and potentially be sustainable for months or years. In addition, a system made up of AUVs is conceivably less vulnerable to threats such as mines and quiet diesel-electric submarines because the AUVs themselves are small and quiet [5].

Currently, two Bluefin AUVs combine acoustic sensing capability with mobility, facilitating adaptive search behaviors under both fully autonomous and supervisory control. MIT employs Mission Oriented Operating Suite (MOOS) software to accomplish feats such as directing AUVs to track and conduct motion analysis on a potential target, to include applying collaborative behaviors between adjacent mobile sensors. For

example, one vehicle can detect the approach of an acoustic source. After formulating a preliminary track estimate, this first vehicle can communicate its estimate to the second vehicle not in contact with the target. The second vehicle can adaptively converge on, re-acquire, and classify the target. Factors such as avoiding obstacles, AUV energy state, cell sensor coverage requirements, and the need to maintain a functioning communications network constrain all AUV behaviors [5].

Figure 1-2: Overview of Project PLUSNet [4].

1.2   Thesis objective

Detection and tracking theory is always evolving as old problems are solved and new ones arise. In order to deem a truly autonomous network as trustworthy, the AUVs themselves must be able to autonomously and adaptively respond to problematic situations. One of the existing challenges is tracking a target in an anisotropic noise field.

In order to overcome the hindrance to underwater target tracking that an anisotropic noise field introduces, we conduct an onboard estimation of the horizontal noise directionality in the real-time processing suite of an AUV towing a horizontal line array. We apply the methods presented in [6-8] to formulate the autonomous behavior that performs the noise directionality estimation.

This estimation of the noise directionality is a precursor to another adaptive behavior: optimizing tracking capability of a towed array by choosing an array heading that minimizes the detection level in the target's direction. Based on theories previously explored in the Appendix of [9], the AUV behavior utilizes the horizontal noise directionality that has already been estimated in order to determine the optimal towed array heading.

We use a simulated acoustic environment, more thoroughly described in *Appendix A*, in order to explore the performance of these autonomous, adaptive behaviors. We simulate five distinct noise fields so as to assess the success of both behaviors in variable environments. In all of the cases, the AUV successfully calculates the optimal towed array headings based on the real-time estimations of the horizontal component of the three-dimensional noise directionalities. A thorough exploration of the effectiveness of utilizing an optimal towed array heading follows, and the findings strongly support the utility of applying these behaviors to a surveillance fleet of AUVs.

1.3   Thesis organization

This thesis will be presented in the following manner: first, Chapter 2 will describe the noise directionality estimation algorithm presented in [8]. Following, Chapter 3 will expand on the theory introduced in [9] that describes how to determine an optimal towed array heading based on the horizontal noise directionality. Chapters 4 and 5 will provide the software and hardware context in which the theories are applied. This includes equipment specifications as well as the network structure of the various components.

Chapter 6 will follow, presenting simulation results beginning with the noise directionality estimation results from five simulated noise fields. The first simulated noise field is isotropic, in which no directional source is present. The second, called Case A, has a highly directional noise field generated by a narrowband source. The third, called

Case B, has a subtly directional noise field, also generated by a narrowband source. The fourth, called Case C, has a highly directional noise field generated by a narrowband source from a greater distance than in Cases A or B, allowing the waveguide to more significantly affect the noise propagation. The fifth, called Case D, has a highly directional noise field generated by a broadband source. Based on the noise directionality estimation for each of the cases, the optimal towed array headings are determined.

Chapter 7 will evaluate the findings from the simulations and then put the results into the context of live experiments in order to assess the practicality of the algorithms in real-time and in real environments, specifically considering implementation of the autonomous behaviors into the PLUSNet (or any other autonomous distributed sensor network) vehicles. Finally, suggestions for future work will conclude this thesis.

Chapter 2

Estimation of Noise Directionality

2.1 Previous work

The method developed in [6-8], later dubbed the WIT algorithm, provides an effective manner in which to estimate the horizontal noise directionality. First introduced in 1977, [6] differs from its contemporary methods [10-13] because it utilizes the noise directionality term itself for the estimation. The WIT algorithm functions using a sonar array of any geometry, to include a towed horizontal line array, which is often the cheapest and most practical type of array to use. The WIT algorithm in [6] produces a two-dimensional estimate of the pseudo-stationary noise field in the horizontal plane. The earliest version, however, lacks consideration of the vertical arrival structures of noise that are an inherent part of a three-dimensional environment. Thus, [7] further expands the method in 1979 to include vertical arrivals of the noise field and arbitrary array pitch, yielding more precise but still two-dimensional results. Several experiments [9,14-16] test the WIT algorithm and confirm the technique's capabilities.

2.2 Noise directionality estimation: WIT algorithm

The WIT algorithm evolves into a three-dimensional estimate of the noise directionality in [8]. The algorithm employs an iterative technique and utilizes the array's measurements from each of J headings, helping to resolve a conventional hydrophone line array's left-right ambiguity.

Several assumptions about the ambient noise immediately simplify the algorithm calculations. If the total ambient noise is represented as:

$$N(\theta,\phi,t) = n(\theta,\phi) + \zeta(\theta,\phi,t) + \varepsilon(\theta,\phi,t), \tag{1}$$

where θ is azimuth in spherical coordinates, ϕ is elevation in spherical coordinates, and t is time, then $N(\theta,\phi,t)$ is the total ambient noise; $n(\theta,\phi)$ is the pseudo-stationary background noise field directionality; $\zeta(\theta,\phi,t)$ is the time-dependent component of the noise due to the fluctuations in acoustic propagation, noise source movement, changes in noise source levels (i.e., transients); and $\varepsilon(\theta,\phi,t)$ is the error introduced in the measurements by the towing vehicle's noise, array nonlinearities, flow noise, system faults, etc. Estimating $n(\theta,\phi)$ is the ultimate goal.

We assume sufficient temporal averaging so that $\zeta(\theta,\phi,t)$ can be considered negligible. Similarly, we assume sufficient array grooming and appropriate error discrimination processing techniques so that $\varepsilon(\theta,\phi,t)$ can also be considered negligible. $n(\theta,\phi)$ is therefore estimated using the set of beam output noise intensities (with power units), $r_{i,j}$, measured by the ith beam while on the jth array heading with the following equation:

$$r_{i,j} = \frac{1}{T}\int_0^T \frac{1}{2\pi}dt\int_0^{2\pi}\frac{1}{2}d\theta\int_{-\pi/2}^{\pi/2} N(\theta,\phi,t)\times b_i(\theta-\gamma_j,\phi)\cos\phi\, d\phi\, , \tag{2}$$

where T is the measurement time interval, $b_i(\theta-\gamma_j,\phi)$ is the ith beam power response pattern of I beams, and γ_j is the jth array heading of J array headings. With the aforementioned temporal averaging, array grooming, and error discrimination processing techniques, (2) reduces to:

$$r_{i,j} \approx \frac{1}{2\pi} \int_0^{2\pi} 1 \, d\theta \int_{-\pi/2}^{\pi/2} n(\theta, \phi) b_i(\theta - \gamma_j, \phi) \cos\phi \, d\phi. \tag{3}$$

The beam responses are calculated in conical angle space, and then their magnitudes are squared in order to obtain the beam power responses. The beam power responses are approximations of broadband beamforming. In order to approximate the broadband beam response, a narrowband beam response for a specific frequency is calculated at intervals over the given broadband frequency range. Finally, all of the narrowband beam responses are averaged together and normalized, representing a broadband beam response. The expression for each narrowband beam response [17] using the notation in [8] is

$$B(\overline{\beta} : \overline{\beta}_T) = \frac{1}{N} \overline{v}_{\overline{\beta}}^H (\overline{\beta}_T) \overline{v}_{\overline{\beta}}(\overline{\beta}), \tag{4}$$

where $\overline{\beta}$ is the conical angle, $\overline{\beta}_T$ refers to the beams' "look" directions, N is the number of hydrophones in the array, $\overline{v}_{\overline{\beta}}$ is the array manifold vector, and H indicated the Hermetian transform of a vector or matrix. $\overline{v}_{\overline{\beta}}$ breaks down further as follows:

$$\overline{v}_{\overline{\beta}}(\overline{\beta}) = \begin{bmatrix} e^{-j\overline{k}^T \overline{p}_0} \\ e^{-j\overline{k}^T \overline{p}_1} \\ \vdots \\ e^{-j\overline{k}^T \overline{p}_{N-1}} \end{bmatrix}, \tag{5}$$

where $\overline{k}$ is the wavenumber for plane waves propagating in a locally homogeneous medium and is defined as

$$\overline{k} = -\frac{2\pi}{\lambda} \overline{\beta} = -\frac{2\pi}{\lambda} \begin{bmatrix} \sin\theta \cos\phi \\ \sin\theta \sin\varphi \\ \cos\theta \end{bmatrix}. \tag{6}$$

Since a horizontal line array is the sonar instrument utilized in this thesis, only the $\cos\theta$ component of $\bar{\beta}$ is relevant. This component represents the angular width of the conical beam that the array produces. Thus, the wavenumber used in the beamformer is

$$k_x = -\frac{2\pi}{\lambda}\beta_x = -\frac{2\pi}{\lambda}\cos\theta, \tag{7}$$

where λ is c/f; and c is speed of sound in meters per second, and f is frequency in Hertz.

Finally, the $\bar{p}_n$ term in (5) is the position vector for the array's hydrophones. Since the towed array lies on the x-axis, we represent the hydrophone positions, p_{x_n}, in the following way:

$$p_{x_n} = \left(n - \frac{N-1}{2}\right)d, \; n = 0,1,...,N-1. \tag{8}$$

There exists a complication due to the nature of this thesis's applications: the calculated beam patterns are expressed in conical angle. The desired noise directionality estimate, however, requires that the mathematical representation of the towed array and its beam responses be three-dimensional since the anisotropic noise field depends on both the horizontal and vertical. The spherical coordinate system is ideal for such three-dimensional fields. Unfortunately, spherical coordinates are unable to uniquely represent the measured beam noise power, and thus, a discrepancy exists between the most practical coordinate system to utilize throughout the algorithm.

Luckily, a relationship between the spherical and the conical coordinate systems can be expressed as follows (Figure 2-1) [7]:

$$\beta = \cos^{-1}\left\{\begin{array}{l}\cos\theta\cos\phi\cos\alpha + \sin\left[\cos^{-1}(\cos\theta\cos\phi)\right]\\ \times\sin\alpha\sin\left[\tan^{-1}(\sin\phi/\cos\phi\sin\theta)\right]\end{array}\right\}, \tag{9}$$

making the transition between the conical coordinate system when calculating the beam patterns and the spherical coordinate system when estimating the noise directionality

feasible. The geometric relationship defined in (9) transforms the beam responses into a spherical coordinate representation for direct use in (3).

Since the transformation from conical to spherical coordinates is computationally intensive, and thus time consuming, it is both possible and useful to calculate the three-dimensional beam patterns *a priori* and save them for later use. As long as the array specifications and the beamforming method do not change, then the stored beam patterns in spherical coordinates are always applicable.
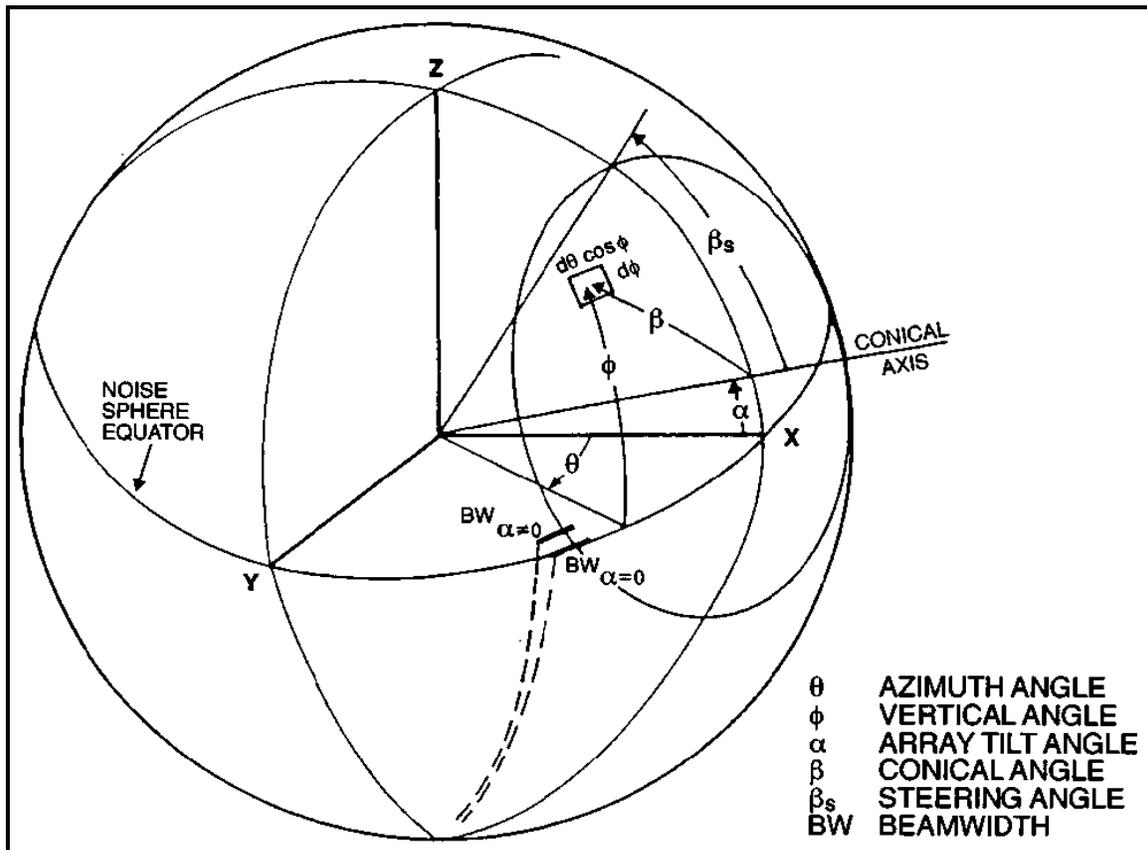
Figure 2-1: Geometric relationship between spherical and conical coordinates [8]. The derivation of (9) is based on this geometric relationship.

In addition to the beam response patterns, the set of beam output noise intensities, $r_{i,j}$, is necessary for running the WIT algorithm. In the case of this thesis, the values for $r_{i,j}$ are obtained by accessing the array's published beam-time response (BTR) files recorded over time for every beam on each of six headings that form a hexagon. Beam output intensity values are only considered when the array is approximately straight in order to preserve computational power as well as maintain accuracy in the final results. Outputs are then converted to decibels using

$$R_{i,j} \ \mathrm{dB} = 10\log(r_{i,j}). \tag{10}$$

The result is an I x J matrix that must be stored for use in the iterative algorithm, where I is the total number of beams and J is the total number of headings.

To initiate the noise directionality estimation, we assume an arbitrary three-dimensional noise field, the most straightforward and logical choice being an isotropic noise field ($\hat{n}_0(\theta,\phi) = C$). This noise field is plugged into the following equation:

$$\hat{r}_{i,j,k} \approx \frac{1}{2\pi} \int_0^{2\pi} 1 \, d\theta \int_{-\pi/2}^{\pi/2} \hat{n}_k(\theta,\phi) b_i(\theta - \gamma_j, \phi) \cos\phi \, d\phi, \tag{11}$$

where $\hat{r}_{i,j,k}$ is the set of estimated beam output intensities. $\hat{r}_{i,j,k}$ is transformed into decibels in the same manner shown in (10). The estimated intensities are compared to the measured intensities using (12), generating an error matrix. The intensities are subtracted in decibels to avoid diversion in the direction of negative intensities.

$$\Delta_{i,j,k} = R_{i,j} - \hat{R}_{i,j,k} \tag{12}$$

The error is then immediately converted back to power units so that all following computations have physical meaning. Each $\Delta_{i,j,k}$ is divided by two for successive underrelaxation and then mapped to its corresponding $b_i(\theta - \gamma_j, \phi)$. This accomplishes correct error distribution over all cells in the θ and ϕ domain.

A guarantee that the iterative algorithm will converge requires normalization, so we apply the conservation of energy to the error distributions. The distributed value of $\Delta_{i,j,k}$ integrated over its corresponding differential conical angle should equal the distributed value of $\Delta_{i,j,k}$ integrated over its differential azimuth and elevation angles, as depicted in (13):

$$\int_0^\pi \Delta_{i,j,k}\, b_i(\beta_j) d\beta = \frac{1}{4\pi} \int_0^{2\pi} 1 d\theta \int_{-\pi/2}^{\pi/2} \Delta_{i,j,k} \cdot b_i(\theta - \gamma_j, \phi) \cdot \cos(\phi) d\phi. \tag{13}$$

The next step is to modify $\hat{n}_k(\theta,\phi)$ according to the calculated errors in order to obtain $\hat{n}_{k+1}(\theta,\phi)$. This new estimate for the noise directionality is then plugged into (11), and the procedure is repeated until the $\Delta_{i,j,k}$ are less than the preset error threshold.

Chapter 3

Determining Optimal Towed Array Heading

One of the obstacles that complicates detecting and tracking underwater targets is anisotropic ambient noise. Classically, the broadside beam is preferred for tracking targets because of its superior angular resolution and high directivity (D), referred to as directivity index (DI) when converted to decibels [17]. This principle most certainly holds true in isotropic noise fields, but variation of ambient noise levels with azimuth introduces the possibility of better beam choices for tracking targets. An experiment presented in [9] explores this idea and shows promising results: the findings demonstrate that oftentimes in a horizontally directional noise field, sonar beams besides the default broadside beam provide superior tracking.

In order to determine the best beam for tracking a target in a given location within a specific noise field, we attempt to minimize the detection level (DL) in the direction of the target. We therefore choose the array orientation that allows the greatest probability of detection with respect to minimum detection level (MDL). In terms of the likelihood of detection, a target located at a position so that its source level is equal to or greater than MDL has greater than a 50% probability of being detected.

The mathematical representation of MDL is expressed in (14), where TL is transmission loss, DT is detection threshold, and NF is the noise field as shown in Equation (15). NL is the omni noise level, and φ is the ship heading.

$$MDL(r,\theta) = TL + DT - DI + NF \tag{14}$$

$$NF = 10\log\left\{\frac{180\left[N_B(\theta+\varphi) + N_B(\theta-\varphi)\right]}{\sin\varphi}\right\} \tag{15}$$

$$N_B = 10^{NL/10} \tag{16}$$

Of the components that comprise MDL, DI is one that can vary in a controlled way; that is to say, in an anisotropic noise field, the DI will change depending on the array orientation. Since minimizing MDL requires maximizing DI, the ultimate goal becomes to find the heading that corresponds to the maximum DI in a particular look direction.

In a known horizontal noise directionality, DI must be computed for all possible towed array headings. This yields numerous DIs as functions of azimuth, and in turn, produces a range of DIs for each look direction that depends on the towed array's heading. We compute DI using (17) [17], where φ_l is the array heading.

$$DI = \left(\frac{1}{4\pi}\int_{-\pi/2}^{\pi/2}\int_0^{2\pi} n(\theta,\phi)b_i(\theta-\varphi_l,\phi)d\theta d\phi\right)^{-1}, \ l = 0,1,2...359 \text{ degrees} \tag{17}$$

DI essentially represents the maximum intensity (power per unit solid angle) divided by average intensity (averaged over the three dimensional noise field). The numerator is the power due to a signal arriving from a target that lies in a particular beam's look direction. In other words, the numerator is the beam power response in the target's direction; and since the beam power response is normalized, it equals one. The denominator represents the noise power at the array output due to anisotropic noise.

Once the range of DIs have been calculated for a particular look direction, then we identify the maximum DI, along with its corresponding heading. This heading becomes the "optimal" one because it creates the MDL for that particular scenario, thus increasing the probability of successful target tracking. Care must be taken to avoid endfire, however, because of its inferior angular resolution.

Chapter 4

Equipment Specifications

4.1 The autonomous underwater vehicle

The current AUV model used in PLUSNet experiments is the Bluefin-21 AUV. Figure 4-1 shows a schematic of the AUV, and Figure 4-2 shows the AUV as it is recovered from the water [18]. The Bluefin-21 is one of the smallest deep-water survey AUVs. It is operationally capable of performing successfully at a 200 m depth, but it can go much deeper [5]. The Bluefin-21 AUV is beneficial for use in PLUSNet experiments for several reasons: it is easy to handle and can quickly be turned around on deck so underwater time is maximized. Additionally, support vessels can carry and operate many Bluefin-21 AUVs simultaneously in order to test multi-AUV operations [18].

The Bluefin-21 AUV is approximately 4.6 m long, 0.53 m in diameter, and displaces approximately 340 kg. Part of its payload includes a conductivity, temperature, and depth sensor, a micro-modem for underwater acoustic communications, and a fixed combined RF (Freewave LAN) and GPS antenna mast. Scripps Marine Physical Laboratory designed a quiet tail-cone for the vehicle with a low-noise, direct-drive propulsor tuned for low-speed operation, improving low frequency acoustic setting [5]. The Bluefin-21 also has a unique, pressure tolerant lithium battery system which eliminates the need to open and reseal pressurized vessels [18]. This battery, which has an advertised 30 hour endurance, drives the ducted propeller, which pushes the vehicle

29

to a 3 knot cruising and 5 knot top speed [5]. Figures 4-3 and 4-4 [19] show the AUV's performance while conducting a 60 degree turn.
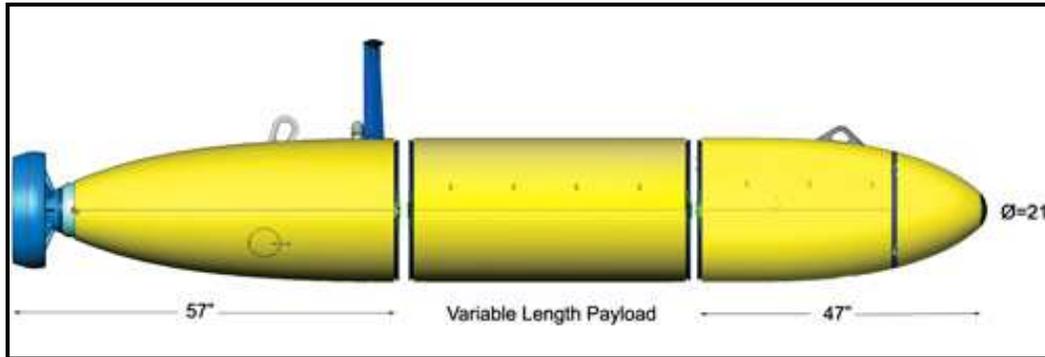
Figure 1-1: Schematic of Bluefin-21 AUV [18]. The total length of the vehicle when used for PLUSNet missions is approximately 4.6 m long (180 in.), making the payload about 1.9 m (76 in.) long. The AUV displaces about 340 kg.
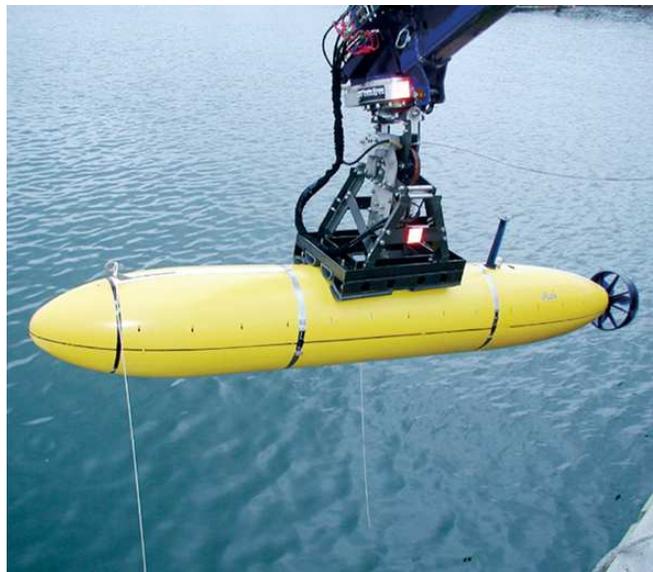
Figure 4-2: Bluefin-21 AUV being recovered from the water [18].
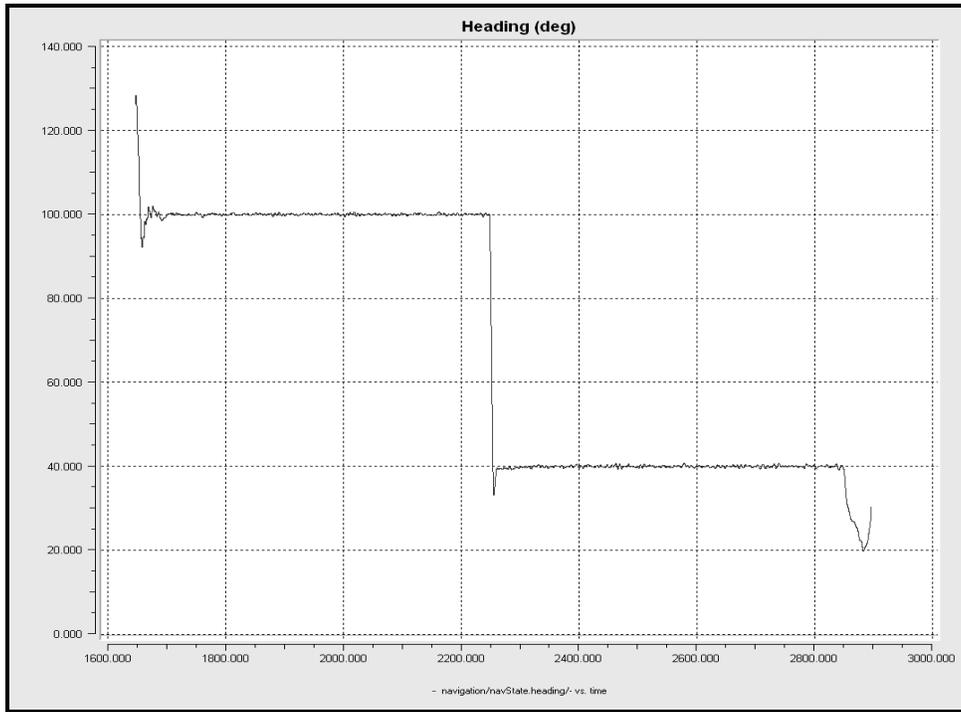
Figure 4-3: AUV heading during 60 degree turn [19].
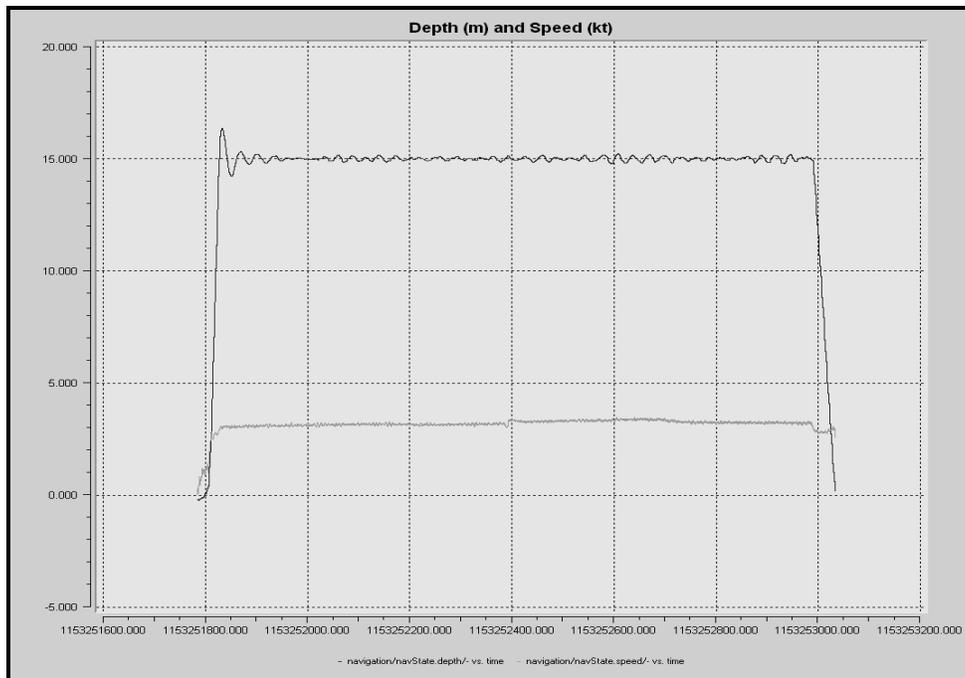
Figure 4-4: AUV depth and speed during 60 degree turn [19]. The top line is depth, and the bottom line is speed.

4.2 The sonar array

One of the sonar array most readily used in current experiments and modeled in the simulator is the Defense University Research Instrumentation Program (DURIP) sonar array [20]. The DURIP sonar array is a conventional hydrophone line array with a depth rating of 300 m. It has a total of 32 hydrophones with 30 m of acoustic aperture. Eleven of the hydrophones are spaced 0.75 m apart and nested in with the remaining 21 hydrophones, which are spaced at 1.5 m. The hydrophones have an acoustic bandwidth of 100 to 1200 Hz and a sensitivity of -176 dB/V/µPa. They have a sensitivity tolerance of +/- 1 dB, and the system noise floor is less than sea state zero. Figure 4-5 summarizes the array's specifications.

The DURIP array has compasses both at the forward and aft end of the array. Additionally, a pressure sensor is mounted just aft of the acoustic portion of the array. While the hydrophone channels are simultaneously sampled to 16 bit resolution at approximately 3 kHz, the heading and pressure are sampled at lower rates and merged with digital hydrophone samples. All of the data is then formatted into a serial data stream and sent via wire to the AUV.

Inside the AUV's payload are a Telemetry Interface Unit and a Data Logging Computer. The Telemetry Interface Unit formats serial-digital array data and sends them to a PC-104 data logging computer, which records the data to a hard disk. The hard disk has a capacity of approximately 72 hours of continuous recording. Simultaneously, data are broadcast via network to an onboard processing computer. The AUV regulates power to the array and data logging computer (24-32 Vdc).
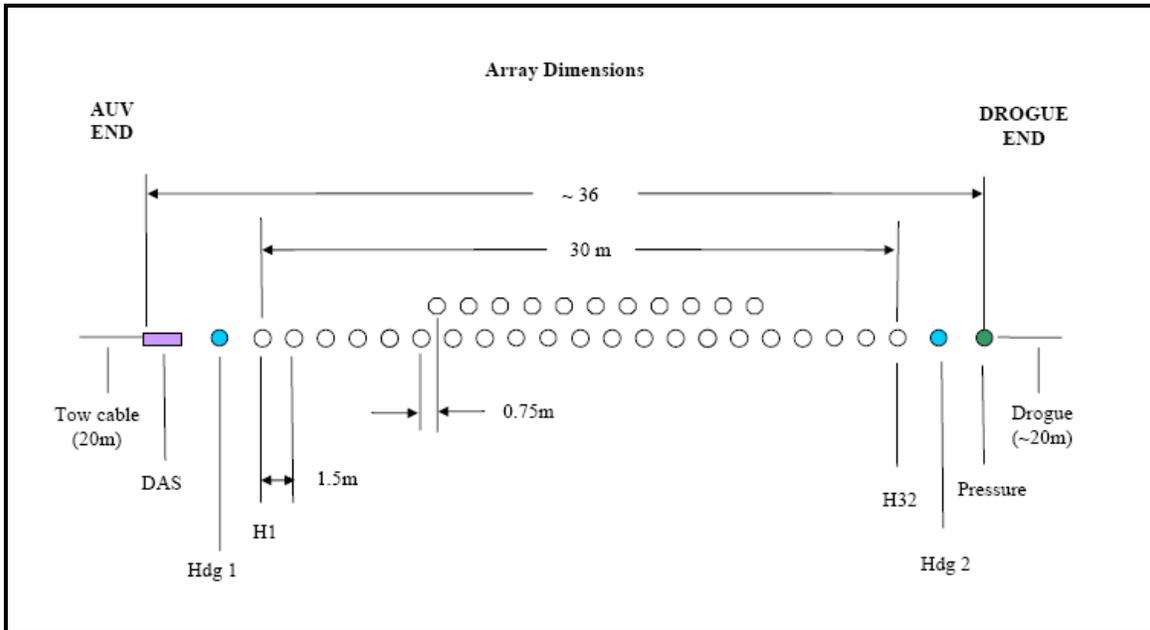
Figure 4-5: DURIP array schematic [20]. Note that the acoustic portion of the array is 30 m long. Also, note the fore- and aft-end compasses (designated as "Hdg 1" and "Hdg 2," respectively), as well as the pressure sensor on the aft end (designated as "Pressure").

Chapter 5

Vehicle Network Structure

Reliable communications between all involved nodes is vital to successful operations with an autonomous surveillance network. Operational communications with or among AUVs may occur on the surface, underwater or a combination of both. Depending on the location and status of the AUVs, there are several communications systems employed during operations. Communication between the host vessel and a surface node, which may include an AUV on the surface, is achieved via IP over radio. When underwater, communication between nodes is achieved by utilizing acoustic modems. Figure 5-1 [21] shows a schematic of the different communication methods.

In addition to reliable communication systems, a sound structure through which all participants in the autonomous network can access data and make decisions as a cohesive unit is necessary. For this, MOOS software is utilized to its fullest advantage.

5.1   MOOS-IvP architecture and Helm-IvP

MOOS is an open source software project for coordinating software processes running on an autonomous platform, typically under GNU/Linux [22]. More specifically, MOOS is an umbrella term for a set of libraries and applications designed to facilitate research in the mobile robotic domain. In Project PLUSNet, MOOS software provides an architecture that allows AUVs to detect, track, and classify potential targets as a

cooperative network. These missions require the application of collaborative behaviors between adjacent mobile sensors. Factors such as avoiding obstacles, AUV energy state, cell sensor coverage requirements, and the need to maintain a functioning communications network constrain the behaviors [5].

As levels of autonomy escalate, the problem of effectively controlling underwater vehicles becomes increasingly difficult. Actions such as obstacle avoidance, adherence of the Rules of the Road, and cooperative moving are simultaneous objectives for vehicle behavior; and with so many objectives, it is difficult without the human mind to decide which objective takes priority. The response to this complex, behavior based system is a method for representing and solving multi-objective optimization problems suitable for controlling vehicles. This method, called Interval Programming (IvP), has been in development throughout the past few decades [23-26]. Figure 5-2 [27] compares the conventional control methods with behavior based control methods.
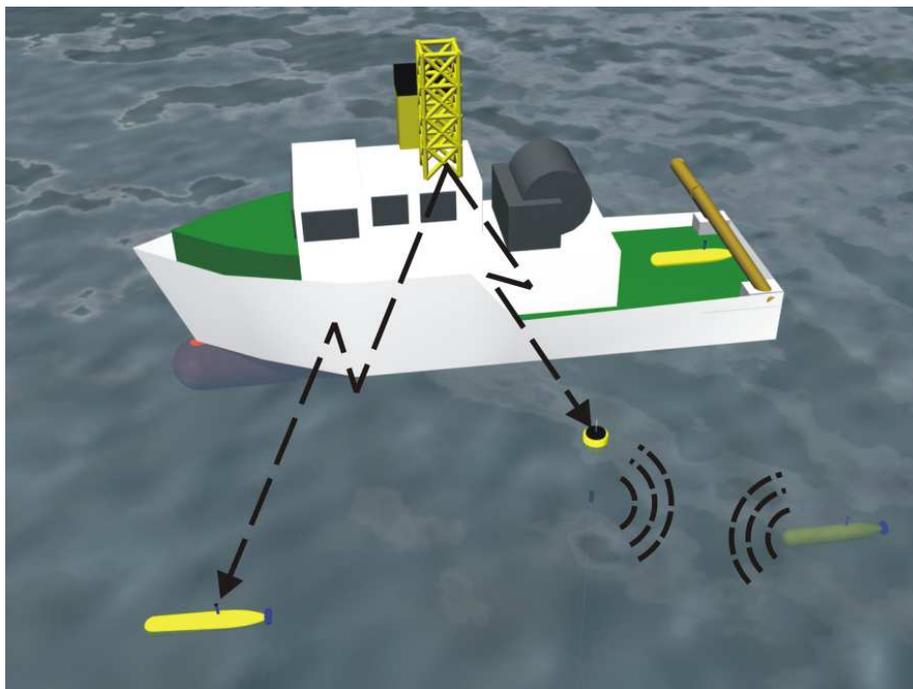
Figure 5-1: Schematic of network communications [21]. Communication from the host vessel to any surface node is achieved via IP over radio. This figure shows a surface node to be a buoy or an AUV on the surface. Underwater communications are achieved via acoustic modem.

The AUVs in PLUSNet use the MOOS-IvP architecture—which consists of MOOS and the IvP Helm—for autonomous marine vehicle control. IvP Helm is a behavior based helm that runs as a single MOOS process and uses multi-objective optimization with the IvP model for behavior coordination [27-29]. A MOOS community contains processes that communicate through a database process called the MOOSDB, as shown in Figure 5-3 [30]. MOOS directs processes to execute their iterations at a specified frequency; and thus, it handles new mail on each iteration in a publish-and-subscribe manner. The IvP Helm runs as the MOOS process pHelmIvP as seen in Figure 5-4 [29]. Each iteration of the helm contains the following steps [27]: (1) mail is read from the MOOSDB, (2) mail information is parsed and stored in a local buffer to be available to the behaviors, (3) the conditions required for behavior activity are evaluated for each behavior, (4) active behaviors produce an objective function if applicable, (5) the objective functions are resolved to produce an action which is (6) published to the MOOSDB for other MOOS processes handling lower-level vehicle control to consume.

5.2   Autonomous vehicle behaviors

Significant properties of autonomous vehicle behaviors are that the behaviors have state and are coordinated through multi-objective optimization. Thus, behaviors may influence each other between iterations. They generate and base their output on plans; and they can also run in sequences, in effect executing a plan [24-25,31].
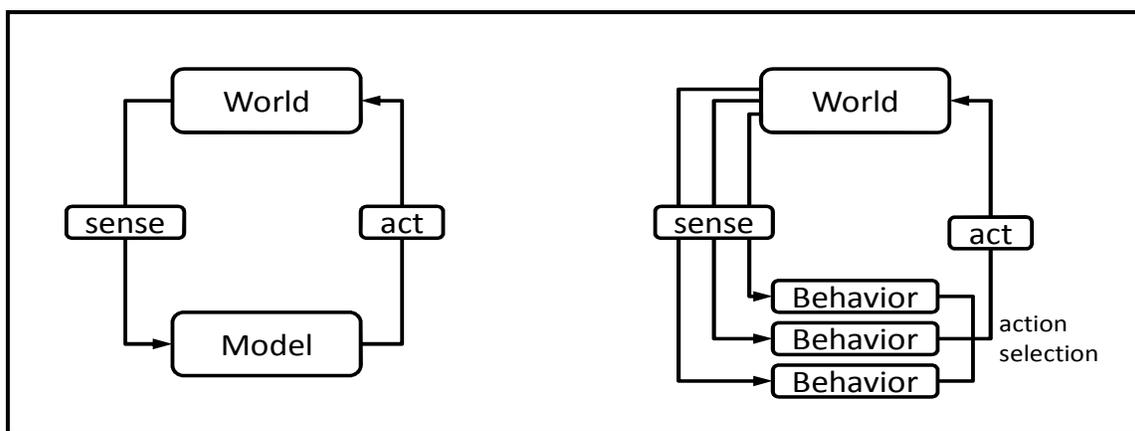
Figure 5-2: Conventional *versus* behavior based control [27].

In this thesis, there is one specific behavior on which the execution of the WIT algorithm—or "plan"—depends: it is the Loiter behavior. Figures 5-5 and 5-6 [31] show specifications detailing the behavior. The Loiter behavior directs the AUV to orbit a fixed point. Given the central coordinates, the Loiter behavior enables the AUV to dynamically determine a list of waypoints to form the orbit. For PLUSNet, the number of waypoints in the Loiter behavior is six, but this number can potentially be changed. Other parameters include travelling clockwise or counterclockwise and vehicle speed. Once in the loiter pattern, the AUV is subject to decisions based on the Waypoint behavior [30].
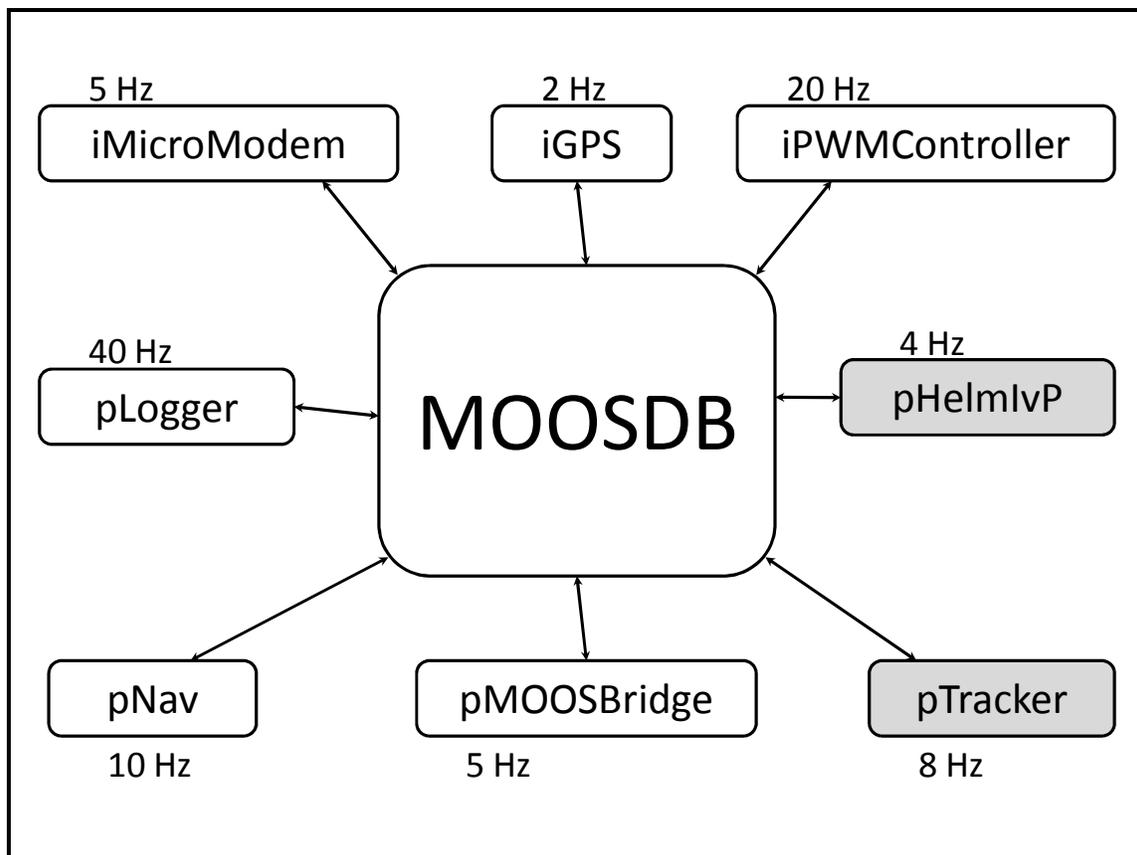
Figure 5-3: MOOS community architecture [30]. The IvP Helm runs as a process called pHelmIvP in a MOOS community. MOOS may be composed of processes for data logging (pLogger), data fusion (pNav), actuation (iPWMController), sensing (iGPS), communication (pMOOSBridge, iMicroModem), and much more. All processes can run at different frequencies.

The Waypoint behavior is responsible for moving the sensor platform—the AUV in this case—from one point to another along the shortest path. The behavior is configured with a list of waypoints and produces objective functions that favorably rank actions with smaller detour distances along the shortest path to the next waypoint. The target vehicle uses this behavior during operation to form a constant velocity motion, for example, and multiple waypoints can be sequenced together to form platform motion along arbitrary polygons, such as the hexagon in PLUSNet experiments. The objective function for the Waypoint behavior is three-dimensional over course, speed, and time [30].

The behavior that this thesis will be attempting to amend is called the ArrayAngle behavior. The ArrayAngle behavior is responsible for holding a vehicle course such that
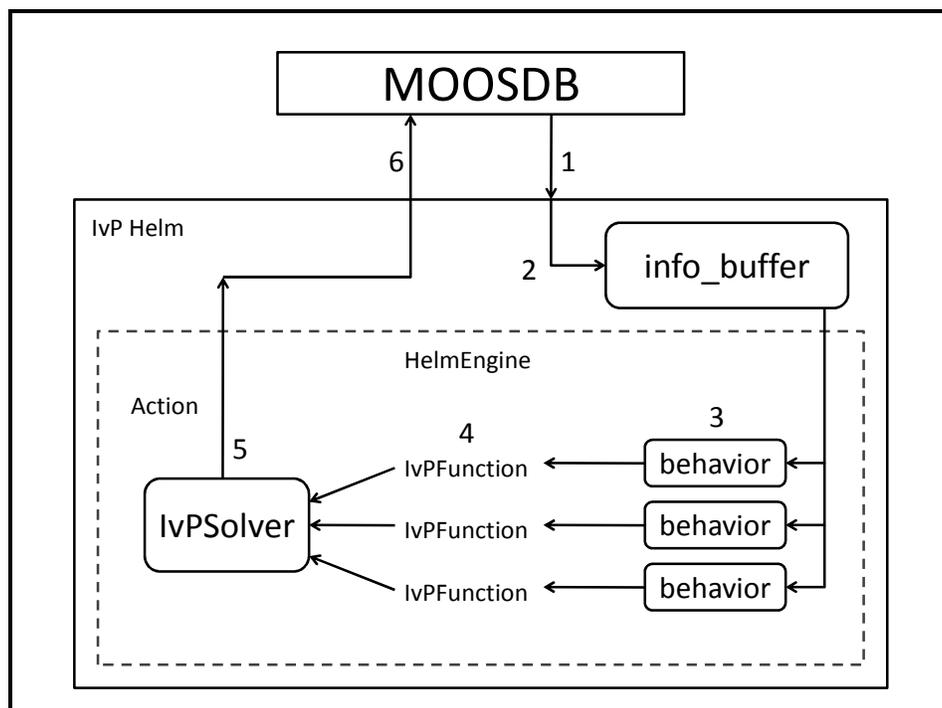
Figure 5-4: Components of the IvP Helm and its iterations flow [27]. (1) Mail is read from the MOOSDB, (2) mail information is parsed and stored in a local buffer to be available to the behaviors, (3) the conditions required for behavior activity are evaluated for each behavior, (4) active behaviors produce an objective function if applicable, (5) the objective functions are resolved to produce an action which is (6) published to the MOOSDB for other MOOS processes handling lower-level vehicle control to consume.

sensor platforms with acoustic line arrays will have the array as close as possible to broadside with the target given the other constraints on vehicle motion. The objective function for this behavior is one-dimensional over course and bimodal, with the modes centered around the two possible course choices that keep the array oriented at broadside with respect to the target. The mode that is centered at the course closest to the vehicle's current course is weighted in order to prevent frequent oscillation between the two modes [30].

Based on the results of this thesis, we will consider modifications to the current ArrayAngle behavior. Namely, the new behavior will be responsible for holding a vehicle course such that sensor platforms with acoustic line arrays will have the target in the array's optimal beam based on MDL and maximum DI given the other constraints on vehicle motion.
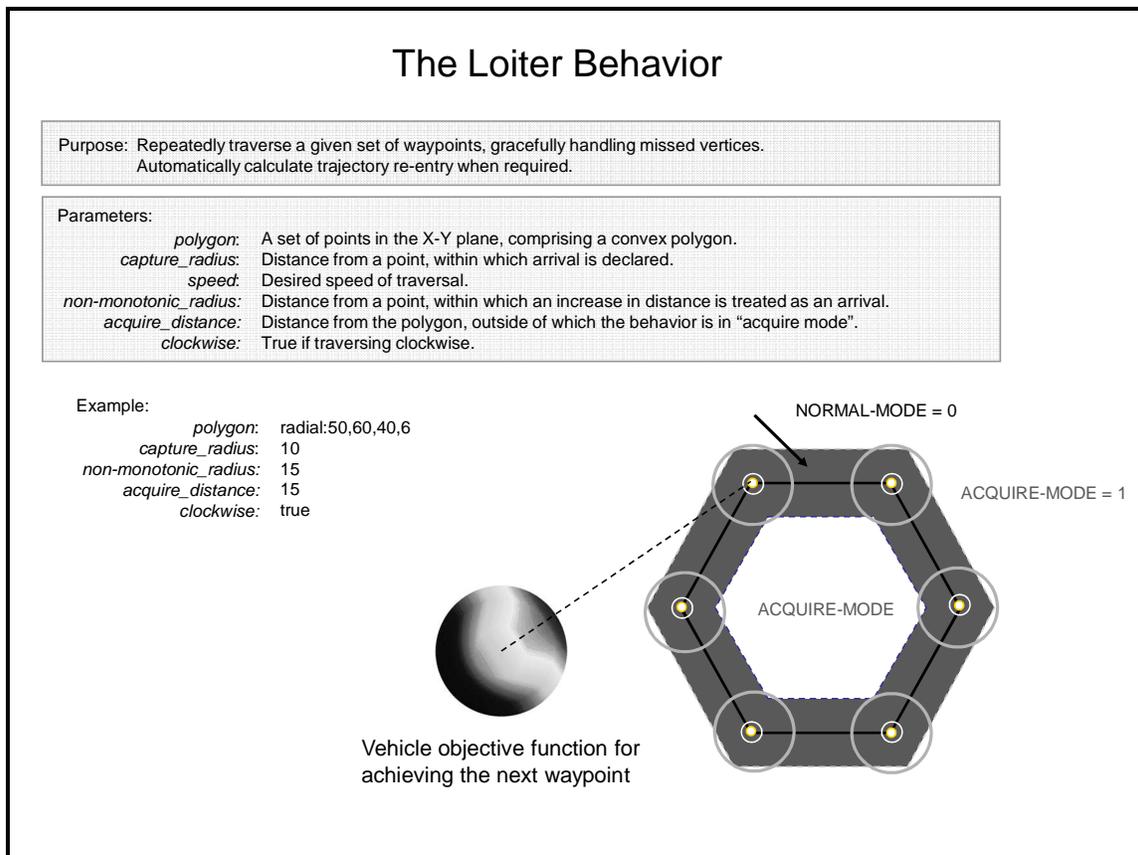
Figure 5-5: Loiter behavior specifications [31].

The Loiter Behavior

(Acquire Vertex Policy - External Case)

Purpose: Repeatedly traverse a given set of waypoints, gracefully handling missed vertices.
Automatically calculate trajectory re-entry when required.

Parameters:
- *polygon*: A set of points in the X-Y plane, comprising a convex polygon.
- *capture_radius*: Distance from a point, within which arrival is declared.
- *speed*: Desired speed of traversal.
- *non-monotonic_radius*: Distance from a point, within which an increase in distance is treated as an arrival.
- *acquire_distance*: Distance from the polygon, outside of which the behavior is in "acquire mode".
- *clockwise*: True if traversing clockwise.

Acquire Vertex Policy (External Case):

acquire_vertex = vi

where $i = \text{argmin}(\theta_i)$

v_i is viewable from p

200 meters
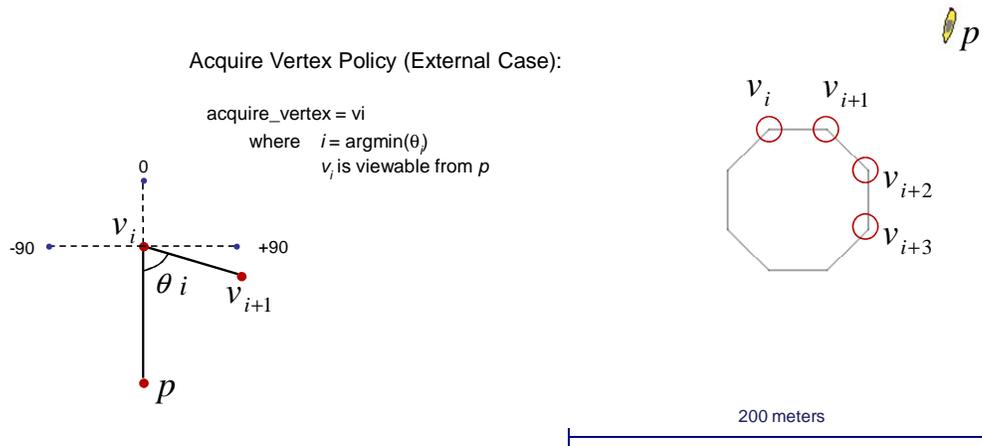
Figure 5-6: Loiter behavior specifications: Acquire Vertex Policy [31].

Chapter 6

Simulation Results

Before reviewing the results, it is important to note that the beam outputs in the simulator are not normalized, so the analyses in this thesis are based on relative, not absolute, values. Quantitative analyses of absolute data values are not of primary concern for the purposes of this thesis; but the qualitative evaluation of one case relative to another is. As an aside, normalization of the BTR data requires consideration of the duration of each data set, the total bandwidth of the array, the number of elements in the array being utilized, etc.

6.1   Horizontal noise directionality estimation

Estimating the noise directionality in a simulated environment before doing so in real-time during a live experiment is essential for assessing the effectiveness and accuracy of the WIT algorithm. We run the algorithm several times, each time with a different simulated ambient noise field in order to evaluate the algorithm's performance in various scenarios. The simulator utilizes data from the live Monterey Bay 2006 PLUSNet experiment. *Appendix A* provides more in depth information addressing the specifics of the simulation environment.

The first step necessary to run the WIT algorithm is to calculate the three-dimensional beam power responses. For this thesis, the beam responses are calculated

with a one degree resolution using a conventional beamformer for twenty-seven beams linearly spaced in cosine space. In order to approximate the desired broadband beam response, we average many narrowband beam responses together; and the frequency bin size for each narrowband beam response is 1 Hz. The true bin size can be calculated by dividing the MOOS variables def_samplerate and def_fftlength, which yields 4000/4096 = 0.9766 Hz; thus validating 1 Hz as a sufficiently close approximation of the true frequency bin size.

One concern that arises from digitally calculating the beampatterns is their inherently discrete nature. We consider smoothing the beam responses in order to minimize the discretization: adding more data points using interpolation or any other comparable method achieves resolution beyond one degree; but in doing so, computation times of ensuing equations greatly increase, along with the memory required to store the more finely resolved beampatterns. In light of the consequences, it is best to maintain the one degree resolution as a compromise between minimizing the discrete nature and maximizing the effectiveness of the beam responses as used in the WIT algorithm.

Another matter to consider when calculating the beam responses is the underwater sound speed value used in the calculations, namely, in (7). The actual speed of sound is variable with range and depth, and can vary significantly throughout different geographical locations. In this thesis, we assume an average sound speed of 1480 m/s, which is an isothermal approximation of the actual Monterey Bay sound speed profile. Such an approximation is sufficient for assessing the WIT algorithm's success.

Figure 6-1 displays four of the twenty-seven conventional beam power responses in cosine space. Figure 6-2 displays the same beams in a polar representation; and it is in this plot that the variable widths of the main lobes are most obviously visible. The endfire beams clearly have the widest main lobes, while the broadside beam has the narrowest. The width of the endfire beams make them undesirable for use in detecting and tracking a target since they introduces so much uncertainty with respect to the target's azimuthal direction.
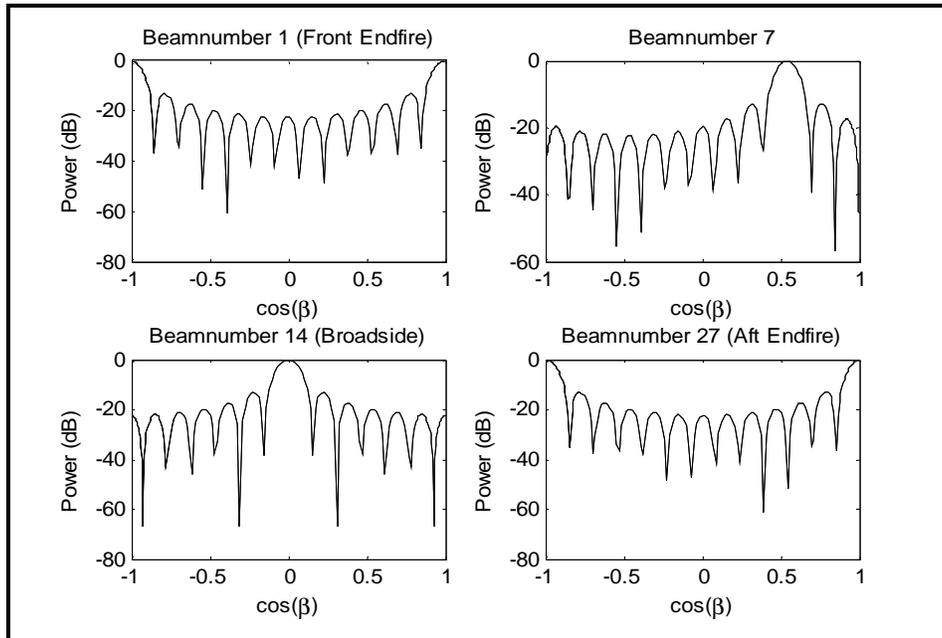
Figure 6-1: Beam power response in cosine space obtained using a conventional beamforming algorithm.
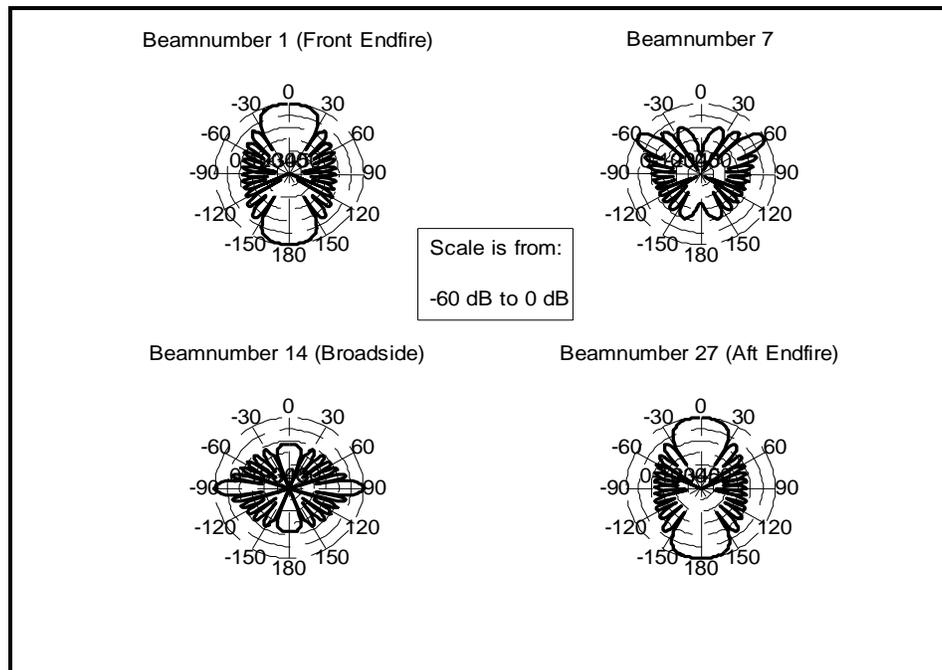
Figure 6-22: Beam power response in polar representation. Note the wide endfire beams' main lobes as compared the broadside beam's narrow main lobe.

Figure 6-3 displays conical angles mapped to their respective locations in spherical coordinates. The geometry depicted in Figure 2-1 and defined in (9) governs the relationship between the two coordinate systems. We obtain the three-dimensional beam power responses by mapping the two-dimensional beam power responses according to Figure 6-3. Note the symmetry in Figure 6-3, which is evidence of the conical shape of sonar beams. Figures 6-4, 6-5, and 6-6 display three-dimensional beam responses for the forward endfire, broadside, and aft endfire, respectively. The mainlobes and sidelobes of the beampatterns are visible in this figure, as is the discrete nature of the stored beam responses.
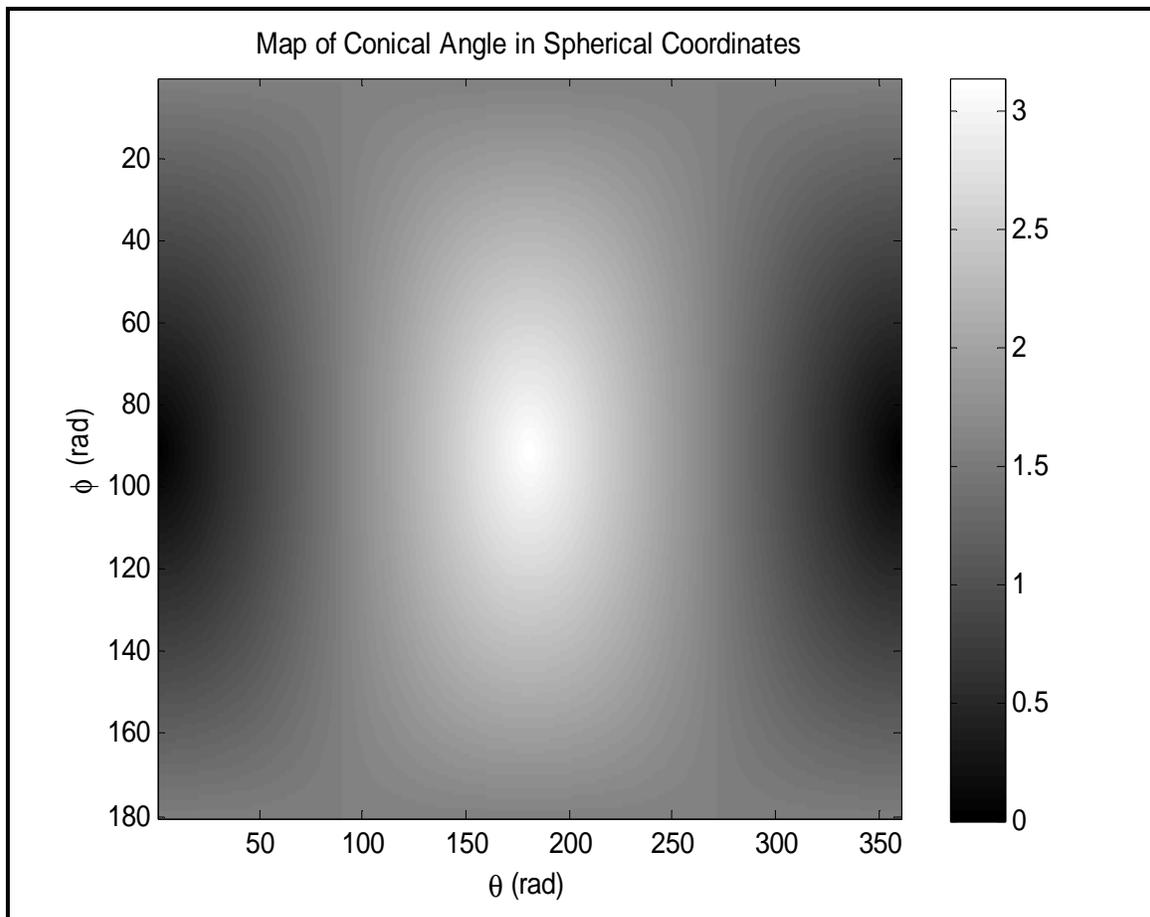
Figure 6-3: Conical angles mapped to spherical coordinates. Note the symmetry, which is evidence of the conical shape of sonar beams.
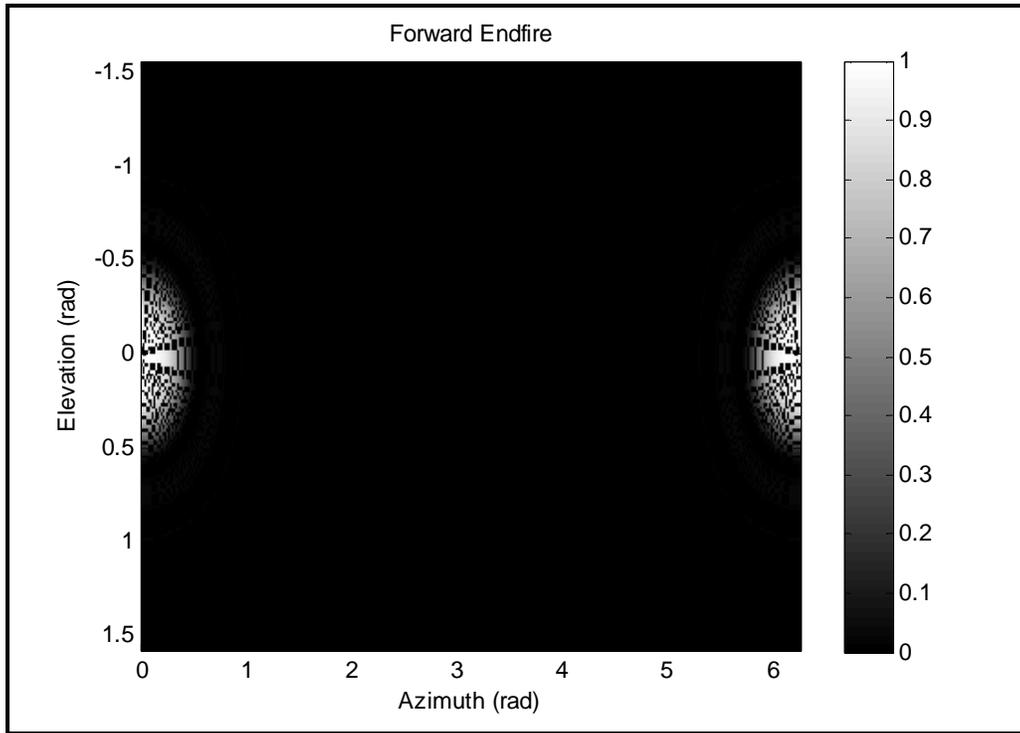
Figure 6-4: Three-dimensional beam power response: Forward Endfire.
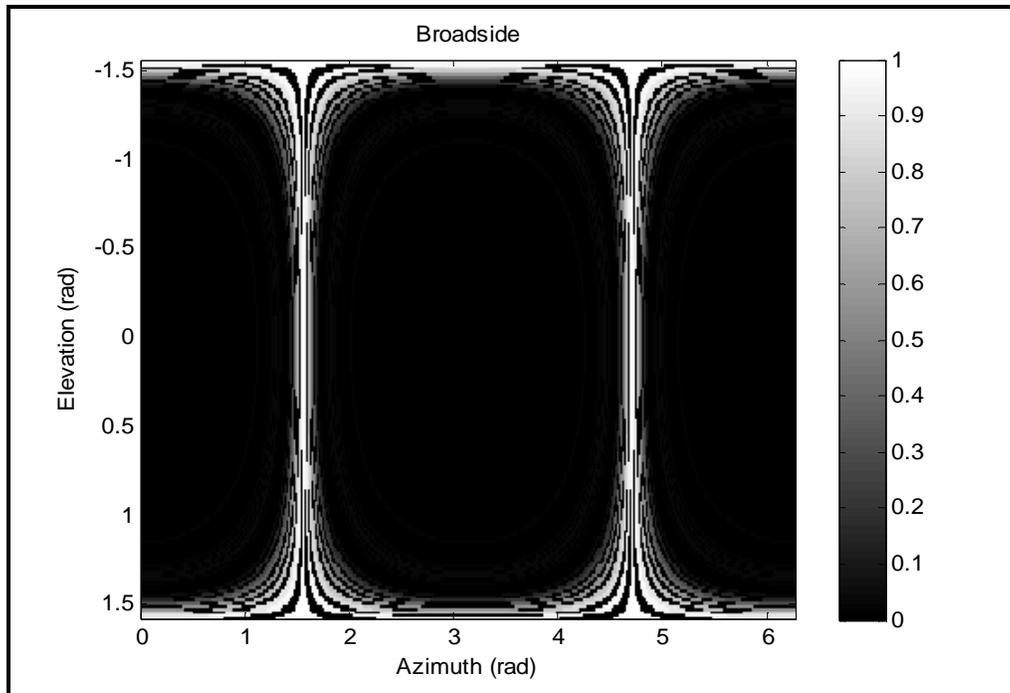
Figure 6-5: Three-dimensional beam power response: Broadside.
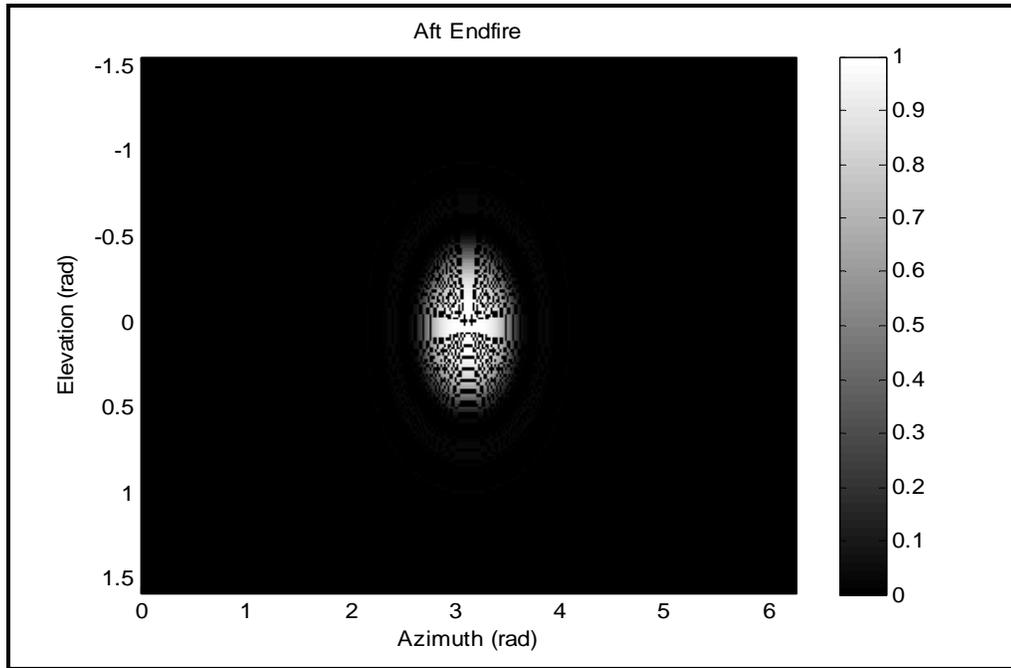
Figure 6-6: Three-dimensional beam power response: Aft Endfire.

The transformation of the beam responses from conical angle to spherical coordinates is computationally intensive. Thus, the three-dimensional beam responses must be calculated *a priori* and stored in a library for later use, since calculating them in real-time is not feasible with the current available computational power. See *Appendix B.1* to review the MATLAB code that computes the three-dimensional beams.

After storing the three-dimensional beam power responses, the WIT algorithm can commence its noise directionality estimation. First, The MATLAB code in *Appendix B.2* records the necessary data and pre-processes them for use in estimating the horizontal noise directionality. Then the code in *Appendix B.3* actually runs the WIT algorithm, which—on an important note—can run in real-time, as the calculations involved in estimating the noise directionality do not have limiting computational demands. Further reductions in computation time also occur after translating the MATLAB code in *Appendices B.2, B.3, and B.4* to a lower level code such as C++.

Although the beam outputs are recorded in their entirety, the only data actually applied towards estimating the horizontal noise directionality are the beam outputs that occur while the sonar array is straight enough. Restricting the acceptable curvature of

the array helps preserve the integrity of the beam output data used in the WIT algorithm by eliminating the data that are too distorted to be reliable. In this thesis's simulations, the array is considered "straight enough" when the hydrophones' y-positions (left-right from the tow point) have a standard deviation less than 1 m and the z-positions (up-down from the tow point) have a standard deviation less than 0.5 m. With a 30 m long acoustic aperture, 1 m is 3.33% and 0.5 m is 1.67% of the array length. In simulation, the array rarely approaches these curvature limits, indicating that the data are reliable. In actuality, however, the allowable curvature limit for an array is less than $\lambda/10$ to ensure uncompromised beamforming accuracy. For the sake of testing the robustness of the WIT algorithm in future live experiments, the curvature limits set in these simulations are more lenient.

Another detail to consider when defining the WIT algorithm parameters is the error threshold on which the accuracy of the final noise directionality estimate depends. Too large of a threshold grants excessive latitude in the possible azimuthal distribution of noise energy, while too small of a threshold might be cause for a divergent result. The error threshold for these simulations is 0.075, which is in the unnormalized power units. Since the original beam intensity outputs used to calculate the error are on the order of 10^{12} and greater, the chosen error threshold is sufficiently stringent. It is important to remember that the error is originally calculated in decibels, as shown in (12), to avoid diversion in the direction of negative intensities. This quantity is then transformed back into power units and divided by two, and it becomes the value that determines whether the algorithm will continue on to perform another iteration, or terminate with a final noise directionality estimate. The error threshold amount may be somewhat arbitrary, as long as it is affords a practically accurate result in the final noise directionality estimate.

Figure 6-7 shows the beam intensity outputs that the sonar measures in the simulated isotropic noise field; the results are corrected for the sonar array's headings during the measurements. The beam outputs are as theoretically expected, with uniform noise levels despite array orientation. Figure 6-8 shows the resulting horizontal noise directionality estimate after applying the WIT algorithm to the beam intensity outputs shown in Figure 6-7. Though there are small variations in the noise levels due to the discrete nature of all the data, the horizontal noise directionality field is clearly isotropic.

For direct comparison with Figure 6-7, Figure 6-9 shows the beam intensity outputs that the sonar measures in the simulated noise field in Case A. Again, the results are corrected for the sonar array's headings during the measurements. In Case A, a stationary, almost narrowband, source with center frequency 800 Hz and +/-10 Hz
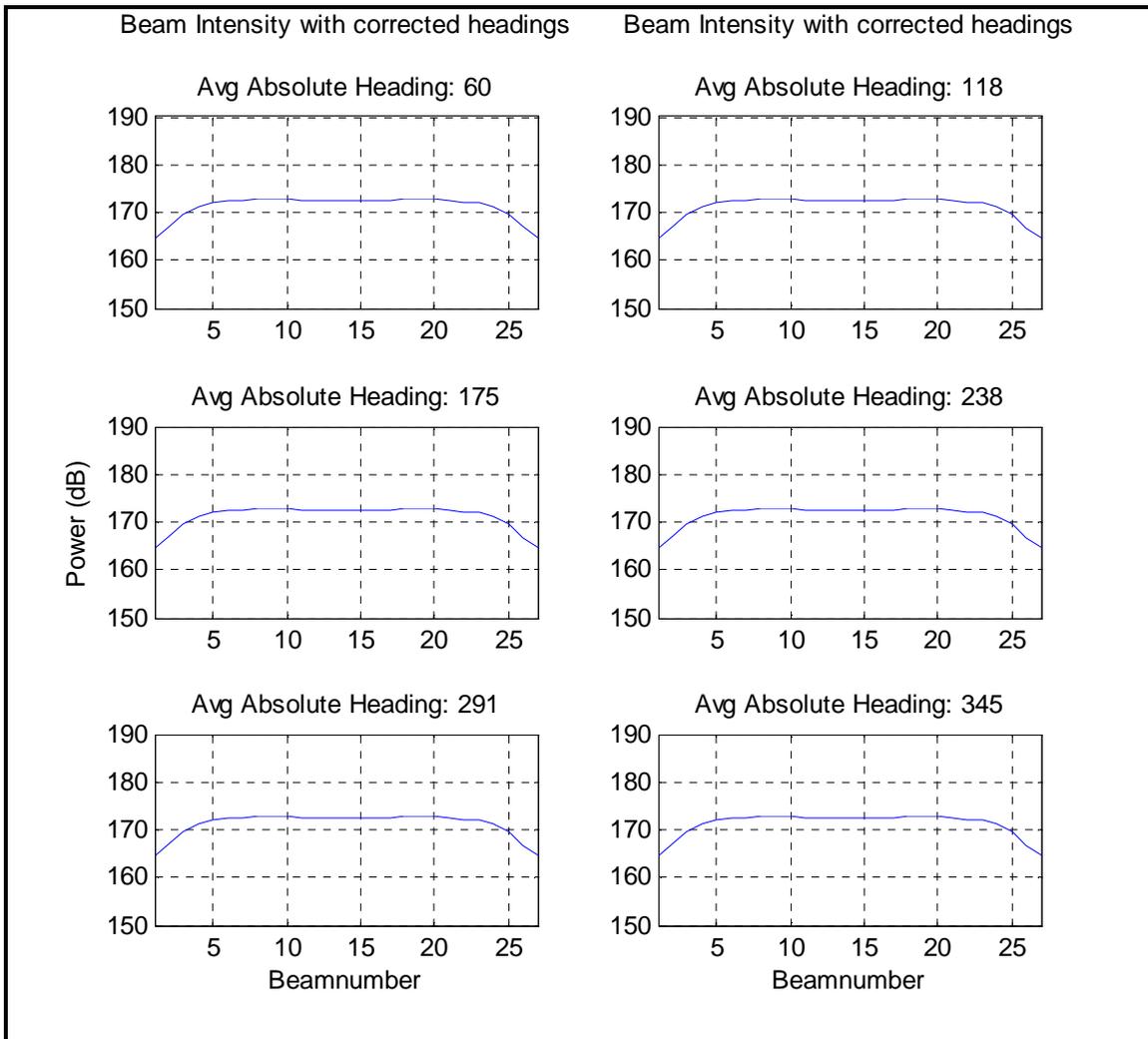
Figure 6-7: Beam intensity outputs in absolute heading: Isotropic Case. Note the uniform responses, despite array orientation.
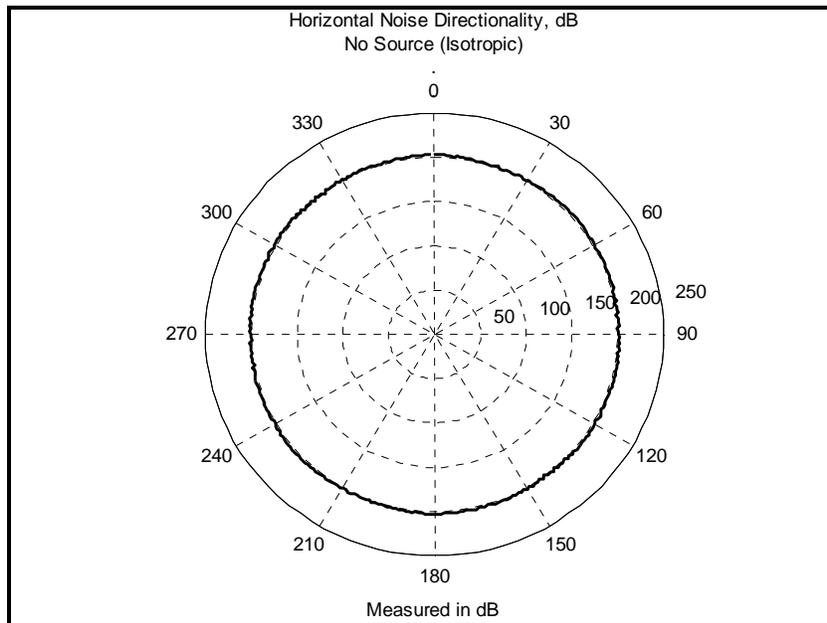
Figure 6-8: Estimated horizontal noise directionality: Isotropic Case.

bandwidth is located 1000 m due south of the AUV's central loitering coordinates, generating a powerful noise pressure level to simulate a highly directional noise field. Unlike Figure 6-7's uniform beam response, Figure 6-9 displays some obvious peaks that are caused by the source in Case A. The waveguide disperses the energy radiating from the source; otherwise, we would expect a sharp peak in the direction of the source, and this would be a highly unrealistic result [32]. The levels of the beam intensity peaks in Figure 6-9 are between 15 dB and 20 dB more than the noise floor. After accounting for spreading, absorption, and other underwater physics, it is fitting then that the resulting horizontal noise directionality estimation should have maximum noise levels that exceed the noise floor by about 20 dB to 25 dB.

Figure 6-10 shows the resulting horizontal noise directionality estimation for Case A after running the WIT algorithm. To better observe the effects of the source on the noise field, Figure 6-11 shows the same noise directionality rose with a reduced dynamic range, obtained by subtracting the noise floor from the horizontal noise directionality. In this "zeroed" figure, it is quite easy to see the sound pressure that the source generates. The figure also readily shows the effects of the waveguide on the directional source: though the source is located directly south of the loiter point, the maximum in the noise

49

field does not occur exactly at 180 degrees, but closely on either side of this point. There is noise spreading throughout the majority of the southern azimuths of the noise field. Such attributes are in accordance with the waveguide theory [32], thus suggesting success and a trustworthy degree of accuracy in the performance of the WIT algorithm.

We run a second instance of the WIT algorithm in a simulated noise field called Case B. A source with a center frequency of 800 Hz, a +/-10 Hz bandwidth and located at the same depth as the AUV forms Case B's directional noise field. Although
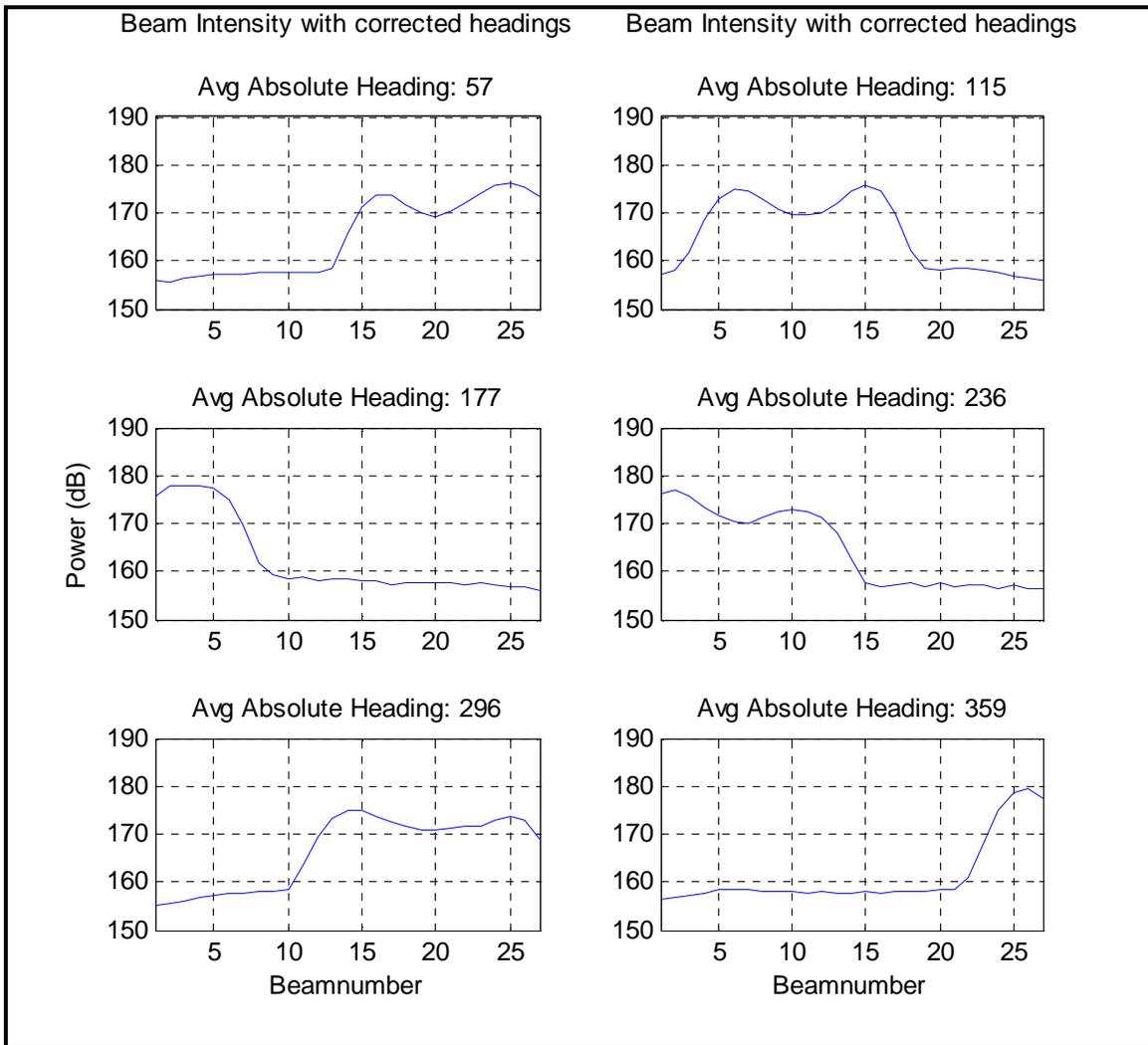
Figure 6-9: Beam intensity output in absolute heading: Case A. The peaks are due to the directional noise field created for Case A.

almost identical to the source in Case A, the source in Case B has a significantly lower pressure level; and therefore, the resulting horizontal noise directionality estimation has lower maximum noise levels with respect to the noise floor than in Case A. Figure 6-12 shows the horizontal noise directionality estimate for Case B, and Figure 6-13 shows the "zeroed" noise field with a reduced dynamic range. In Figure 6-13, it is evident that maxima in the noise field in Case B exceed the noise floor by only about 8 dB to 10 dB. Figure 6-13 also shows similar waveguide effects on the source's noise energy as in Case A, further suggesting the reliability and accuracy of the WIT algorithm.

A narrowband source as in Case A, but with a slightly lower pressure level, creates the simulated noise field in Case C. Additionally, the source is located 2000 m south of the loiter point instead of 1000 m. The increased distance enhances the effects of spreading, attenuation, and other waveguide physics on the source's noise energy. Figure 6-14 shows the horizontal noise directionality estimate of Case C, and Figure 6-15 shows the "zeroed" noise field. The results for Case C demonstrate that with a source at a greater distance, the noise field becomes more narrowly directional, with less noise leakage to the northern azimuths. There is, however, greater loss in sound pressure levels, which is expected due to the greater travel distance.

A broadband source creates the final simulated noise field, called Case D. The source in Case D is located 1000 m due south of the loiter point at the same depth as the AUV with a noise level equal to the source in Case C; and it had a center frequency of 890 Hz with a +/-50 Hz bandwidth. Figure 6-16 shows the horizontal noise directionality estimate for Case D, and Figure 6-17 shows the same estimate with a reduced dynamic range. The broadband source generates a noise directionality with a wider noise energy distribution than with a narrowband source. Additionally, the noise levels are relatively higher: even though the source in Case D has a lower pressure level than in Case A, noise field's maxima are nearly the same as or greater than those in Case A. Along the same lines, the broadband source also generates more leakage into the northern azimuths and visibly undergoes more spreading.

Horizontal Noise Directionality, dB
Source: 1000 m south from central loitering coordinates

Measured in dB
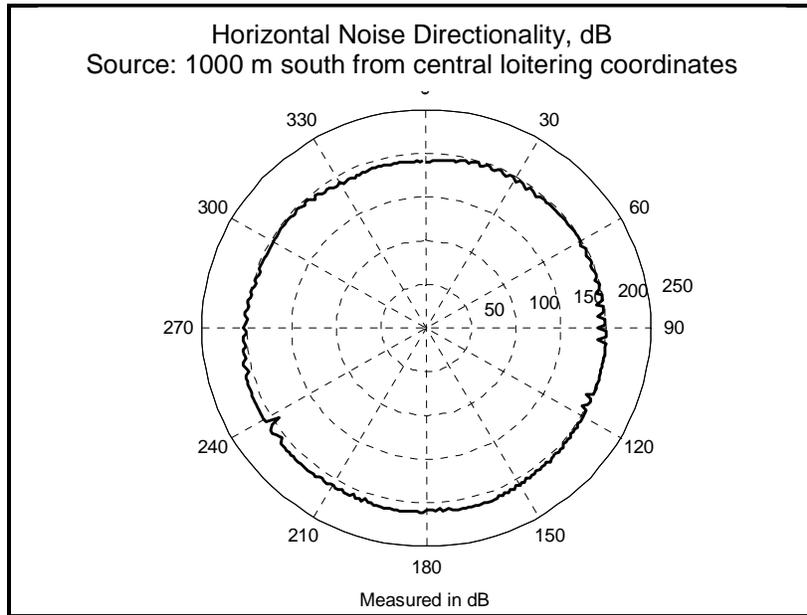
Figure 6-10: Estimated horizontal noise directionality: Case A. Note the higher noise levels in the southern azimuths due to the narrowband source located 1000 m due south of the AUV's central loitering coordinates.

Horizontal Noise Directionality, dB
Reduced dynamic range

Measured in dB
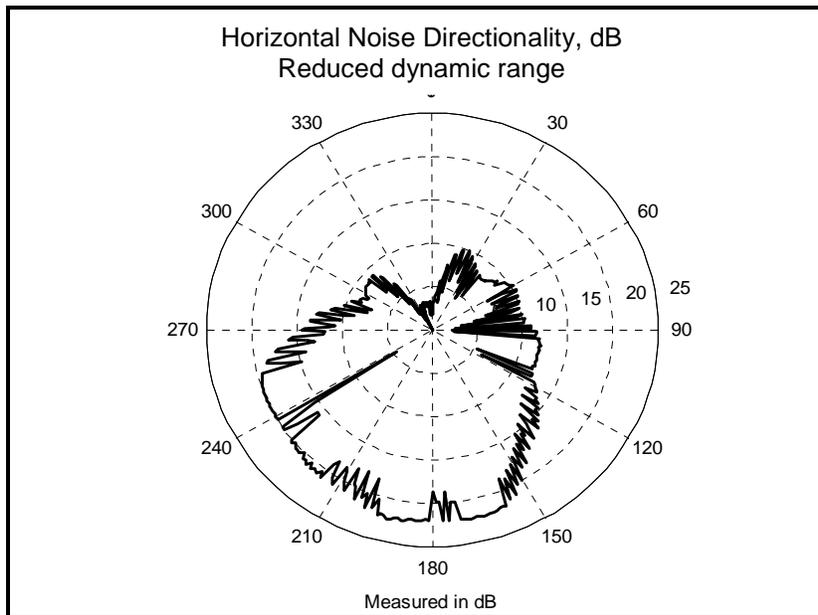
Figure 6-11: Estimated horizontal noise directionality: Case A with reduced dynamic range.
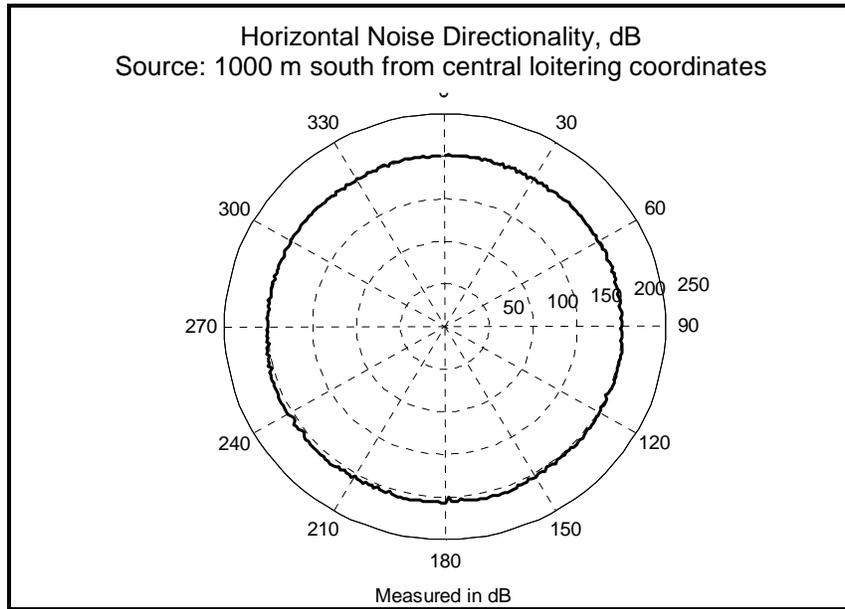
Figure 6-12: Estimated horizontal noise directionality: Case B.
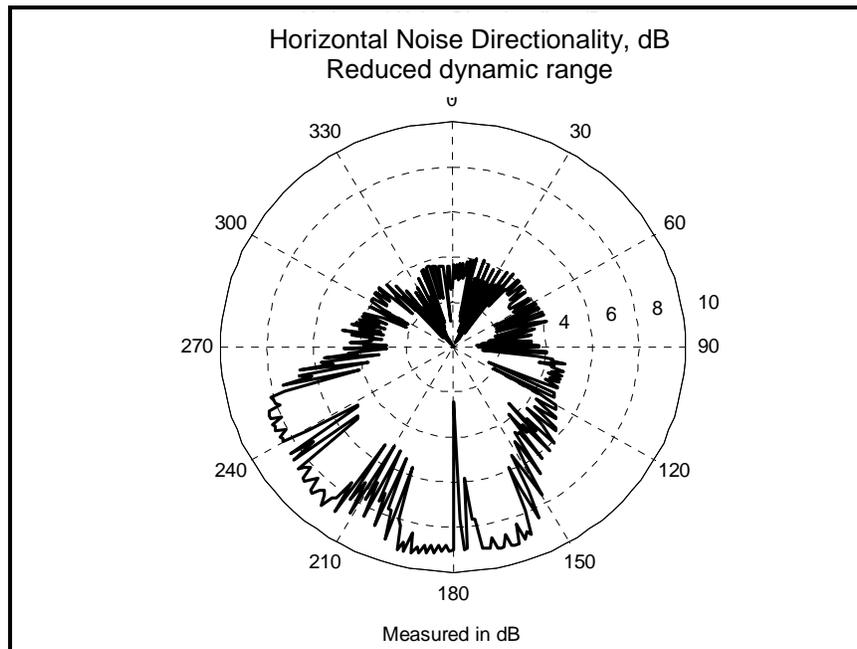
Figure 6-13: Estimated horizontal noise directionality: Case B with reduced dynamic range.

Horizontal Noise Directionality, dB
Source: 2000 m south from central loitering coordinates

330   0   30
300   60
270   50 100 150 200 250   90
240   120
210   150
180

Measured in dB
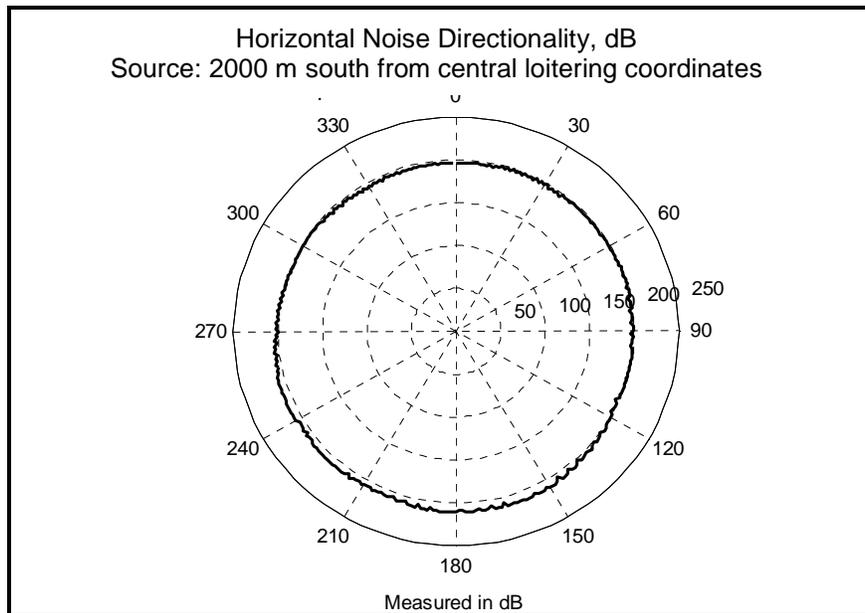
Figure 6-14: Estimated horizontal noise directionality: Case C

Horizontal Noise Directionality, dB
Reduced dynamic range

330   0   30
300   60
270   5 10 15 20   90
240   120
210   150
180

Measured in dB
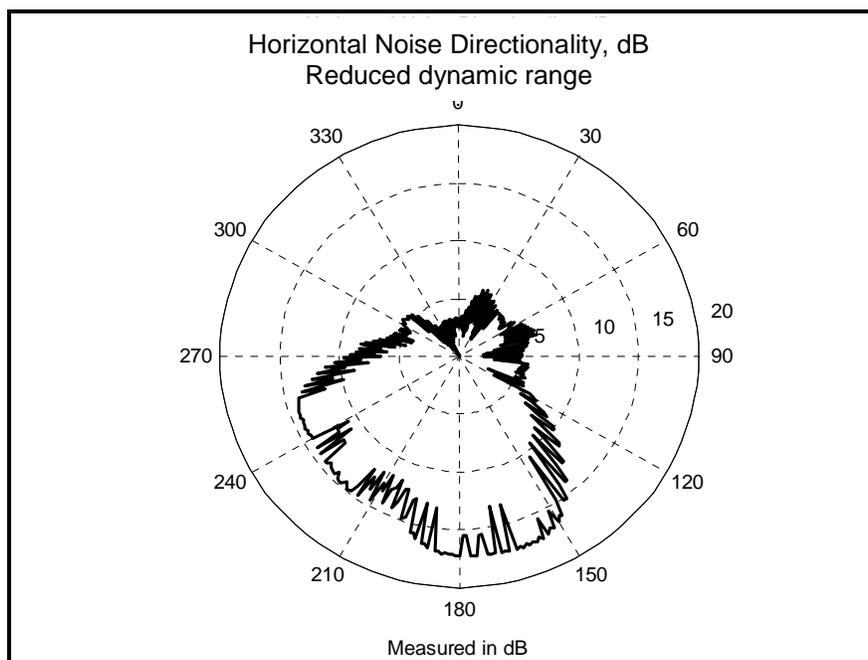
Figure 6-15: Estimated horizontal noise directionality: Case C with reduced dynamic range.
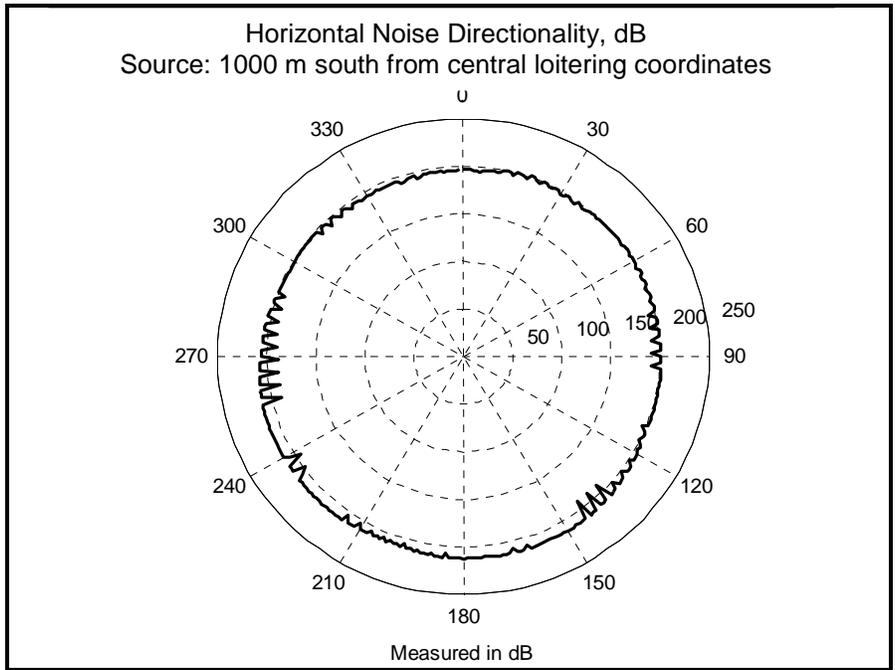
Figure 6-16: Estimated horizontal noise directionality: Case D
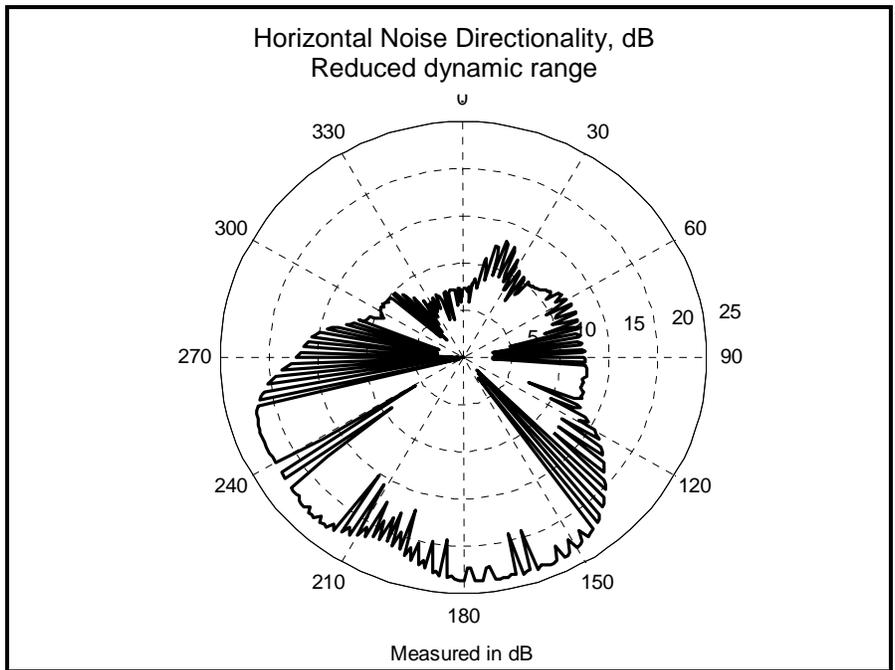
Figure 6-17: Estimated horizontal noise directionality: Case D with reduced dynamic range.

6.2   Optimal towed array heading

When detecting and tracking targets, one of the major goals is to maximize the ratio of target noise to ambient noise in a particular look direction. In order to aid in this, the orientation of the sonar array with respect to the target should be such that the DL is minimized. Keeping in mind the definition of MDL, the objective is to therefore determine the greatest possible DI in the target's direction by changing the heading orientation of the array, with the constraint of avoiding forward or aft endfire.

The MATLAB code in *Appendix B.4* computes the set of DIs of the array for every possible array orientation, using the estimated noise directionalities from Case A, Case B, Case C, and Case D. When a noise field is isotropic, the DI is the same for all array orientations. With the anisotropic noise fields, however, the DIs vary based on the array's orientation, and it is this property that we exploit. Figure 6-18 depicts the DIs for various array headings in Case A's horizontal noise directionality. Disregarding the peaks in DI generated by the endfire beams, Figure 6-18 reveals that different towed
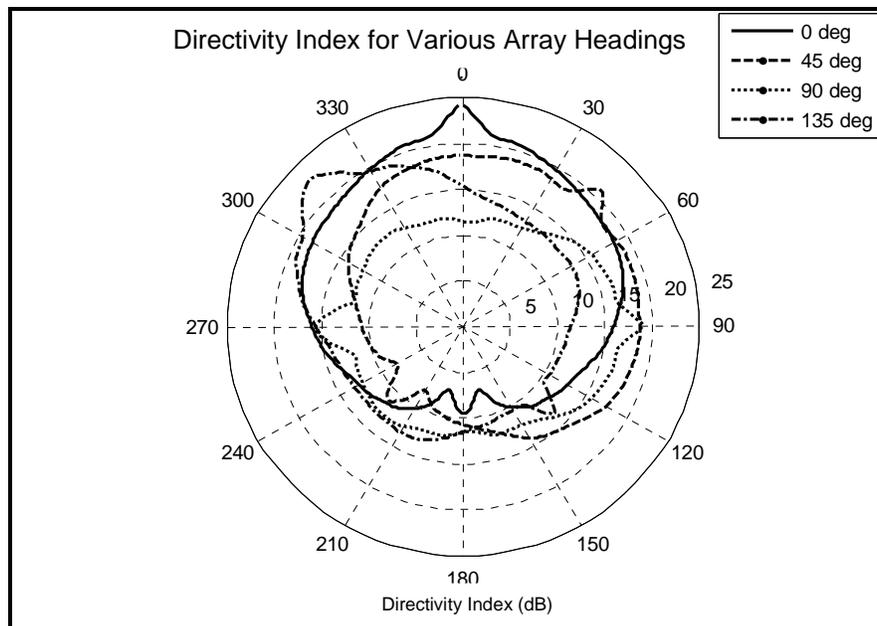
Figure 6-18: DI vs. azimuth for various towed array headings: Case A. The sudden peaks correspond to endfire beams and are not considered when determining the optimal towed array heading.

array headings yield DIs that are clearly greater in specific azimuthal look directions. For instance, for a look direction of 90 degrees, an array heading of 45 degrees (dashed line) clearly provides the best DI. Furthermore, this DI maximum does not correspond to a broadside beam, but actually to the fifth beam back from forward endfire.

Figure 6-19 shows the DIs for various array headings in the noise directionality estimated in Case B. While the DIs from different array headings still vary among each other, the differences are less pronounced than they are in Case A. Figure 6-20 shows the DIs for various array headings in the noise field estimated in Case C. The results have characteristics that fall between those from Case A and Case B: Case C shows greater ranges between the DIs than Case B does, but smaller ranges than Case A. The DIs corresponding to Case D, however, show nearly as much variation as those in Case A, as depicted in Figure 6-21. We can trace the slightly smaller variations in Case D back to the smaller variations in the actual noise directionality estimate for Case D; although the horizontal noise directionality has greater maxima in Case D, the noise leakage and spreading is also greater.
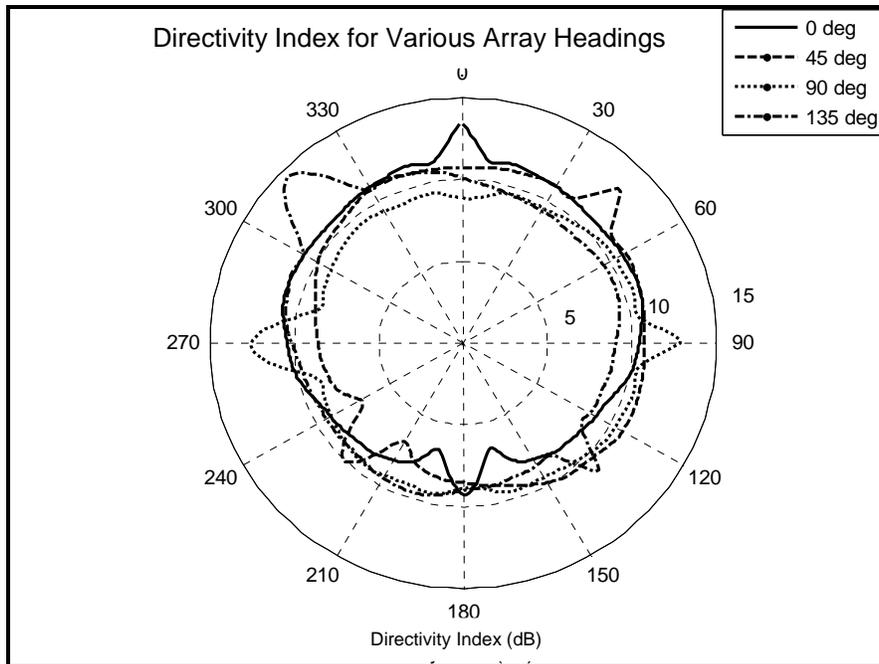
Figure 6-19: DI vs. azimuth for various towed array headings: Case B

57
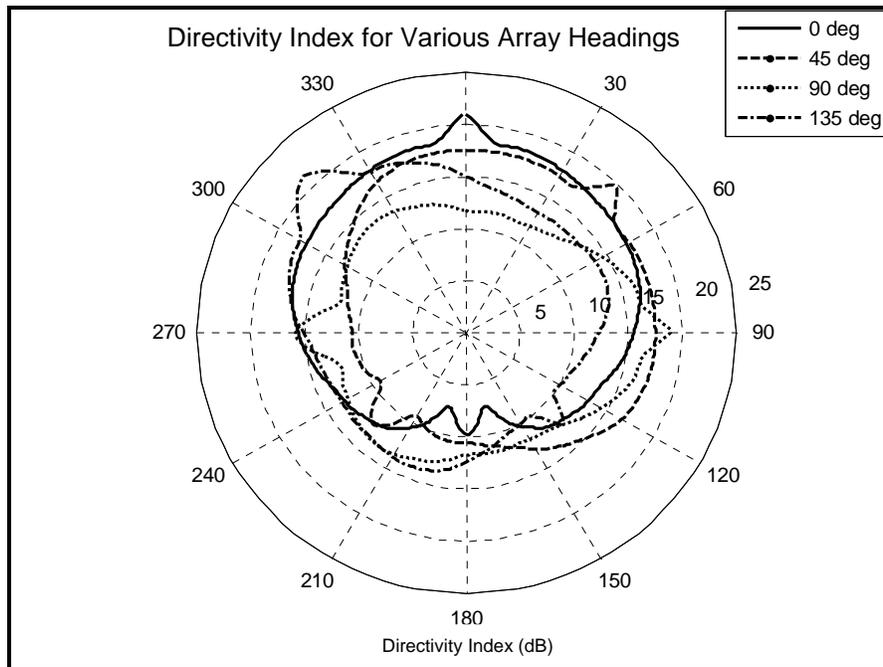
Figure 6-20: DI vs. azimuth for various towed array headings: Case C
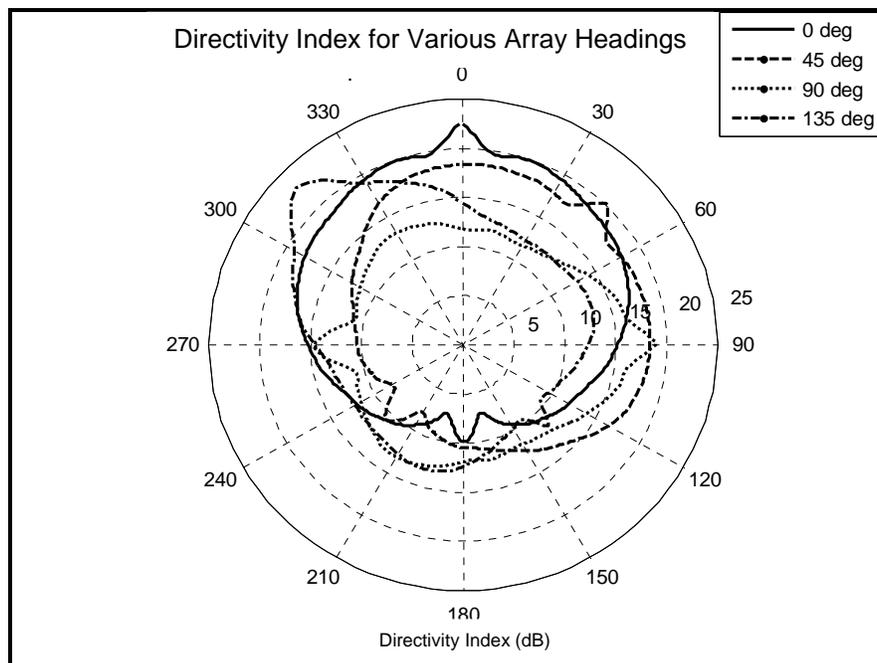
Figure 6-21: DI vs. azimuth for various towed array headings: Case D

Chapter 7

Discussion

Ultimately, the results of the simulation-based experiments presented in this thesis are promising: the WIT algorithm produces reliable estimates of horizontal noise directionality, and the optimal towed array heading improves target tracking capabilities; and both the noise directionality and the optimal towed array heading are determined quickly enough to be useful during real-time operation. The level of advantage that tracking a target via the optimal towed array heading method provides varies with the environment, however, and is an important consideration. A more in depth evaluation follows.

7.1   Effectiveness of noise directionality estimation

As expressed in the 1997 review by [33], many acousticians agree that the WIT algorithm is a reliable method for estimating the pseudo-stationary ambient noise field. The successful noise directionality estimates we obtain using the WIT algorithm in the simulator confirm the opinions in [33]. Each of the estimates correctly reflects acoustic propagation theory in its noise energy spread resulting from the source in its simulated noise field. The extensive studies on waveguide physics in [32] also help verify the accuracy with which the WIT algorithm estimates the noise directionality in the five distinct cases.

Cases A, B, C, and D all have sources generating noise levels that essentially create a simulated anisotropic noise field. The different source characteristics provide grounds for comparison that help determine whether or not the WIT algorithm satisfactorily estimates the directional noise fields for later use to determine an optimal towed array heading. The final noise field estimates successfully reflect the variable source noise levels in each of the anisotropic cases, thus validating the WIT algorithm's noise sensitivity: for instance, Case B has the weakest source, and as expected, its noise field's directionality is the least exaggerated. Additionally, the results also show evidence of the different frequencies that generate the noise fields. Although the source level in Case D is lower than in Case A, the fact that it is a broadband source creates noise levels above the noise floor that are practically equal to those in Case A, in part due to lower attenuation effects over a larger frequency spread. Along the same lines, Case D shows more varied noise energy distribution, again evidence that it is a broadband source that is creating the noise field. Finally, Case C, in which the source is farthest away from the AUV's central loitering coordinates, predictably shows the least amount of noise "leakage" into the northern azimuths.

The review in [33] utilizes a polygon with eight sides—instead of the six used in the PLUSNet operations—for the "loiter pattern," which may affect the quality of the results. Most likely, assuming navigational accuracy, a noise directionality estimate using an eight-sided polygon is more accurate than an estimate using a six-sided polygon. However, the version of the WIT algorithm reviewed in [33] is the older two-dimensional one; and therefore, a degree of superior performance can be attributed to the results in this thesis because the three-dimensional WIT algorithm considers the entire noise field in order to obtain more accurate results.

Nevertheless, the findings in this thesis collectively help us conclude that the WIT algorithm generates sufficiently accurate noise directionality estimates for the ultimate goal of determining an optimal towed array heading. Furthermore, the WIT algorithm proves to be a practical and reliable method with which to create an autonomous behavior for an AUV within a network such as PLUSNet.

7.2 Effectiveness of determining optimal towed array heading

Figures 7-1, 7-2, 7-3, and 7-4 are schematics using Case A's noise directionality estimate that intuitively and qualitatively demonstrate the advantage that an optimal towed array heading orientation provides with respect to maximizing the DI (or minimizing the DL).

Figure 7-1 shows the classical target tracking technique in which the target remains within the sights of the broadside beam for maximum angular resolution and for maximum DI when in an isotropic noise field. It is evident when studying the figure, however, that keeping the target at broadside in this particular noise field is far from desirable, as the ambiguous beam "sees" high noise levels. This noise may threaten to mask the target noise, and ultimately, may cause the tracking system to perform in a substandard manner or even fail. For a direct comparison, Figure 7-2 shows the same target location scenario, but this time the AUV tracks the target employing the optimal towed array heading method. In this case, the sonar array's ambiguous beam "sees" a significantly lower noise level, thus producing a desirably high DI for the array in the target look direction. Figures 7-3 and 7-4 are other target location scenarios in which the optimal towed array heading method is applied. In both of these cases, the ambiguous beam receives lower noise levels, and subsequently, the systems' DIs increase.

Evaluating each of the four anisotropic cases reveals more about the utility of a behavior that determines an optimal towed array heading for an AUV tracking a target within a network of autonomous vehicles. In Case A, the difference between the optimal towed array heading's DI and the broadside DI tends to be most significant in the look directions with lower noise (Table I), which in this case is roughly north. This pattern is consistent with the findings in [9]. The pattern holds true for Cases B, C, and D as well (Tables II, III, IV). Such findings suggest that perhaps the technique of finding the optimal towed array heading is most effective in a noise field with a more significant difference between its maximum and minimum levels. For instance, Case A provides a noise field in which the difference between the maximum and noise floor levels is about 20 dB. In Case A, the largest difference between the optimal and the broadside DI is

61

9.0338 dB (Table I), which is potentially significant advantage for target tracking. The minimum difference in Case A, however, is on the order of 0.01 dB, demonstrating that the technique does not always provide staggering improvements over the conventional broadside beam detection and tracking techniques.

Case B—the simulated noise field generated by the weakest source—further emphasizes the futility of applying this algorithm without a noticeably directional noise field. Case B's maximum DI advantage over the broadside DI is only 2.2301 dB (Table II), as compared to the maximum advantages in Cases A, C, and D. With results as subtle as those from Case B, it is questionable whether or not conducting this algorithm in real-time is worthwhile in environments without prominent directional properties. Cases C and D are more promising, with maximum advantages over the broadside DI of 6.4792 dB and 8.2858 dB, respectively. The lower maximum advantage in Case C is easily predictable since the noise directionality field is less exaggerated than Case A's. It is interesting to note, however, the value of the maximum DI advantage in Case D: the source level generating the noise field in Case D is 10 dB lower than in Case A, but the DI results show values quite close to that of Case A. A broadband source apparently improves the performance of the method.

Programming a behavior that autonomously determines an optimal towed array heading in real-time onboard an AUV is most certainly worthwhile. The degree of anisotropy must be considered, however, when assessing the effectiveness of the method: the more directional the noise field is, the more advantage this tracking method will provide, and *vice versa*. In addition, the method may be responsible for a loss in angular resolution. With a conventional beamformer, the broadside beam possesses the highest angular resolution, as it is the narrowest beam [17]. When an optimal towed array heading places the target in any beam's sight other than broadside, a loss of angular resolution occurs. Depending on the desired target tracking applications, such a consequence may or may not matter; it is up to the user to decide.

TABLE I
Optimal towed array heading data: Case A

Target Bearing (Abs Deg)	Beamno with Max DI	Optimal Tow Heading (Abs Deg)	Beamno with Min DI	Max DI minus Broadside DI
0	26	156/336	14	**9.0338**
45	25	196/16	19	**7.0153**
90	6	37/217	24	**2.3954**
135	16	235/55	26	**0.3661**
180	9	111/291	26	**0.8467**
225	8	289/109	26	**0.6301**
270	16	172/352	25	**0.0472**
315	26	159/339	19	**4.2278**

Note that the maximum advantage over broadside is 9.0338 dB.

TABLE II
Optimal towed array heading data: Case B

Target Bearing (Abs Deg)	Beamno with Max DI	Optimal Tow Heading (Abs Deg)	Beamno with Min DI	Max DI minus Broadside DI
0	2	26/206	14	**2.2301**
45	3	16/196	18	**1.8834**
90	7	32/212	24	**0.4740**
135	11	57/237	26	**0.2580**
180	10	253/73	26	**0.6777**
225	9	294/114	25	**0.5245**
270	15	176/356	25	**0.0669**
315	26	159/339	19	**0.6686**

Note that the maximum advantage over broadside is 2.2301 dB.

TABLE III
Optimal towed array heading data: Case C

Target Bearing (Abs Deg)	Beamno with Max DI	Optimal Tow Heading (Abs Deg)	Beamno with Min DI	Max DI minus Broadside DI
0	26	155/235	15	**6.4792**
45	2	18/198	21	**4.7355**
90	6	38/218	25	**1.8355**
135	10	208/28	26	**0.4120**
180	8	116/296	26	**1.1939**
225	8	290/110	26	**0.7315**
270	12	190/10	23	**0.3031**
315	26	159/239	18	**2.6529**

Note that the maximum advantage over broadside is 6.4792 dB.

TABLE IV
Optimal towed array heading data: Case D

Target Bearing (Abs Deg)	Beamno with Max DI	Optimal Tow Heading (Abs Deg)	Beamno with Min DI	Max DI minus Broadside DI
0	26	156	15	**8.2858**
45	3	16	21	**6.9493**
90	6	36	24	**2.8662**
135	11	213	26	**0.6031**
180	9	113	26	**1.3185**
225	6	276	26	**1.0498**
270	12	189	23	**0.2083**
315	26	159	18	**4.8867**

Note that the maximum advantage over broadside is 8.2858 dB.
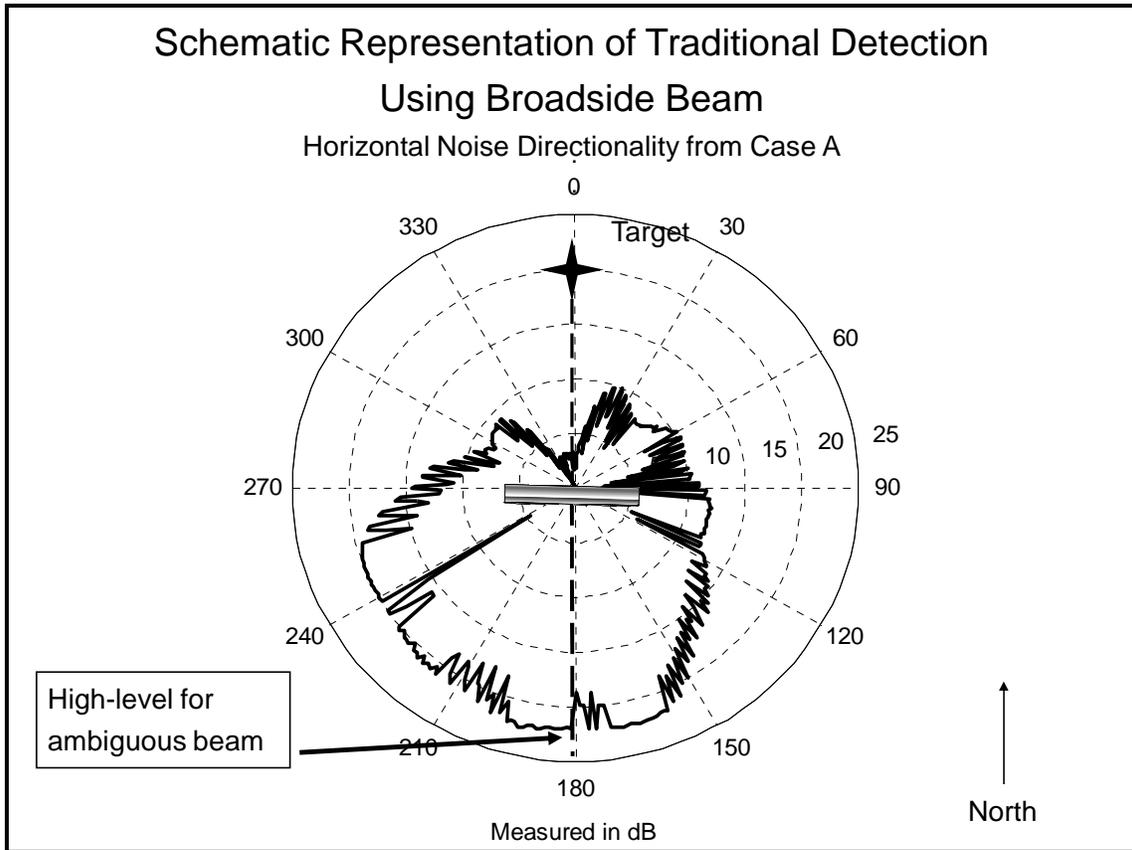
Figure 7-1: Target tracking using broadside beam. The rectangular bar in the middle represents a towed array. Note the high noise level at 180 degrees that the ambiguous beam will detect. This noise may threaten to mask the target while tracking.
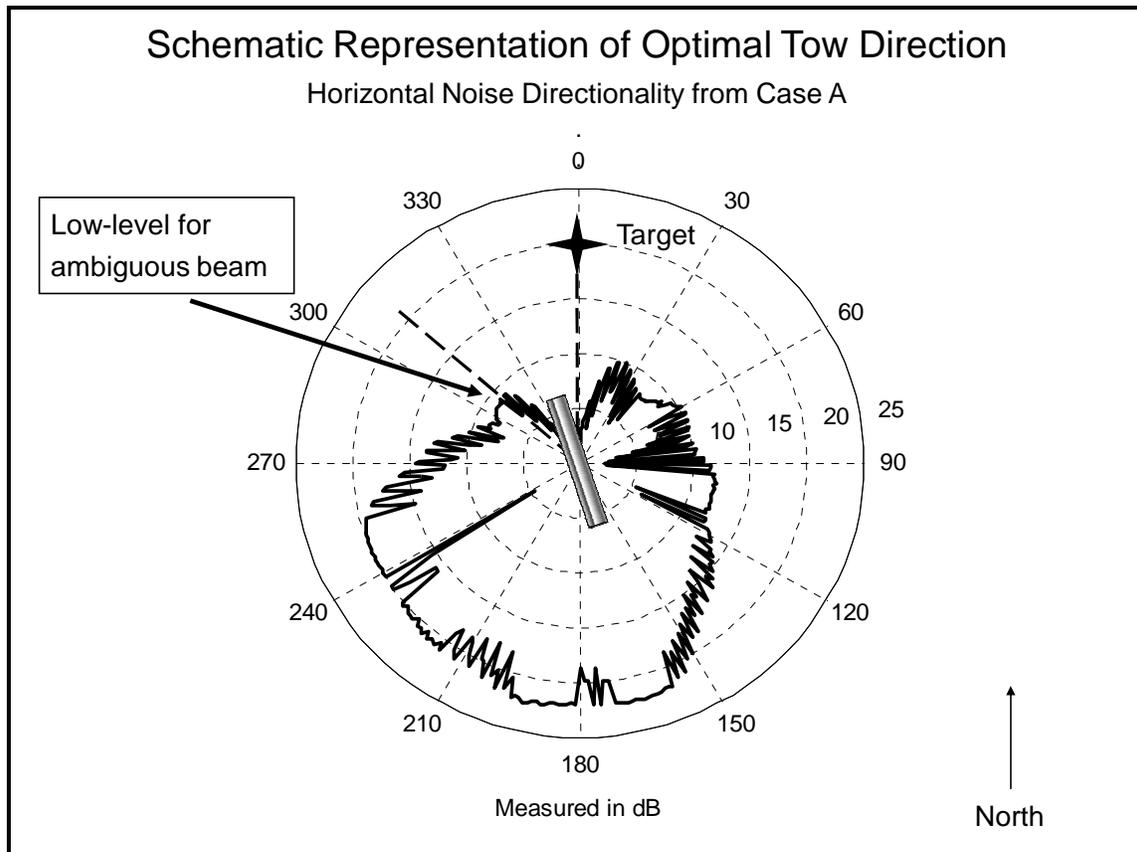
Figure 7-2: Target tracking using optimal towed array heading: Target at 0 degrees. In this scenario, the target is in the same location as in Figure 7-1, but the optimal towed array heading is 156 degrees, putting the target in the sight of the 26th beam. With this orientation, the ambiguous beam sees a much lower noise level, allowing the target noise to stand out more prominently.
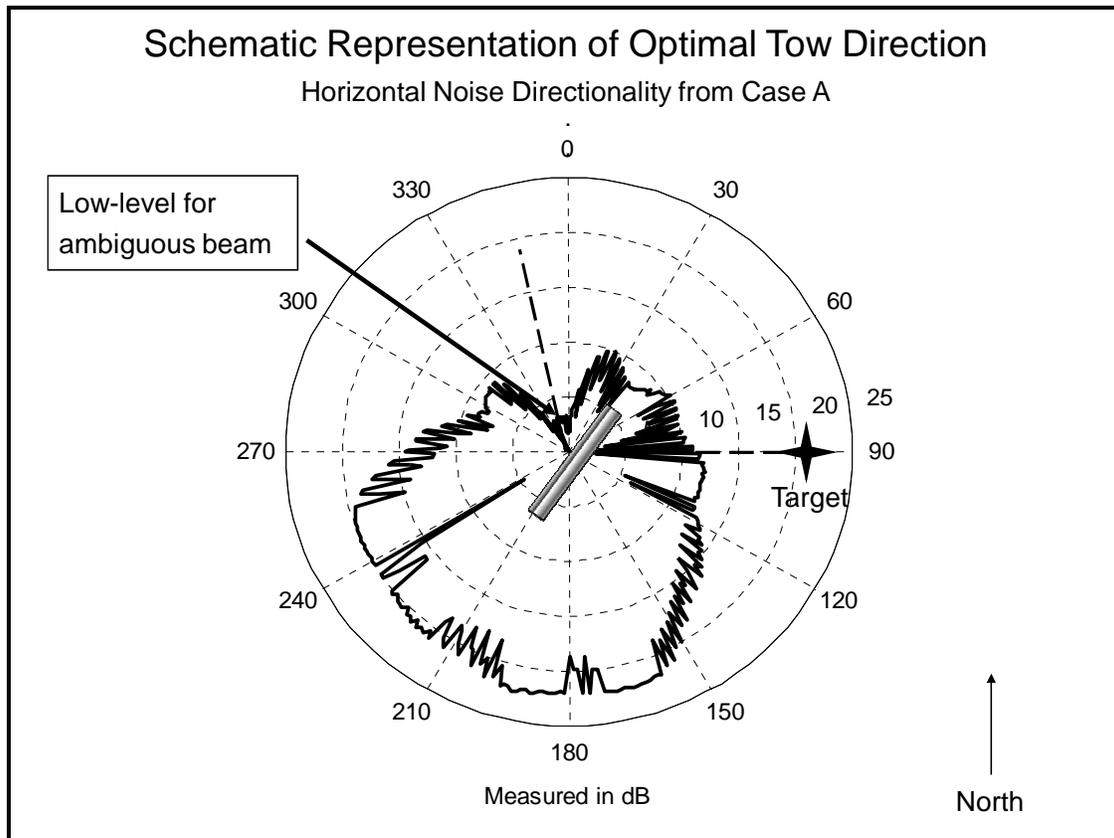
Figure 7-3: Target tracking using optimal tow heading: Target at 90 degrees. The optimal towed array heading in this scenario is 37 degrees, putting the target in sight of the sixth beam. The ambiguous beam sees very low noise levels.
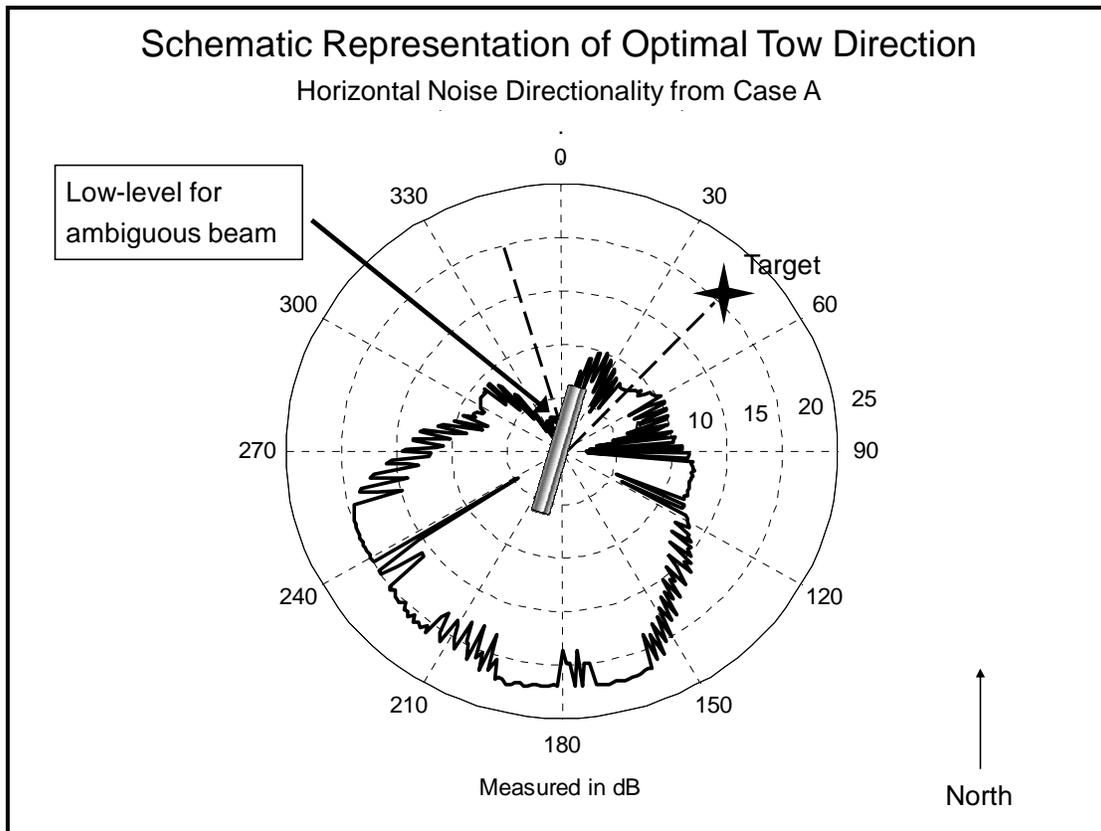
Figure 7-4: Target tracking using optimal tow heading: Target at 45 degrees. The optimal towed array heading in this scenario is 196 degrees, putting the target in sight of the 25th beam. The ambiguous beam sees very low noise levels.

7.3   Theoretical *versus* practical application

Successful implementation of either the WIT algorithm or determining an optimal towed array heading in a full scale experiment is most certainly not far off for a fleet of AUVs. It is important, however, to note that these algorithms may not always be of practical use. Principally, an evaluation of the operational environment is vital: there exist sites, such as some parts of the Norwegian Sea, which do not have stable horizontal noise directionalities. The instabilities may be due to transient fishing fleets or poor propagation conditions in the presence of both nearby and distant shipping lanes [33]. If the directional sources are transient with respect to the time required to estimate the noise directionality as well as the duration of the planned experiment, then those sources may not adversely affect the use of either algorithm. If, however, the directional sources have similar time scales with respect to the time required to estimate the noise directionality or the duration of the planned experiment, then the utility of both algorithms may be compromised. Such environments require knowledge of each and every foreground ship position and source level in order to conduct successful experiments similar to the ones described in this thesis. Barring this particular or any comparable type of environment, an AUV can successfully include the WIT algorithm and the optimal towed array heading method in its behavior suite.

Another aspect to consider when evaluating the practicality of these algorithms in live experiments is that the WIT algorithm ideally calls for continuous data from a closed-circuit path, namely, anything that approaches the perfect shape: a circle. In this thesis, the Loiter behavior pattern is a hexagon, and the heading of the array throughout the algorithm is approximated as the average heading while on each leg of the hexagon. In essence, the hexagon is an approximation for the circle. Perhaps, approximating the circle with a polygon greater than six sides would produce more detailed results. On the other hand, increasing the number of sides in the Loiter pattern might decrease the navigational accuracy. Furthermore, the array might never be straight enough with more waypoints to produce usable, undistorted data without greatly increasing the size of the polygon. The increase in waypoints might also reduce the AUV's consistent ability to hit each waypoint, which is already a slight concern as current, wind, and other weather factors unpredictably affect the environment. It is difficult to say without live

experimentation what the optimal Loiter pattern shape is when considering what is best for the WIT algorithm in conjunction with what is best for the AUV's other missions and the performance capabilities of the AUV.

Yet another factor affecting the application of the algorithms in a live environment is that the equipment on the physical (*versus* virtual) array offers the following sensor information: heading of the first and last hydrophone, as well as the depths of the first and last hydrophones. This is significantly less information than is available in the simulator, which offers x-, y-, and z-coordinates for each and every hydrophone. In the simulator, therefore, the shape of the array is easily and reliably determined. If the array is too curved or significantly tilted, the data can be discarded. In real-time, however, the limited array position information may complicate the assessment of whether the array is straight enough for the data to be usable.

The findings produced by running the WIT algorithm and the optimal towed array heading method are conclusive enough to confidently implement the algorithms as behaviors on an AUV. The behaviors may require slight alterations for use in live environments, but the theoretical basis and the robustness of each behavior is sound. Based on the excellent results of the simulations conducted for this thesis, and barring any unforeseen problems, effectuation of both algorithms on an AUV will improve target tracking in directional noise fields.

Chapter 8

Conclusions and Recommendations

In all simulated anisotropic noise fields, the AUV successfully calculates the optimal towed array headings based on the real-time estimations of the horizontal noise directionalities. A clear advantage over the conventional broadside beam tracking method is revealed, with some limitations due predominately to the noise field itself. Such behaviors can indeed be usefully applied to an AUV surveillance fleet. More testing of the method will reveal the best way in which to implement the theory in a live application. For instance, it may be more helpful to determine an optimal sector for towing the array instead of a single optimal heading to minimize the MDL.

There are several other options that deserve future exploration. Chiefly, use of a Vector Sensor Array (VSA) would certainly eliminate some of the constraints that the WIT algorithm presents due to the use of a standard hydrophone array. Namely, implementation of a VSA eliminates the need for resolving left-right ambiguity since the VSA inherently has the capability to do so [34,35]. Another consideration that might prove to be useful would be to utilize the three-dimensional noise directionality estimation in its entirety. In this thesis, we only consider the horizontal noise directionality, although the full three-dimensional noise directionality is available. Perhaps giving more consideration to vertical noise field levels would enhance the success of the AUV's ability to detect and track a target.

This thesis ultimately demonstrates that the WIT algorithm provides a reliable way for an AUV towing a horizontal line array to estimate the noise directionality, and

based on the estimate, the AUV can determine an optimal towed array heading to improve target tracking. The findings of the optimal towed array heading method show a clear advantage over broadside beam target tracking in terms of minimizing the DL. In order to learn the full advantages that such a behavior may provide in live experiments, it will be necessary to provide scenarios to test the behavior and compare its tracking performance to its broadside beam counterpart.

References

[1] Owen R. Côté, "The Third Battle: Innovation in the US Navy's Silent Cold War Struggle with Soviet Submarines," *Newport Paper*, No. 16., Naval War College: Center for Naval Warfare Studies, Newport, RI, 2003.

[2] Xavier Lurton, *An Introduction to Underwater Acoustics: Principles and Applications*. Chichester: Praxis, 2002.

[3] John R. Benedict, "The Unraveling and Revitalization of the U.S. Navy Antisubmarine Warfare," *Naval War College Review*, Vol. 58, No. 2, Spring 2005.

[4] Mitchell Shipley, "Persistent Littoral Undersea Surveillance Network (PLUSNet) in MB06," Presented in Monterey Bay Aquarium Research Institute, 23 August 2006.

[5] Marc S. Stewart and John Pavlos, "A means to networked persistent undersea surveillance," *Submarine Technology Symposium*, Applied Physics Laboratory, University of Washington, 2006.

[6] Ronald A. Wagstaff, "Iterative technique for ambient-noise horizontal-directionality estimation from towed line-array data," *J. Acoust. Soc. Am*. 63(3), March 1978.

[7] Ronald A. Wagstaff, "Horizontal directionality estimation considering array tilt and noise field vertical arrival structure," *J. Acoust. Soc. Am*. 67(4), April 1980.

[8] Ronald A. Wagstaff and J. Newcomb, "Three-dimensional noise field directionality estimation from single-line towed array data," *J. Acoust. Soc. Am.* 102(2), Part 1, August 1997.

 [9] William A. Kuperman, John S. Perkins, and Kieth Jerome, "Two NSAP/TASP Projects: Northwest Pacific and Mediterranean Sea," Washington, D.C.: Naval Research Laboratory, 1988.

[10] A. H. Nuttall, "Estimation of Noise Directionality Spectrum," Naval Underwater Systems Center TR 4345, Sept. 1972.

[11] A. H. Nuttall, "Resolving the Directional Ambiguities of a Line Array of Hydrophones," Naval Underwater Systems Center TR 4385, Sept. 1972.

[12] J. H. Wilson. "Ambient-noise horizontal directionality measurements with linear arrays," *J. Acoust. Soc. Am.* 60, pp. 955-960, 1976.

[13] E. M. Wilson, "Directional Noise Measurements with Line Arrays," Rep. No. ARL/M/N16, Admiralty Research Laboratory, Teddington, Middlesex, June 1973.

[14] Jean-Louis Berrou, Theo C. Cheston, Patrizio Sesto-Rubino, Ronald Tompkins, and Robert Seynaeve, "An evaluation of a towed-array system and its use in horizontal noise evaluation." SACLANT ASW Research Centre Memorandum SM-134, North Atlantic Treaty Organization, La Spezia, Italy, 1 February 1980.

[15] Ronald A. Wagstaff, Jean-Louis Berrou, and Frederick Cotaras, "Use of the towship for assessing towed-array performance and analyzing data quality." SACLANT ASW Research Centre Report SR-64, North Atlantic Treaty Organization, La Spezia, Italy, 15 November 1982.

[16] Jean-Louis Berrou and Ronald A. Wagstaff, "Virtual beams from an FFT beamformer and their use to assess the quality of a towed-array system," SACLANT ASW Research Centre Memorandum SM-164, North Atlantic Treaty Organization, La Spezia, Italy, 15 January 1983.

[17] Harry L. Van Trees, *Optimum Array Processing (Detection, Estimation, and Modulation Theory, Part IV),* New York: Wiley, 2002.

[18] Bluefin Robotics Corporation, "Bluefin-21 Vehicle Specifications," Available HTTP: http://www.bluefinrobotics.com/bluefin21.htm

[19] Joseph E. Bondaryk, "Towed Array Turns," Bluefin Robotics Corporation, 18 July 2006, unpublished.

[20] LAMSS, "Towed Array for Research into Autonomous and Adaptive Acoustic Surveillance in the Littoral," Specification document of DURIP Array, Autonomous Marine Sensing Systems Lab, MIT, January 2007.

[21] Alex Bahr and Matt Walter, "AUV 101," Presented in LAMSS Seminar, September 2005, Available HTTP: http://mercator.csail.mit.edu/~abahr/powerpoint/AUV_101.ppt

[22] Paul Michael Newman, "MOOS – Mission Oriented Operating Suite," Department of Ocean Engineering, Massachusetts Institute of Technology, 2007.

[23] Rodney A. Brooks, "A Robust Layered Control System for a Mobile Robot," *IEEE Journal of Robotics and Automation*, RA-2(1): 14-23, April 1986.

[24] J. K. Rosenblatt, "DAMN: A Distributed Architecture for Mobile Navigation," Ph.D. dissertation, Carnegie Mellon University, Pittsburgh, PA, 1997.

[25] P. Perjanian, "Multiple Objective Action Selection and Behavior Fusion," Ph.D. dissertation, Aalborg University, 1998.

[26] Michael R. Benjamin, "Multi-Objective Navigation and Control Using Interval Programming," Multi-Robot Systems Workshop, NRL, Washington DC, March 2003.

[27] Michael R. Benjamin, "A Guide to the IvP Helm for Autonomous Marine Vehicle Control," NAVSEA Division Newport, RI, Massachusetts Institute of Technology, 28 Feb 2007.

[28] Michael R. Benjamin, "Interval Programming: A Multi-Objective Optimization Model for Autonomous Vehicle Control," Ph.D. dissertation, Brown University, Providence, RI, May 2002.

[29] Michael R. Benjamin and J. Curcio, "COLREGS-Based Navigation in Unmanned Marine Vehicles," In *AUV-2004*, Sebasco Harbor, Maine, June 2004.

[30] Donald P. Eickstedt and Michael R. Benjamin, "Cooperative Target Tracking in a Distributed Autonomous Sensor Network," In *Proceedings of OCEANS 2006*, Boston, September 2006.

[31] Michael R. Benjamin, "Autonomy Algorithms for Adaptive Control of Unmanned Marine Vehicles," Presented in MIT Seminar, 18 April 2007.

[32] Finn B. Jensen, William A. Kuperman, Michael B. Porter, and Henrik Schmidt, *Computational Ocean Acoustics*, New York: AIP Press, 2000.

[33] R. M. Hamson, "The modeling of ambient noise due to shipping and wind sources in complex environments," *Applied Acoustics*, Vol. 51, No. 3, pp. 251-287. Surrey, UK, 1997.

[34] Bruce M. Abraham, "Ambient noise measurements with vector acoustic hydrophones," In *IEEE Oceans '06,* Boston, MA, 2006.

[35] A. Nehorai and E. Paldi, "Acoustic vector-sensor array processing," *IEEE Trans. on Signal Processing*, Vol. SP-42, pp. 2481-2491, Sept. 1994.

Appendix A: Simulation Environment

The PLUSNet simulator combines acoustic modeling, platform dynamics, and network communication and control. Before executing a live experiment, we link the physics-based simulations to sensor simulations to help plan and analyze the actual oceanographic experiments. The simulated environment contains as many features as possible in order to emulate a live environment such as a propagation model, sound-speed profile, bottom parameters, and bathymetry. Platforms are also simulated by modeling sensors on vehicles, vehicle dynamics, sonar array dynamics and capabilities, target location, and target characteristics. Experimental simulated results can include transmitted signals, received signals, detection and classification of targets, transmission loss, scattering, vehicle direction, vehicle velocity, *etc.*

The simulator used for PLUSNet experiments utilizes a comprehensive sonar simulation toolbox which ties together a wide array of acoustic models, including acoustic propagation models, target scattering, ambient noise, and reverberation. This toolbox is called Synthetic Environment Acoustics Laboratory (SEALAB) ocean acoustics modeling and simulation [1].

The propagation model used in the simulator is based on SACLANTCEN Normal Mode Acoustic Propagation Model (SNAP) [2], which is a normal mode propagation model developed at SACLANT Undersea Research Center. SNAP is designed to give a realistic treatment of the ocean environment, including arbitrary sound-speed profiles in both water column and bottom, compressional and shear wave attenuation, scattering at rough boundaries, and range dependence. A newer version of SNAP, called Coupled SACLANTCEN Normal Mode Acoustic Propagation Model (C-SNAP) [3], combined with exact Fourier decomposition of the azimuthal dependence of the scattered field, enables high-fidelity modeling of three-dimensional propagation in shallow water with complex bathymetry [4]. C-SNAP has been modified for implementation on UNIX platforms by Schmidt.

The simulation environment utilizes a version of the Kuperman-Ingenito noise model for the ambient noise [5,6]. The Kuperman-Ingenito noise model provides a wave

theory treatment of noise propagation in terms of continuous and discrete modes of the propagation channel. The original model [5] is valid only in range independent applications; however, the updated model presented in [6] includes range dependent propagation.

The SEALAB acoustic simulation framework is linked with a real-time MOOS simulator, generating element-level time series using Green's functions using the environmental model, C-SNAP.

Appendix A References

[1] VASA Associates, "SEALAB Brochure," Available HTTP:
     www.vasaassociates.com/doc/SEALAB_brochure.pdf

[2] Finn B. Jensen and M. C. Ferla, "SNAP: The SACLANTCEN Normal-Mode Acoustic
     Propagation Model," SACLANTCEN Memorandum SM-121, SACLANT Undersea
     Research Center, La Spezia, Italy, Jan. 1979.

[3] C. M. Ferla, M. B. Parter, and Finn B. Jensen, "C-SNAP: Coupled SACLANTCEN
     normal-mode propagation loss model," SACLANTCEN Memorandum SM-274,
     SACLANT Undersea Research Center, La Spezia, Italy, Dec. 1993.

[4] W. Luo, "A Three-Dimensional Coupled Mode Solution for Range-Independent
     Waveguides," MS Thesis, MIT, Feb. 2005.

[5] William A. Kuperman and F. Ingenito, "Spatial correlation of surface generated noise
     in a stratified-ocean," *J. Acoust. Soc. Am.* 67, 1988-1996 (1980).

[6] J. Perkins, William A. Kuperman, F. Ingenito, and L. Fialkowski, "Modeling ambient
     noise in three-dimensional ocean environments," *J. Acoust. Soc. Am.* 93, pp. 739-
     752, 1993.

Appendix B: MATLAB Code

Following is the MATLAB code used to (1) generate the three-dimensional beampattern responses, (2) record the necessary beam outputs due to a simulated noise field, (3) process the data using the WIT algorithm; and, in turn, apply the results of the WIT algorithm to (4) find an optimal towed array heading for tracking a target. Each of these four sections are presented separately, accompanied by a description of how each step is accomplished.

While MATLAB is sufficiently efficient in a simulation environment, it is not powerful enough for use in real-time with limited computational power. Lower level languages, such as C++, are much more efficient; and so transforming the MATLAB code in B.1, B.2, B.3, and B.4 into a lower level language is ideal for live applications.

B.1   Calculating beampatterns in spherical coordinates

This segment of MATLAB code does the following:

1. Defines the number of hydrophones on the sonar array, the distance between them, and the number of beams for the beamformer.

2. Defines the desired frequency range.

3. Defines the particular beam being formed and calculates it using a conventional beamforming algorithm in conical angle.

4. Relates conical angle to spherical coordinates.

5. Maps the beampattern values in conical angle to their corresponding locations in spherical coordinates, creating a three-dimensional beam pattern response. This step is broken down into two separate `for` loops for computational efficiency.

6. Normalizes and plots the beampatterns in the visible spectrum.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Maria Parra-Orlandoni
% Beam Response Pattern in Spherical Coordinates
% Conventional beamformer for DURIP Array
% Based on MB06 data
% 2007
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all; close all;

% azimuth - CW from x-axis (towards y-axis)
theta = 0:pi/180:2*pi - pi/180;
% vertical - up from horizontal
phi = -pi + pi/180:pi/180:pi;

H = 13;                         % # hydrophone elements
I = 27;                         % # beams
%I = def_beams;                 % # beams defined in pBearings

% if d > lambda, grating lobes appear in visible region
d = 0.75;                       % distance between 1 kHz hydrophones (m)
% d = 1.5;                      % distance between 500 Hz hydrophones (m)

c = 1480;                       % average soundspeed, m/s

% beam "1" = front endfire; beam "-1" = aft endfire
u_T = linspace(1,-1,I);
theta_T = acos(u_T);

% tilt angle measured from x-axis in deg
tilt = 0*pi/180;

% position vector: assume COG of array is center, so sum(Pn) = 0
h = 0:H - 1;
Ph = (h - (H - 1)/2)*d;         % [1 x H]

% def_centerfreq = 896;
% def_bandwidth = 192;
% bin_size = def_samplerate/def_fftlength;
bin_size = 1;                   % approximated beamsize
f = 810:bin_size:982;           % Hz
```

```matlab
% initialize values
lambda = 0;
k = zeros(1,360);
v = zeros(H,360);
k_T = zeros(1,I);
v_T = zeros(H,I);
w = zeros(H,I);
B = zeros(I,360);
C = zeros(360);
r_a = zeros(360,360,180);
r_b = zeros(360,360,180);
b_a = zeros(360,360);
b_b = zeros(360,360);
b_temp = zeros(360,360,60);
b = zeros(360);

% beamnumber being calculated (1 through 27)
bn = 14;


for kk = 1:length(f)

    lambda = c/f(kk);                   % wavelength
    k = -(2*pi/lambda)*cos(theta);      % wavenumber

    % manifold vector
    v = exp(j*(Ph'*k));                 % [H x 720]

    % steering
    for q = 1:I

        % target wavenumber
        k_T(q) = -(2*pi/lambda)*u_T(q);
        v_T(:,q) = exp(j*Ph'*k_T(q));       % [H x N]
        w = (1/H)*v_T;                      % [H x N]

    end

    % visible region: 0 < theta < pi
    B = w'*v;

    % Relation between conical and spherical coordinates
    for ii = 1:length(theta)
        for jj = 1:length(phi)

            C(ii,jj) = acos(cos(theta(ii))*cos(phi(jj))*cos(tilt)
            +sin(acos(cos(theta(ii))*cos(phi(jj))))*sin(tilt)*...
            sin(atan(sin(phi(ii))/cos(phi(ii))*sin(theta(ii)))));

        end
    end


    C = round(C*100)/100;
```

```matlab
        theta = round(theta*100)/100;
        phi = round(phi*100)/100;

        % find location of conical angles in spherical coordinate map
        % break up into 2 "for" loops to speed up computation
        for ii = 1:180

                % locate theta = C (1 matrix per theta value); multiply by
                % B at theta
                r_a(:,:,ii) = eq(theta(ii),C)*B(bn,ii);
                b_a = b_a + r_a(:,:,ii);

        end

        for ii = 1:180

                % locate theta = C (1 matrix per theta value); multiply by
                % B at theta
                r_b(:,:,ii) = eq(theta(ii + 180),C)*B(bn,ii + 180);
                b_b = b_b + r_b(:,:,ii);

        end

        b_temp(:,:,kk) = b_a + b_b;
        b = b + b_old(:,:,kk);

end

% normalize and save only visible space
b14 = b/max(max(b));
b14 = b14(91:270,:);
save b14

figure(100)
imagesc(theta,phi(91:270),abs(b14))
colorbar
xlabel('\theta (rad)'); ylabel('\phi (rad)');
title({'Beampattern in Spherical Coordinates'; ['Beam # ',num2str(bn)];
['Tilt = ',num2str(tilt),' rad']});
```

B.2 Obtaining Data for Noise Directionality Estimation

This segment of MALTAB code is merely an extension of `small_uVis.m`, written by Dr. Henrik Schmidt. The new additions to the `small_uVis.m` code do the following:

1. Call on the necessary MOOSDB variables.

2. Loads the stored library of three-dimensional beam responses.

3. Creates an iteration count variable (called `kk` in the code).

4. Determines whether sonar array has a less than 5 degree pitch. Also determines whether array is straight enough based on the standard deviation of hydrophone positions.

5. Records published BTR data and records headings for each segment of loiter pattern hexagon. Each segment is delineated by variable `Hits`.

6. Averages the headings for each segment of the loiter pattern hexagon.

7. Separates the BTR data accordingly for each of the loiter pattern hexagon's segments.

8. Averages the BTR data over time.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Henrik Schmidt, with additions written by Maria Parra-Orlandoni
% Record data for estimation of horizontal noise directionality
% Based on MB06 data
% 2007
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all;

iMatlab('INIT','MOOSNAME','noise_dir','CONFIG_FILE','NoiseDir.moos')
```

```
iMatlab('MOOS_PAUSE',0.1);

% MINUS'07 box
xbmin = -500; xbmax = 4500;
ybmin = -500; ybmax = 4500;

v_id = NaN;
node = 'Unknown';

x = NaN;
y = NaN;
z = NaN;
target_x = NaN;
target_y = NaN;
target_heading = NaN;
target_speed = NaN;

target_xpos = NaN;
target_ypos = NaN;
xb = NaN;
yb = NaN;
xo = NaN;
yo = NaN;
xp = NaN;
yp = NaN;
zp = NaN;
bstate = NaN;
newx = NaN;
newy = NaN;
head = NaN;
rhead = NaN;
towpX = NaN;
towpY = NaN;
towpZ = NaN;
water_depth = NaN;
got_arrayX = 0;
got_arrayY = 0;
got_arrayZ = 0;
array_on = 0;
time_old = 0;
nhist = 100;
ihist = 0;
pstate = '          ';
btrfile = ' ';
nbeam = 0;
nbtr = 0;
frame = 0;
mark = 'm.';
msize = 5;
newbtr = 0;
mission_time = 0;
start_time = 0;
```

```
frame_time = 0;
ael_pitch = NaN;
ael_heading = NaN;
sample_freq = NaN;
fft_length = NaN;
loiter_index = NaN;
loiter_acquire = NaN;
target_heading = NaN;
Hits = 0;


loiter_report = '    ';


II = 27;                    % # beams
JJ = 6;                     % # headings - hydrophone array
H = 13;                     % # elements

theta = 0:pi/180:2*pi - pi/180;       % azimuth (360)
phi = -pi/2 + pi/180:pi/180:pi/2;     % vertical (180)
c = cos(phi);                         % [1 x 180]


stdX = 0; stdY = 0; stdZ = 0;
r0 = [];
r1 = [];
r2 = [];
r3 = [];
r4 = [];
r5 = [];
q1 = zeros(II,JJ);
heading_0 = 0; heading_1 = 0; heading_2 = 0;
heading_3 = 0; heading_4 = 0; heading_5 = 0;
h0_avg = 0; h1_avg = 0; h2_avg = 0;
h3_avg = 0; h4_avg = 0; h5_avg = 0;
b = [];
b_int = [];
rHat = zeros(II,JJ);
del = zeros(II,JJ);
b_err = [];
error = zeros(length(phi),length(theta));
err = [];
n_horiz = zeros(1,length(theta));


% noise directionality for ith beam in jth direction
% discretized along azimuth (row) and vertical angle (column)
% assume isotropic noise for zeroth iteration (initialization)
n = ones(length(phi),length(theta));    % [180 x 360]


% load spherical coordinate beampatterns
load b_0_1KHZ


kk = 0;


mail=iMatlab('MOOS_MAIL_RX');
```

```matlab
while(1)

    %Keep track of time so we can iterate every .8 s
    clockTime1 = clock;

    % Extract message content
    mail = iMatlab('MOOS_MAIL_RX');
    messages = length(mail);

    if messages == 0
        continue;
    end

    for m = 1:messages

        key = mail(m).KEY;
        val = mail(m).DBL;
        str = mail(m).STR;

        switch key

            case 'VEHICLE_ID'
                v_id = floor(val);
                if (v_id == 3)
                    node = 'Unicorn';
                else
                    if (v_id == 4)
                        node = 'Macrura';
                    end
                    node = 'Unknown';
                end


            case 'NAV_HEADING'
                head = val;

            case 'NAV_DEPTH'
                z = val;

            case 'NAV_X'
                x = val;
                newx = 1;

            case 'NAV_Y'
                y = val;
                newy = 1;

            case 'BATHY_Z'
                water_depth = -val;
```

```
case 'TOW_POS_X'
    towpX = val;
    tfl(1) = 1;


case 'TOW_POS_Y'
    towpY = val;
    tfl(2) = 1;


case 'TOW_POS_Z'
    towpZ = val;
    tfl(3) = 1;


case 'HYDROPHONE_X'
    ac_arrayX = str;
    got_arrayX = 1;


case 'HYDROPHONE_Y'
    ac_arrayY = str;
    got_arrayY = 1;


case 'HYDROPHONE_Z'
    ac_arrayZ = str;
    got_arrayZ = 1;


case 'TARGET_XPOS'
    target_xpos = val;


case 'TARGET_YPOS'
    target_ypos = val;


case 'TARGET_X'
    target_x = val;


case 'TARGET_Y'
    target_y = val;


case 'TARGET_HEADING'
    target_heading = val;


case 'TARGET_SPEED'
    target_speed = val;


case 'BEARING_STAT'
    bStat = str;
    bStat = sscanf(bStat,'node=%d,state=%d,bearing=%f,…
            xp=%f,yp=%f,beamno=%d,sigma=%f,time=%f');

    if(length(bStat) >= 6)

        Bearing1 = bStat(3);
```

```matlab
            Bstate = bStat(2);

        end

    case 'PROSECUTE_STATE'
        pstate = str;

    case 'AEL_PITCH'
        ael_pitch = val;

    case 'AEL_HEADING'
        ael_heading = val;

    case 'SAMPLE_FREQ'
        sample_freq = val;

    case 'FFT_LENGTH'
        fft_length = val;

    case 'LOITER_INDEX'
        loiter_index = val;

    case 'LOITER_ACQUIRE'
        loiter_acquire = val;

    case 'LOITER_REPORT'
        loiter_report = str;
        [Pt Dist Hits NM_Hits AQ_MODE] =
        strread(loiter_report,...
            '%*s %d %*s %f %*s %d %*s %d %*s %s',…
            'delimiter',':');

    case 'TARGET_HEADING'
        target_heading = val;

    case 'BTR_DATA'
        beams=str2num(str)';
        %'
        nbeam=length(beams)
        nbtr=nbtr+1;
        if (nbtr == 1)
            btr = beams;
        else
            btr = [btr beams];
        end
        newbtr = 1;

    end %"switch…"

end %"for m=1:messages..."
```

90

```
gfc = figure(1);
if (newx + newy == 2)
    newx = 0; newy = 0;
    % set current time
    moos_time = now;
    if (start_time == 0)
        start_time == moos_time;
    else
        mission_time = moos_time - start_time
    end

    % update auv position history
    if (ihist < nhist)
        ihist=ihist+1;
        xhist(ihist) = x;
        yhist(ihist) = y;
        zhist(ihist) = z;
    else
        xhist(1:nhist-1) = xhist(2:nhist);
        yhist(1:nhist-1) = yhist(2:nhist);
        zhist(1:nhist-1) = zhist(2:nhist);
        xhist(nhist) = x;
        yhist(nhist) = y;
        zhist(nhist) = z;
    end

    % Absolute position of the hydrophones of the towed array
    if ((got_arrayX + got_arrayY + got_arrayZ) == 3)
        arrayX = towpX + str2num(ac_arrayX);
        arrayY = towpY + str2num(ac_arrayY);
        arrayZ = -towpZ + str2num(ac_arrayZ);
        xb1 = mean(arrayX);
        yb1 = mean(arrayY);
        array_on = 1;
    else
        xb1=xp;
        yb1=yp;
    end

    %          figure(1);
    subplot(3,2,[1 3]);
    plot([xp+5*cos(rhead) xp+100*cos(rhead)],[yp+5*sin(rhead)…
        yp+100*sin(rhead)],'w');
    rhead = pi/2-head*pi/180;
    plot([x+5*cos(rhead) x+100*cos(rhead)],[y+5*sin(rhead)…
        y+100*sin(rhead)],'k');
    if (array_on == 1)
        h = plot(arrayX,arrayY,'b');
        set(h,'Linewidth',2);
    end
    if (bstate > 1)
        mark = 'r.';
        h = plot(x,y,mark);
```

91

```
        msize = 3;
        set(h,'MarkerSize',msize);
        set(h,'Linewidth',2);
else
        if (pstate(1:5) == 'PROSE')
            mark = 'm.';
            h = plot(x,y,mark);
            msize = 3;
            set(h,'MarkerSize',msize);
            set(h,'Linewidth',2);
        else
            mark = 'm.';
            h = plot(x,y,mark);
            msize = 3;
            set(h,'MarkerSize',msize);
            set(h,'Linewidth',2);
        end
end

xlabel('x');
ylabel('y');
h = title(node);
set(h,'FontSize',16);
axis([xbmin xbmax ybmin ybmax]);
grid on
hold on
if (bstate == 0)
    plot(target_xpos, target_ypos, 'co')
else
    plot(target_xpos, target_ypos, 'bo')
end

% Plot target track
H = plot(target_x, target_y, 'r.');
set(h,'Markersize',20);
thv_a = (90.0-target_heading)*pi/180.0;
thv_x = target_x+250*target_speed*cos(thv_a);
thv_y = target_y+250*target_speed*sin(thv_a);
h = plot([target_x thv_x],[target_y thv_y],'r');
set(h,'Linewidth',2);
h = plot([thv_x thv_x+150*cos(thv_a+160*pi/180)], ...
    [thv_y thv_y+150*sin(thv_a+160*pi/180)],'r');
set(h,'Linewidth',2);
h = plot([thv_x thv_x+150*cos(thv_a-160*pi/180)], ...
    [thv_y thv_y+150*sin(thv_a-160*pi/180)],'r');
set(h,'Linewidth',2);

if (bstate > 1)
    plot([xo xb],[yo yb],'w');
    xo = x + (xb - x)*0.05; yo = y + (yb - y)*0.05;
    bear = (90 - Bearing1)*pi/180;
    xb = x + 2000*cos(bear);
    yb = y + 2000*sin(bear);
```

```matlab
        plot([xo xb],[yo yb],'g');
    else
        if (bstate == 0)
            plot([xo xb],[yo yb],'w');
        end
    end

    h = text(xbmin + 500,ybmax - 500,['Time = '…
        num2str(mission_time) 's']);
    h = text(xbmin + 100,ybmax - 300,datestr(now));
    set(h,'FontSize',12);
    set(h,'BackgroundColor','w');
    drawnow


%      figure(2)
    subplot(3,2,[6]);
    hold off;
    plot([x-125 x+125],[0 0],'b');
    hold on
    h = plot([x-125 x+125],[-water_depth -water_depth],'g');
    set(h,'Linewidth',2);
    plot(xhist,-zhist,'m');
    h = plot(x,-z,mark);
    set(h,'MarkerSize',20);
    if (array_on == 1)
        h = plot([x arrayX(1)],[-z -arrayZ(1)],'b');
        h = plot(arrayX,-arrayZ,'b');
        set(h,'Linewidth',3);
    end
    h = title(node);
    set(h,'FontSize',16);
    h = xlabel('x');
    set(h,'FontSize',14);
    h = ylabel('z');
    set(h,'FontSize',14);
    axis([ x-125 x+125 -100 10]);
    grid on
    drawnow

    subplot(3,2,[5]);
    hold off;
    h = plot([x x+20*cos(rhead)],[y y+20*sin(rhead)],'k');
    set(h,'Linewidth',2);
    hold on
    plot(xhist,yhist,'m');
    h = plot(x,y,mark);
    set(h,'MarkerSize',20);
    if (array_on == 1)
        h = plot(arrayX,arrayY,'b');
        set(h,'Linewidth',3);
        h = plot([x arrayX(1)],[y arrayY(1)],'b');
    end
    if (bstate > 1)
```

```
        bear = (90-Bearing1)*pi/180;
        xb2 = xb1+100*cos(bear);
        yb2 = yb1+100*sin(bear);
        plot([xb1 xb2],[yb1 yb2],'g');
    end
    h = title(node);
    set(h,'FontSize',16);
    h = xlabel('x');
    set(h,'FontSize',14);
    h = ylabel('y');
    set(h,'FontSize',14);
    axis([ x-125 x+125 y-125 y+125]);
    axis equal;
    grid on
    drawnow

    xp = x; yp = y; zp = z;

end %"if newx+newy..."

if (nbtr > 0 & newbtr == 1)
    %      figure(3)
    subplot(3,2,[2 4]);

    nrows = min(100,nbtr);
    bmno = [1:nbeam];
    abhm = acos(1.0-(bmno-1)/(nbeam-1)*2.0)*180.0/pi;
    db = zeros(nbeam,100);
    for ii = 1:nbeam
        for jj = 1:100
            db(ii,jj) = NaN;
        end
    end
    if (nbtr > nrows)
        btrno = [nbtr-nrows+1:nbtr]';
        %'
        db = dbp(btr(:,nbtr-nrows+1:nbtr));
    else
        btrno = [nbtr-100+1:nbtr]';
        %'
        db(:,100-nrows+1:100)=dbp(btr(:,nbtr-nrows+1:nbtr));
    end
    wavei(db',abhm,btrno);
    %'
    shading('interp');
    axis ij;
    h = xlabel('Bearing (deg)');
    set(h,'Fontsize',14);
    h = ylabel('Frame');
    set(h,'Fontsize',14);
    drawnow;

    % make movies
```

94

```
        frame = frame+1;
        M = getframe(gfc);
        eval(['save frame_' num2str(frame) ' M']);
        newbtr = 0;
        frame_time = mission_time;
    else
        if ((nbtr == 0 & mission_time-frame_time)>= 4)
            frame = frame+1;
            M = getframe(gfc);
            eval(['save frame_' num2str(frame) ' M']);
            newbtr = 0;
            frame_time = mission_time;
        end


end %"if (nbtr > 0...)"


%-------------------------------------------------------------------
% Noise Directionality Estimation: Obtain Necessary Data
%-------------------------------------------------------------------
% measured beam response [i x j] from BTR files (relative heading)
% bmno --> beam number
% btrno --> file number
% db --> output in dB

% start recording AFTER 1st point
% stop recording after j = 6 different headings

if loiter_acquire == 0

    % record if tilt <= 5 deg
    if (Hits == 2) && (abs(ael_pitch) <= 5)

        % Absolute position of the hydrophones of the towed array
        % back from tow point
        arrayX = towpX + str2num(ac_arrayX);
        % left/right of tow point
        arrayY = towpY + str2num(ac_arrayY);
        % up/down from tow point
        arrayZ = -towpZ + str2num(ac_arrayZ);

        % standard deviation of hydrophone positions
        detX = detrend(arrayX);
        stdX = std(detX);
        detY = detrend(arrayY);
        stdY = std(detY);
        stdZ = std(arrayZ);

        % record only when array is straight-ish
        if (stdY < 1) && (stdZ < 1)

            % BTR file in dB
            r0 = btr;
```

```
        heading_0(kk) = ael_heading;

    end

    sr0 = size(r0)

elseif (Hits == 3) && (abs(ael_pitch) <= 5)

    % Absolute position of the hydrophones of the towed array
    arrayX = towpX + str2num(ac_arrayX);
    arrayY = towpY + str2num(ac_arrayY);
    arrayZ = -towpZ + str2num(ac_arrayZ);

    % standard deviation of hydrophone positions
    detX = detrend(arrayX);
    stdX = std(detX);
    detY = detrend(arrayY);
    stdY = std(detY);
    stdZ = std(arrayZ);

    if (stdY < 1) && (stdZ < 1)

        % BTR file in dB
        r1 = btr;
        heading_1(kk) = ael_heading;

    end

    sr1 = size(r1)

elseif (Hits == 4) && (abs(ael_pitch) <= 5)

    % Absolute position of the hydrophones of the towed array
    arrayX = towpX + str2num(ac_arrayX);
    arrayY = towpY + str2num(ac_arrayY);
    arrayZ = -towpZ + str2num(ac_arrayZ);

    % standard deviation of hydrophone positions
    detX = detrend(arrayX);
    stdX = std(detX);
    detY = detrend(arrayY);
    stdY = std(detY);
    stdZ = std(arrayZ);

    if (stdY < 1) && (stdZ < 1)

        % BTR file in dB
        r2 = btr;
        heading_2(kk) = ael_heading;

    end
```

```matlab
    sr2 = size(r2)

elseif (Hits == 5) && (abs(ael_pitch) <= 5)

    % Absolute position of the hydrophones of the towed array
    arrayX = towpX + str2num(ac_arrayX);
    arrayY = towpY + str2num(ac_arrayY);
    arrayZ = -towpZ + str2num(ac_arrayZ);

    % standard deviation of hydrophone positions
    detX = detrend(arrayX);
    stdX = std(detX);
    detY = detrend(arrayY);
    stdY = std(detY);
    stdZ = std(arrayZ);

    if (stdY < 1) && (stdZ < 1)

        % BTR file in dB
        r3 = btr;
        heading_3(kk) = ael_heading;

    end

    sr3 = size(r3)

elseif (Hits == 6) && (abs(ael_pitch) <= 5)

    % Absolute position of the hydrophones of the towed array
    arrayX = towpX + str2num(ac_arrayX);
    arrayY = towpY + str2num(ac_arrayY);
    arrayZ = -towpZ + str2num(ac_arrayZ);

    % standard deviation of hydrophone positions
    detX = detrend(arrayX);
    stdX = std(detX);
    detY = detrend(arrayY);
    stdY = std(detY);
    stdZ = std(arrayZ);

    if (stdY < 1) && (stdZ < 1)

        % BTR file in dB
        r4 = btr;
        heading_4(kk) = ael_heading;

    end

    sr4 = size(r4)
```

```
        elseif (Hits == 7) && (abs(ael_pitch) <= 5)

            % Absolute position of the hydrophones of the towed array
            arrayX = towpX + str2num(ac_arrayX);
            arrayY = towpY + str2num(ac_arrayY);
            arrayZ = -towpZ + str2num(ac_arrayZ);

            % standard deviation of hydrophone positions
            detX = detrend(arrayX);
            stdX = std(detX);
            detY = detrend(arrayY);
            stdY = std(detY);
            stdZ = std(arrayZ);

            if (stdY < 1) && (stdZ < 1)

                % BTR file in dB
                r5 = btr;
                heading_5(kk) = ael_heading;

            end

            sr5 = size(r5)

        end % "if (Hits == ..."

    elseif loiter_acquire == 1

        l_a = 1;

    end % "if loiter_acquire..."

    %----------------------------------------------------------------------
    % Clean up the data for processing
    %----------------------------------------------------------------------

    if (Hits >= 8)

        % get rid of zeros in heading records
        i0 = find(heading_0);
        i1 = find(heading_1);
        i2 = find(heading_2);
        i3 = find(heading_3);
        i4 = find(heading_4);
        i5 = find(heading_5);

        heading_0 = heading_0(i0);
        heading_1 = heading_1(i1);
        heading_2 = heading_2(i2);
        heading_3 = heading_3(i3);
```

```matlab
heading_4 = heading_4(i4);
heading_5 = heading_5(i5);


% time avg, round to nearest degree while on each side of hex
for i0 = 1:length(heading_0) - 1
    if abs(heading_0(i0 + 1) - heading_0(i0)) > 270
        for j0 = 1:length(heading_0)
            if (heading_0(j0) >= 0) && (heading_0(j0) < 90)

                heading_0(j0) = heading_0(j0) + 360;

            end
        end
    end
end

h0_avg = round(mean(heading_0));

if h0_avg > 360

    h0_avg = h0_avg - 360

end

for i1 = 1:length(heading_1) - 1
    if abs(heading_1(i1 + 1) - heading_1(i1)) > 270
        for j1 = 1:length(heading_1)
            if (heading_1(j1) >= 0) && (heading_1(j1) < 90)

                heading_1(j1) = heading_1(j1) + 360;

            end
        end
    end
end

h1_avg = round(mean(heading_1));

if h1_avg > 360

    h1_avg = h1_avg - 360

end

for i2 = 1:length(heading_2) - 1
    if abs(heading_2(i2 + 1) - heading_2(i2)) > 270
        for j2 = 1:length(heading_2)
            if (heading_2(j2) >= 0) && (heading_2(j2) < 90)

                heading_2(j2) = heading_2(j2) + 360;
```

```
                end
            end
        end
end

h2_avg = round(mean(heading_2));

if h2_avg > 360

    h2_avg = h2_avg - 360

end

for i3 = 1:length(heading_3) - 1
    if abs(heading_3(i3 + 1) - heading_3(i3)) > 270
        for j3 = 1:length(heading_3)
            if (heading_3(j3) >= 0) && (heading_3(j3) < 90)

                heading_3(j3) = heading_3(j3) + 360;

            end
        end
    end
end

h3_avg = round(mean(heading_3));

if h3_avg > 360

    h3_avg = h3_avg - 360

end

for i4 = 1:length(heading_4) - 1
    if abs(heading_4(i4 + 1) - heading_4(i4)) > 270
        for j4 = 1:length(heading_4)
            if (heading_4(j4) >= 0) && (heading_4(j4) < 90)

                heading_4(j4) = heading_4(j4) + 360;

            end
        end
    end
end

h4_avg = round(mean(heading_4));

if h4_avg > 360
```

```matlab
        h4_avg = h4_avg - 360

    end

    for i5 = 1:length(heading_5) - 1
        if abs(heading_5(i5 + 1) - heading_5(i5)) > 270
            for j5 = 1:length(heading_5)
                if (heading_5(j5) >= 0) && (heading_5(j5) < 90)

                    heading_5(j5) = heading_5(j5) + 360;

                end
            end
        end
    end

    h5_avg = round(mean(heading_5));

    if h5_avg > 360

        h5_avg = h5_avg - 360

    end

    % put beampatterns in relative angle
    b1 = circshift(b_0_1KHz,[0 h0_avg 0]);
    b2 = circshift(b_0_1KHz,[0 h1_avg 0]);
    b3 = circshift(b_0_1KHz,[0 h2_avg 0]);
    b4 = circshift(b_0_1KHz,[0 h3_avg 0]);
    b5 = circshift(b_0_1KHz,[0 h4_avg 0]);
    b6 = circshift(b_0_1KHz,[0 h5_avg 0]);

    b = cat(4,b1,b2,b3,b4,b5,b6);
    sb = size(b)

    % separate BTR files according to leg of hexagon
    r0 = r0(:,sr0(2)-10:sr0(2)); srp0 = size(r0)
    r1 = r1(:,sr0(2)+1:sr1(2)); srp1 = size(r1)
    r2 = r2(:,sr1(2)+1:sr2(2)); srp2 = size(r2)
    r3 = r3(:,sr2(2)+1:sr3(2)); srp3 = size(r3)
    r4 = r4(:,sr3(2)+1:sr4(2)); srp4 = size(r4)
    r5 = r5(:,sr4(2)+1:sr5(2)); srp5 = size(r5)

    % avg beam response power
    r0 = mean(r0,2);
    r1 = mean(r1,2);
    r2 = mean(r2,2);
    r3 = mean(r3,2);
    r4 = mean(r4,2);
    r5 = mean(r5,2);
```

```matlab
        % [i x j]
        q1 = [r0 r1 r2 r3 r4 r5];

        save q1

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % Here, run "algorithm" code, and then "TowOpt" code
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    end %"if Hits..."

    kk = kk + 1;

end %"while length(mail)"
```

B.3 Estimating horizontal noise directionality

This segment of MATLAB code does the following:

1. Loads the estimated noise fields calculated previously using the WIT algorithm. This would be an unnecessary when the entire process was done in real-time, for the data would be utilized immediately after calculation.

2. Loads the stored library of three-dimensional beam responses as well the library of two-dimensional beam responses.

3. Computes the DI for each beam for all sonar headings to one degree of accuracy. This includes correcting the resulting DI functions so that they are in absolute heading.

4. Adjusts all beampatterns so that they are in absolute coordinates.

5. Calculates the estimated beam output noise intensities. Then it finds the error between the estimate and the measured values.

6. Maps the error to their corresponding spherical coordinate domain.
7. Normalizes the distributed error map based on energy conservation.

8. Adds the corresponding error to the estimated noise field and repeats step 5 through step 8 until the error falls below the desired threshold.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Estimate Noise Directionality based on WIT algorithm
% Maria Parra-Orlandoni
% 2007
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all; close all;

II = 27;                                % # beams
JJ = 6;                                 % # headings - hydrophone array
H = 13;                                 % # elements
```

```
theta = 0:pi/180:2*pi-pi/180;        % azimuth (360)
phi = -pi/2+pi/180:pi/180:pi/2;      % vertical (180)
dtheta = [diff(theta) pi/180];
dtheta2D = ones(length(phi),length(theta))*(pi/180);
dphi = [diff(phi) pi/180];
dphi2D = ones(length(phi),length(theta))*(pi/180);


c = cos(phi);                             % [1 x 180]


nq = ones(length(phi),length(theta));     % [180 x 360]
Nq = zeros(length(phi),length(theta));


% load spherical coordinate beampatterns
load('b_0_1KHZ')
load('B_1D')


% load Beam Intensity Outputs obtained from AUV in Loiter Pattern
load('q')                    % Case A
mean_q = mean(mean(q));


% put 2D beampattern in relative angle
% circshift based on average headings computed in noise_dir.m named:
% h0_avg,h1_avg,h2_avg,h3_avg,h4_avg,h5avg


% Case A
b1q = circshift(b_0_1KHz,[0 68 0]);
b2q = circshift(b_0_1KHz,[0 107 0]);
b3q = circshift(b_0_1KHz,[0 179 0]);
b4q = circshift(b_0_1KHz,[0 248 0]);
b5q = circshift(b_0_1KHz,[0 290 0]);
b6q = circshift(b_0_1KHz,[0 5 0]);


bq = cat(4,b1q,b2q,b3q,b4q,b5q,b6q);
bq_power = abs(bq).*abs(bq);


% 1D beampatterns
B1q = circshift(B,[0 68]);
B2q = circshift(B,[0 107]);
B3q = circshift(B,[0 179]);
B4q = circshift(B,[0 248]);
B5q = circshift(B,[0 290]);
B6q = circshift(B,[0 5]);


Bq_cat = cat(3,B1q,B2q,B3q,B4q,B5q,B6q);
Bq_power = abs(Bq_cat).*abs(Bq_cat);


%-------------------------------------------------------------------
% initialize algorithm element values
% beam power integrated over all azimuth and all vertical angles
% each rHat(i,j) is [1 x 1]; rHat will be [I x J]
```

```
% Case A
delq = ones(II,JJ)*100;
delq_meas = ones(II,JJ)*100;
qHat = zeros(II,JJ);
DELq = ones(II,JJ)*100;
QHAT = zeros(II,JJ);


% while error for any cell is greater than threshold, continue
while (abs(max(max(delq_meas))) >= 0.5)

      errorQ = zeros(length(phi),length(theta));

      for i1 = 1:II
            for j1 = 1:JJ

                  bq_int = squeeze(bq_power(:,:,i1,j1));
                  qHat(i1,j1) = (1/(4*pi))*sum(sum(bq_int.*nq…
                              .*dtheta2D.*dphi2D,2).*c',1);

            end
      end

      % error between estimated and measured beam power: [i x j]
      qDB = dbp(q);                % beam power output in dB
      QHAT = dbp(qHat);            % beam power output estimate in dB
      DELq = (qDB - QHAT);         % error in dB
      delq_meas = 10.^(DELq/10);   % error in power
      delq = delq_meas/2;          % successive underrelaxation

      %map errors to corresponding theta-phi cells
      for ii = 1:II
            for jj = 1:JJ

                  errQ_1D = Bq_power(ii,:,jj)*delq(ii,jj);
                  intQ_1D = sum(errQ_1D.*dtheta);
                  errQ_2D = bq_power(:,:,ii,jj)*delq(ii,jj);
                  intQ_2D = (1/(4*pi))*sum(sum(errQ_2D…
                          .*dtheta2D.*dphi2D,2).*c',1);
                  % energy preservation
                  intQ_norm = intQ_1D/intQ_2D;
                  errQ_2D_norm = errQ_2D*intQ_norm;
                  errorQ = errorQ + errQ_2D_norm;

            end
      end

      nq = nq + errorQ;
      Nq = dbp(nq);


      Nq_horiz = Nq(90,:);
```

```
      figure(1)
      polar(theta,Nq_horiz)
      title('Horizontal Noise Directionality, dB')
      drawnow


end %"while max..."

figure(2)
Ndiff = Nq_horiz - Nr_horiz;
polar(theta,Ndiff)
title('Anisotropic minus Isotropic Noise Directionality')

% Plots of beam power outputs

qDB_plot(:,1) = circshift(qDB(:,1),19);
qDB_plot(:,2) = circshift(qDB(:,2),10);
qDB_plot(:,3) = circshift(qDB(:,3),1);
qDB_plot(:,4) = circshift(qDB(:,4),19);
qDB_plot(:,5) = circshift(qDB(:,5),10);
qDB_plot(:,6) = circshift(qDB(:,6),0);

figure(3)
subplot(3,2,1);plot(qDB_plot(:,1));axis([1 27 120 160]);grid on
subplot(3,2,2);plot(qDB_plot(:,2));axis([1 27 120 160]);grid on
subplot(3,2,3);plot(qDB_plot(:,3));axis([1 27 120 160]);grid on
ylabel('Power (dB)');
subplot(3,2,4);plot(qDB_plot(:,4));axis([1 27 120 160]);grid on
subplot(3,2,5);plot(qDB_plot(:,5));axis([1 27 120 160]);grid on
xlabel('Beamnumber');
subplot(3,2,6);plot(qDB_plot(:,6));axis([1 27 120 160]);grid on
xlabel('Beamnumber');


figure(4)
subplot(3,2,1);plot(qDB(:,1));axis([1 27 120 160]);grid on
subplot(3,2,2);plot(qDB(:,2));axis([1 27 120 160]);grid on
subplot(3,2,3);plot(qDB(:,3));axis([1 27 120 160]);grid on
ylabel('Power (dB)');
subplot(3,2,4);plot(qDB(:,4));axis([1 27 120 160]);grid on
subplot(3,2,5);plot(qDB(:,5));axis([1 27 120 160]);grid on
xlabel('Beamnumber');
subplot(3,2,6);plot(qDB(:,6));axis([1 27 120 160]);grid on
xlabel('Beamnumber');
```

B.4 Determining optimal towed array heading

This segment of MATLAB code does the following:

1. Loads the estimated noise fields calculated previously using the WIT algorithm. This would be an unnecessary when the entire process was done in real-time, for the data would be utilized immediately after calculation.

2. Loads the stored library of three-dimensional beam responses.

3. Computes the DI for each beam for all sonar headings to one degree of accuracy. This includes correcting the resulting DI functions so that they are in absolute heading.

4. Determines a target heading, and based on that, eliminated the use of any endfire beams for detecting and tracking.

5. Determines the heading that produces the maximum DI in the look direction of the target.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Optimize Tow Direction based on Horizontal Noise Directionality
% Maria Parra-Orlandoni
% 2007
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all; close all;

% load estimated noise field: Case A
load('nq')                          % [180 x 360]
nq_norm = nq/(max(max(nq)));        % normalize noise field

% load spherical coordinate beampatterns
load('b_0_1KHZ')                    % [180 x 360 x 27]
b_power = abs(b_0_1KHz).*abs(b_0_1KHz);

II = 27;                            % # beams

% beam "1" = front endfire; beam "-1" = aft endfire
u_T = linspace(1,-1,II);
theta_T = acos(u_T);
```

107

```
theta = 0:pi/180:2*pi-pi/180;       % azimuth
phi = 0+pi/180:pi/180:pi;           % vertical angle
dtheta2D = ones(length(phi),length(theta))*(pi/180);
dphi2D = ones(length(phi),length(theta))*(pi/180);
s = sin(phi');                      % [180 x 1]


% compute directivity for each beam and each look direction
for ii = 1:II
       for look = 1:360


             b = circshift(b_power(:,:,ii),[0 (look-1) 0]);
             % Directivity in anisotropic noise
             DI_temp(:,:,look,ii) = 1/((1/4*pi)*sum(sum(b.*nq_norm…
                                    .*dtheta2D.*dphi2D,2).*s,1));


       end
end


DI_beam = squeeze(DI_temp);


% interpolate to switch from beamnumber domain to angle domain
for u = 1:360


       DI_ang(u,:) = interp(DI_beam(u,:),20);


end


DI_ang = DI_ang(:,1:3:540);


% full 360 degree function
DI_ang2 = fliplr(DI_ang);
DI_ang_power = cat(2,DI_ang,DI_ang2);


DI_ang_dB = dbp(DI_ang_power);


% put in absolute coordinates based on heading
for jj = 1:360


       DI_ang_dB(jj,:) = circshift(DI_ang_dB(jj,:),[0 (jj-1)]);


end


% plot DIs for various headings
figure(1)
polar(theta,DI_ang_dB(1,:),'b')
hold on; grid on
polar(theta,DI_ang_dB(46,:),'r')
polar(theta,DI_ang_dB(91,:),'c')
polar(theta,DI_ang_dB(136,:),'m')
legend('0 deg','45 deg','90 deg','135 deg')
```

```matlab
title('Directivity Index for Various Array Headings')
ylabel('Directivity Index (dB)')
% flips polar coordinates to desired orientation
%view(28.65*pi,-24*pi)


% for a specific target
%target_angle = round(target_angle);      % from MOOSDB
target_angle = 345;                       %deg, between 0 and 359


theta_deg = 1:360;
DI_tgt = [DI_ang_dB(:,(target_angle + 1)) theta_deg'];


% avoid endfire
if (target_angle >= 0) && (target_angle <= 23)

    DI_tgt = DI_tgt([(target_angle + 25):(target_angle + 157)
            (target_angle + 205):(target_angle + 337)],:);

elseif (target_angle >= 24) && (target_angle <= 155)

    DI_tgt = DI_tgt([1:(target_angle - 23) (target_angle + 25):
            (target_angle + 157) (target_angle + 205):(360)],:);

elseif (target_angle >= 156) && (target_angle <= 203)

    DI_tgt = DI_tgt([(target_angle - 155):(target_angle - 23)
            (target_angle + 25):(target_angle + 157)],:);

elseif (target_angle >= 204) && (target_angle <= 335)

    DI_tgt = DI_tgt([1:(target_angle - 203) (target_angle - 155):
            (target_angle - 23) (target_angle + 25):(360)],:);

elseif (target_angle >= 336) && (target_angle <= 359)

    DI_tgt = DI_tgt([(target_angle - 335):(target_angle - 203)
            (target_angle - 155):(target_angle - 23)],:);

end


[DImax,max_dir] = max(DI_tgt(:,1));


% determine optimal tow direction
OptTow = theta_deg(DI_tgt(max_dir,2)) - 1
OptTow_rad = OptTow*pi/180;
DImax
```