NAVAL POSTGRADUATE SCHOOL
Monterey, California

DISSERTATION

ACOUSTIC BASED TACTICAL CONTROL OF UNDERWATER VEHICLES

by

William J. Marr

June 2003

Dissertation Supervisor: Anthony J. Healey

Approved for public release; distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 2003	3. REPORT TYPE AND DATES COVERED Ph.D. Dissertation

4. TITLE AND SUBTITLE: Acoustic Based Tactical Control of Underwater Vehicles	5. FUNDING NUMBERS
6. AUTHOR(S) William J. Marr	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT ***(maximum 200 words)***

Advances in command and control of Autonomous Underwater Vehicles (AUVs) using acoustic communications are crucial to future Fleet objectives, particularly in Very Shallow Water Mine Countermeasures (VSW MCM). Understanding of the capability to redirect missions, provide relatively high rate downloads of mission information, and perform cooperative tracking for multi-vehicle systems is limited to some bounding data based on fixed node experiments.

The main objectives of this dissertation were to investigate and demonstrate the capabilities of tactical acoustic control of a dynamic, operational underwater vehicle in the Very Shallow Water (VSW) and Shallow Water ocean environment. This necessarily required studies on the limitations of Acoustic Control and relatively High Data Rate Transfer when using commercial acoustic modems in underwater vehicles and investigation of their acoustic transmission characteristics. Comprehensive empirical evidence through field validation with the ARIES vehicle indicated that reduced ranges were required for successful acoustic communications in a realistic shallow water environment. A simulation was developed to demonstrate a solution for dealing with reduced range and conducting multi-vehicle behaviors for cooperative tracking and acoustic data transfer.

14. SUBJECT TERMS Autonomous Underwater Vehicles, Acoustics, Acoustic Control, Acoustic Data Transfer, Modems	15. NUMBER OF PAGES 198
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited.

ACOUSTIC BASED TACTICAL CONTROL OF UNDERWATER VEHICLES

William J. Marr
Commander, United States Navy
B.S.M.E., United States Naval Academy, 1981
M.S.M.E., Naval Postgraduate School, 1994

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN MECHANICAL ENGINEERING
from the
NAVAL POSTGRADUATE SCHOOL
June 2003

Author: _____
William J. Marr

Approved by:

_____
Anthony J. Healey
Professor of Mechanical Engineering
Dissertation Supervisor

_____
Roberto Cristi
Professor of Electrical Engineering

_____
Morris Driels
Professor of Mechanical Engineering

_____
Fotis Papoulias
Professor of Mechanical Engineering

_____
Joshua Gordis
Professor of Mechanical Engineering

Approved by: _____
Young Kwon, Chairman, Department of Mechanical Engineering

Approved by: _____
Carson K. Eoyang, Associate Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Advances in command and control of Autonomous Underwater Vehicles (AUVs) using acoustic communications are crucial to future Fleet objectives, particularly in Very Shallow Water Mine Countermeasures (VSW MCM). Understanding of the capability to redirect missions, provide relatively high rate downloads of mission information, and perform cooperative tracking for multi-vehicle systems is currently limited to some bounding data based on fixed node experiments while the impact of working in the environment presented by a moving vehicle is not understood.

The main objectives of this dissertation were to investigate and demonstrate the capabilities of tactical acoustic control of a dynamic, operational underwater vehicle in the Very Shallow Water (VSW) ocean environment. This necessarily required studies on the limitations of Acoustic Control and relatively High Data Rate Transfer when using commercial acoustic modems in underwater vehicles, and an investigation of their acoustic transmission characteristics. Comprehensive empirical evidence through field validation with the ARIES vehicle indicated that reduced ranges were required for successful acoustic communications in a realistic very shallow water environment. Background noise, multipath reflections, and vehicle induced Doppler shifts all limit the communication link. Occasionally, configurations may be found where vehicle body shielding against multipath destructive interference can be used to advantage. A simulation was developed to demonstrate a solution for reducing the range and conducting multi-vehicle behaviors for cooperative tracking and acoustic communications data transfer.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION 1
A. BACKGROUND 1
B. DISCUSSION 2
C. SCOPE OF WORK 7

II. COMMUNICATIONS THEORY REVIEW 9
A. GENERAL UNDERWATER COMMUNICATIONS SYSTEM 9
B. INFORMATION SOURCE AND SOURCE ENCODING 11
C. CHANNEL ERROR CORRECTION CODING 11
1. Block Coding 12
a. Linear Block Codes 13
b. Cyclic Codes 14
c. Advanced Cyclic Codes 15
2. Convolutional Coding 16
3. Advanced Coding Methods 22
D. MODULATION AND TRANSMISSION 23
1. Noncoherent 24
2. Coherent 27
3. Advanced Modulation Techniques 28
a. Frequency Hopping (FH) 28
b. Multiple Access 28
4. Transmission 29
a. Frames or Packets 29
b. Physical Transmission and Reception 30
E. THE PHYSICAL CHANNEL AND ENVIRONMENTAL FACTORS 31

III. ACOUSTIC CONTROL 35
A. FAU MODEM OPERATING PARAMETERS 37
1. Basic Specifications 37
2. Encoding Schemes 38
3. Modulation Methods 38
4. Summary of Theoretical Performance 42
B. HARDWARE ISSUES 43
1. Configuration of Modem 43
a. In the ARIES Autonomous Underwater Vehicle 43
b. Topside Modem 46
2. Ancillary Equipment Used 47
a. DiveTracker™ 47
b. Sea-Bird CTD 47
c. Support Equipment 48
C. SOFTWARE DEVELOPMENT 49
1. Aries Computer Architecture 49

2. Changes to Execution Code....50
3. Development of Modem Code....52
a. The fm Process....53
b. Description of Command Types....54
c. Emergency Abort Command....56
D. CHARACTERIZATION OF TRANSMISSION DELAY TIMES....57
1. Experimental Procedure in Laboratory and Bay....57
2. Results....58
3. Conclusions....61
E. CHARACTERIZATION OF EFFECTIVE RANGE....63
1. Discussion....64
2. Short Range Response (Tranducer Depth Effects)....64
a. Results....65
b. Conclusions....67
3. Parallel Geometry (Constant Depth Channel)....68
a. Results....70
b. Conclusions....72
4. Perpendicular Geometry (Converging Channel)....73
a. Results....74
b. Conclusions....76
F. VEHICLE BEHAVIOR CONTROL USING MODEM....79
1. Discussion....79
2. Test Results....80
3. Conclusions....85
G. ACOUSTIC CONTROL CONCLUSIONS....87

IV. ACOUSTIC DATA TRANSFER....89
A. INTRODUCTION/BACKGROUND....89
B. INTEGRATION INTO THE ARIES AUV....89
1. Benthos Modem Operating Parameters....90
a. Basic Specifications....90
b. Encoding Schemes....91
c. Modulation Methods....92
2. Hardware Installation....93
3. Software Development....96
C. CHARACTERIZATION OF LINK RANGE AND TRANSFER RATES....96
1. Parallel Geometry (Constant Depth Channel)....97
a. Results....98
b. Conclusions....99
2. Diverging (Increasing Depth) Channel Geometry....100
a. Results....102
b. Conclusions....103
3. Converging (Decreasing Depth) Channel Geometry....104
a. Results....106
b. Conclusions....107
4. Transducer Altitude Effects at Short Range....108

a. Results *109*
b. Conclusions *111*
D. CONCLUSIONS **112**

V. COOPERATIVE AUV RENDEZVOUS AND TRACKING SIMULATION **115**
A. DISCUSSION **115**
B. SIMULATION CODE DEVELOPMENT **116**
1. Mission Scenario **117**
2. Highlighted Features of Code **117**
3. Simulation Output **119**
C. SAMPLE RESULTS AND OBSERVATIONS **119**

VI. SUMMARY AND CONCLUSIONS **127**
A. CONTRIBUTIONS **127**
1. Characterized Acoustic Modem Performance on a Dynamic, Operational AUV **127**
2. Demonstrated Acoustic Control of AUV in Adverse Shallow Water Environment **128**
3. Demonstrated Limits on High Speed Asymmetric Data Transfer **129**
4. Developed Simulation Demonstrating Multi-Vehicle Cooperative Tracking Behavior for High Speed Acoustic Data Transfer **129**
B. FURTHER WORK **130**

APPENDIX A **131**

APPENDIX B **133**

APPENDIX C **135**

APPENDIX D **137**

APPENDIX E **141**

APPENDIX F **147**

APPENDIX G **149**

APPENDIX H **151**

LIST OF REFERENCES **169**

INITIAL DISTRIBUTION LIST **175**

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1. Acoustic Modem Block Diagram (After refs [40] and [41]) 10
Figure 2. Binary (2,1,2) Convolutional Encoder (After ref [43]).....17
Figure 3. Trellis for Binary (2,1,2) Convolutional Code.....19
Figure 4. Encoding of Input Sequence (1 0 1 0 1) to (1 1, 1 0 , 0 0, 1 0, 0 0, 1 0, 1 1)...20
Figure 5. Viterbi Decoding Using Hamming Metric 22
Figure 6. Binary FSK, Two Signal Frequencies (After ref [46]) 25
Figure 7. 4-ary FSK, Four Signal Frequencies (After ref [46]) 26
Figure 8. Phase Shift Key Mapping (BPSK and QPSK).....27
Figure 9. Typical Frame or Packet Structure 30
Figure 10. Acoustic Absorption as a Function of Frequency (From ref [56]) 32
Figure 11. Propagation Ray Trace, Very Shallow Water (From ref [57]) 33
Figure 12. Control Communications Flow.....35
Figure 13. Autonomous Control System Block Diagram 36
Figure 14. Mode 4 Modulation, FAU Modem 39
Figure 15. Mode 3 Modulation, FAU Modem 40
Figure 16. Mode 2 Modulation, FAU Modem 41
Figure 17. Mode 1 Modulation, FAU Modem 42
Figure 18. ARIES Hardware Schematic (From ref [36]) 44
Figure 19. Installed FAU Modem Board Set: Power Supply/Amp (cylindrical top) and DSP Board (rectangular bottom).....45
Figure 20. FAU Modem Transducer Mounted on ARIES Fairing.....46
Figure 21. FAU "Topside" Modem Configuration (from [61]) 47
Figure 22. Typical Operating Configuration.....48
Figure 23. Dual Computer Software Architecture (SM is Shared Memory) 50
Figure 24. Pseudo Control Code for Modem Process *fm* 53
Figure 25. Time Delay, Mode 4 Dual Code 59
Figure 26. Time Delay, Mode 4 BCH Block Code 59
Figure 27. Time Delay, Mode 3 Dual Code 60
Figure 28. Time Delay, Mode 3 BCH Block Code 60
Figure 29. Message Success Data (Altitude Effects) 65
Figure 30. Message Success Rate (Altitude Effects) 66
Figure 31. Message Success Data (Configuration Effects).....66
Figure 32. Message Success Rate (Configuration Effects) 67
Figure 33. Representative Monterey Bay Sound Velocity Profile.....70
Figure 34. Message Success Data (500 Meter, Constant Depth Channel).....71
Figure 35. Message Success Data (750 Meter, Constant Depth Channel).....71
Figure 36. Message Success Rate (All Ranges, Constant Depth Channel).....72
Figure 37. Message Success Data (500 Meter, Converging Channel).....75
Figure 38. Message Success Data (750 Meter, Converging Channel).....75
Figure 39. Message Success Data (1000 Meter, Converging Channel).....76
Figure 40. Message Success Rate (All Ranges, Converging Channel).....76
Figure 41. Depth Control, No EMI Shielding, Mission Abort.....81
Figure 42. Depth Control, No EMI Shielding, Point of Mission Abort 82

Figure 43. Depth & Altitude Control Mission, After Shielding.....82
Figure 44. Zoom View of Depth Change After Shielding 83
Figure 45. Representative Depth Control Run Using Acoustic Commands 84
Figure 46. Representative Depth-to-Altitude Control Run Using Acoustic Commands ..85
Figure 47. MFSK Modulation, Benthos Modem 93
Figure 48. Benthos 89X Modem Board Set 94
Figure 49. Benthos Modem Transducer Mounted on ARIES 94
Figure 50. Benthos Deck Box Modem.....95
Figure 51. Benthos LF Transducer and 25 Meter Cable 96
Figure 52. Parallel (Constant Depth) Channel Geometry 98
Figure 53. Message Success Rate (Constant Depth Channel).....99
Figure 54. Diverging (Increasing Depth) Channel Geometry.....102
Figure 55. Message Success Rate (Increasing Depth Channel) 103
Figure 56. Converging (Decreasing Depth) Channel Geometry.....106
Figure 57. Message Success Rate (Decreasing Depth Channel).....107
Figure 58. Short Range Altitude Effects Test Diagram 109
Figure 59. High Data Rate: Altitude Effects 110
Figure 60. High Data Rate: Baud Rate Success 110
Figure 61. High Data Rate: Altitude Effects and Baud Rate Success 111
Figure 62. ARIES Track, 17 Second "Request to Rendezvous".....120
Figure 63. ARIES Track, 80 Second "Request to Rendezvous" 121
Figure 64. ARIES Speed Performance, 17 Second "Request to Rendezvous" 122
Figure 65. ARIES Speed Performance, 80 Second "Request to Rendezvous" 122
Figure 66. Cooperative Tracking Position Error, 17 Second "Request to Rendezvous" 124
Figure 67. Cooperative Tracking Position Error, 80 Second "Request to Rendezvous" 124

LIST OF TABLES

Table 1. Binary Block Code, $k = 3$ and $n = 6$ 13
Table 2. Theoretical FAU Modem Performance (from [61]) 43
Table 3. Summary of Modem Delay Time Characterization 63
Table 4. Summary of Modem Altitude Effects Characterization 68
Table 5. Parallel (Constant Depth) Channel Range Limit Characterization 73
Table 6. Perpendicular (Converging) Channel Range Limit Characterization 78
Table 7. Parallel (Constant Depth) Channel Data Transfer Rate Summary 100
Table 8. Diverging (Increasing Depth) Channel Data Transfer Rate Summary 104
Table 9. Converging (Decreasing Depth) Channel Data Transfer Rate Summary 108
Table 10. Altitude Effects and High Speed Data Transfer Rate Summary 112

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

Thank you to the many people who contributed to my success in this endeavor, including the members of my Doctoral Committee for their knowledge and guidance and the Naval Officers and Staff members that assisted throughout the countless hours of experimental investigations.

My deepest gratitude and thanks to Professor A.J. "Tony" Healey: boss, mentor, and friend. Throughout more than a dozen years of close professional and personal contact, he has been the source of constant guidance and support in my quest for higher education. He will always be my definition of the consummate Professor and gentleman, my role model to emulate as I resume my position in academia.

Last, and certainly not least, I dedicate this work to my family. To Melissa and Bill, Alycia and Shanoa, and my wonderful wife Liz: thank you for your constant love and support. Family always has been and always will be my bedrock, the vital keel to my ship during life's journey.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Advances in command and control of Autonomous Underwater Vehicles (AUVs) using acoustic communications are crucial to future Fleet objectives, particularly in Very Shallow Water Mine Countermeasures (VSW MCM). Understanding of the capability to redirect missions, provide relatively high rate downloads of mission information, and perform cooperative tracking for multi-vehicle systems is critical to future operations. Unfortunately, our present understanding is limited to some bounding data based on fixed node experiments while the impact of vehicles working in the real environment is not well understood.

Work in untethered UUV acoustic communication and control spans only the last few decades. While substantial progress has been made in deep water with vertical acoustic channels, shallow water acoustic communications have been far less successful, primarily due to signal interference, larger background noise, and the multiple reflections encountered between bottom and surface bounces.

The main objectives of this dissertation were to investigate and demonstrate the capabilities of tactical acoustic control of a dynamic, operational underwater vehicle in the Very Shallow Water (VSW) ocean environment. This necessarily required studies on the limitations of Acoustic Control and relatively High Data Rate Transfer when using commercial acoustic modems in underwater vehicles and investigation of their acoustic transmission characteristics. Comprehensive empirical evidence through field evaluation with the ARIES vehicle indicated that reduced ranges were required for successful acoustic communications in a realistic very shallow water environment. A simulation was developed to demonstrate a solution for reducing the range and conducting multi-vehicle behaviors for cooperative tracking and acoustic communications data transfer.

The key result of these studies indicated one inescapable conclusion; Autonomous Underwater Vehicles operating in realistic, adverse Very Shallow Water (VSW) environments are severely limited in the use of acoustic communication for either control or data transfer. Range and baud rates for successful and reliable communication are degraded by multipath reflections, temporal and spatial variations in the acoustic channel properties, and transient noise sources endemic to the specific operating area. The

results of these studies suggests the possible need for a paradigm shift in the method used for AUV Shallow Water acoustic communication, particularly if high speed data transfer is the desired goal. Rather than continuing to design increasingly complex hardware to overcome signal distortion and other difficulties introduced by and inherent to the environment, perhaps the answer may lie in actively decreasing the distance between the autonomous platforms through multi-vehicle cooperative control.

Specific contributions from this body of work include:

- Characterized Acoustic Modem Performance on a Dynamic, Operational AUV: Based on a review of the open literature, a comprehensive study on the bounds of AUV acoustic communications has not previously been done, particularly in a realistic, shallow water environment that closely models a mine warfare scenario. Therefore, two commercially available modem systems were installed in the NPS ARIES AUV (an experimental AUV) and the acoustic transmission performance of each was thoroughly investigated.

Four low speed modulation and coding configurations of a Florida Atlantic University (FAU) modem system were evaluated in various channel geometries. The focus of the FAU studies was to successfully demonstrate tactical control of the ARIES, through both one-way and two-way acoustic communication. Additionally, these studies also concluded that the minimum one-way transmission time, neglecting the channel delay due to sound velocity, was three seconds and there was a clear trend toward improved communication performance when the AUV operated near the sea floor.

For a Benthos modem system, eight configurations with baud rates from 150-5120 bits/second were evaluated in various channel geometries. The primary focus of the Benthos studies was to examine the limits of relatively high rate acoustic data transfer. A secondary goal in these investigations was the demonstration of acoustic control at higher baud rates and a confirmation of the trend toward more reliable communication when the vehicle operated at lower altitudes in the water column. All of the experimental studies were conducted in 15 meters of water depth in Monterey Bay, California.

- Demonstrated Acoustic Control of an AUV in an Adverse Shallow Water Environment: AUV computer software was developed and compiled into the ARIES operating system to successfully alter missions and monitor vehicle status. Tactical

acoustic control in the shallow water environment using baud rates from 55-220 bits/second with the FAU modem system was demonstrated. One-way control modes and operating set points were acoustically sent to, received by, and acted on by the ARIES. Two way queries sent by the topside tactical controller modem were responded to by the AUV. The maximum operating range for effective acoustic control of a dynamic AUV with all integrated systems active operating in about 15 meters water depth in open ocean was around 500 meters in the horizontal plane with ~85% reliability, at bit rates up to a maximum of 220 bits/second.

- **Demonstrated Limits of Acoustic High Speed Asymmetric Data Transfer:** In the most favorable channel geometries, the top combinations of range and rate observed for high speed data transfer were about 440 meters at 800 bits/second (~100% reliability) and under 250 meters at 1200 bits/second (about 90% reliability). The coherently modulated baud rates were unsuccessful in the very shallow water environment. Even at ranges under 100 meters, the 2560 bits/second rate was less than 25% reliable and no replies were received when using the 5120 bits/second baud rate. The maximum range where any communication was possible with even the most robust data transfer rate (150 bits/second) was 685 meters. Assuming uncertain vehicle orientation and channel geometry during acoustic communication, the maximum operating range for effective high speed data transfer in shallow water from a dynamic AUV with all integrated systems active was about 300 meters in the horizontal plane with nearly 100% reliability, at rates up to a maximum of 800 bits/second.

It is apparent that significant limitations to realistic operating communication range exist in the open ocean very shallow water. It is likely that the existence of multiple surface and bottom reflections of transmitted acoustic paths cause destructive interference to the point where at ranges of 20-30 water depths, signal decoding becomes very difficult. Even with noncoherent (MFSK) modulations, data cannot be reliably recovered from a moving vehicle beyond these ranges. High speed coherent (PSK) modulated signals could only be decoded at extremely short range, < 50 meters. With vehicle borne units, power is naturally limited, so simply increasing transmission power is not a viable option. This work indicates that a more comprehensive characterization

and understanding of the very shallow water acoustic propagation paths and attenuation would help but was beyond the scope of the dissertation.

- **Developed Simulation Demonstrating Multi-Vehicle Cooperative Tracking Behavior for High Speed Acoustic Data Transfer:** The simulation was developed to demonstrate the concept of multi-vehicle rendezvous and cooperative tracking for acoustic data transfer with the ultimate goal of transmitting the data collected to Warfare Commanders. A "server" made rendezvous with a "searcher" AUV, paralleled the track to acoustically receive information on the order of hundreds of kilobits, then returned to its original loiter area to transmit the data collected.

Two opportunities for continued research clearly obvious from the studies in this dissertation include the need for further development of multi-vehicle rendezvous and cooperative tracking control to facilitate relatively high speed acoustic data transfer in the VSW and shallow water environment and development of a high speed modem in MHz region to take advantage of the short ranges anticipated during cooperative tracking.

I. INTRODUCTION

A. BACKGROUND

Advances in command and control of Autonomous Underwater Vehicles (AUVs) using acoustic communications are crucial to future Fleet objectives, particularly in Very Shallow Water Mine Countermeasures (VSW MCM). Understanding of the capability to redirect missions, provide relatively high rate downloads of mission information, and perform cooperative tracking for multi-vehicle systems is currently limited to some bounding data based on fixed node experiments while the impact of vehicle motion is not well understood.

The need for continuing research and improvements in our capabilities in the underwater environment is widely recognized as a key factor in our national security posture. A distinguished committee of experts completed a comprehensive study on facets of our undersea science and technology program and found many shortcomings in our current national progress [1]. Their report cites underwater acoustic communication and multi-vehicle networks of Unmanned Underwater Vehicles (UUVs) as two technology areas ripe for improvement or development. In addition to serving as data transporters, they also envision future UUVs as possible hunter/killer weapons to augment or replace torpedos in the littoral areas.

The Navy has long recognized that it must continue to lead in the development of the defense technology vital to maintaining our superiority in the underwater environment. The UUV Master Plan [2] provides the long range vision and establishes a coordinated, coherent direction for future research in critical underwater vehicle areas. Concurrently, the Navy has designated Autonomous Operations (AO) as one of its twelve most important Future Naval Capabilities (FNC) with the goal of transitioning concepts into hardware for the Fleet [3]. While a broad range of underwater topics was discussed in each document, two recurring areas highlighted for further study were underwater communications and multi-vehicle cooperative behavior and control.

Specifically, advances in the area of underwater communications will effect the first three priority signature capabilities defined by the Navy UUV Master Plan: Maritime

Reconnaisance, Undersea Search and Survey, and Communication/Navigation Aids. High quality, high data rate communication systems will be needed to amass the near real-time data collected from a variety of platforms including the undersea nodes of the Net-centric Warfare grid.

Similarly, further research in vehicle control and multi-vehicle cooperative behavior is necessary to achieve the first and second priority signature capabilities. Each signature capability necessarily involves multiple platforms. The ability to have a means of communication and control when more than one vehicle is in an operating area is vital, particularly if it becomes necessary to redirect the mission as the tactical situation changes or to collect and relay in-situ operational data to the warfare commanders.

B. DISCUSSION

Work in untethered UUV acoustic communication and control spans only the last few decades and the greatest successes have been achieved primarily in deep water and vertical acoustic channels. Many modem systems may be physically capable of much greater rates when tested in a static environment and under controlled conditions. Static deep water tests have been published with noncoherent rates up to 2500 bits per second (bps) at a 3.7 kilometer range and coherent rates as high as 30 Kbps at 3.5 kilometers. Similarly, structured testing has generated reported rates from 1200 bps (noncoherent) at a range of 3.0 kilometers to 20 Kbps (coherent) at 900 meters in shallow water. The maximum range reported for any modem was 50 kilometers, in deep water using coherent modulation at a rate of 200 bps [4]. However, the focus in this discussion is to review the documented performance of those systems actually integrated into an operational UUV.

The Advanced Unmanned Search System (AUSS) began a twenty year development program in the 70's by the predecessor to the Space and Naval Warfare Systems Center (SSC) San Diego. This deep ocean vehicle, operationally tested in 1992, was supervised by humans through a half-duplex acoustic communications link [5]. Using a phase shift keying modulation method, commands were transmitted and received through a baffled transducer (EARS Towfish) trailed aft of the support ship at depths up to 300 feet [6]. Compressed images at rates of 2400 bits per second (bps) were achieved in the relatively benign vertical channel from a depth of 12,000 feet [7]. Although

currently inactive, the AUSS remains in a standby status for future missions and has received an updated vertical telemetry system reported to be capable of data rates up to 4800 bps and a vertical range of 10 kilometers [8]. Theseus, a large AUV developed by a Canadian team (International Submarine Engineering Research and the Esquimalt Defence Research Detachment) to lay fiber optic cable in deep channels under the ice, also used acoustic control. However, the primary control exercised was the passing of heading error corrections from stationary acoustic beacon buoys. On the outbound leg of each cablelaying mission, Theseus used the fiber optic cable itself for communications. On the return leg, it used acoustic beacons for heading corrections. In earlier Arctic field trials, it had used a nominal 50 bps system to receive commands at ranges from about 3-8 kilometers in unspecified "shallow" and "deeper" conditions [9][10]. The Defense Advanced Research Projects Agency (DARPA) sponsored the Advanced Minehunting and Mapping Technology (AMMT) program in the 90's to demonstrate and validate advanced UUV capabilities. The integrating contractor, Draper Laboratories, retrofitted one of their existing UUVs with state of the art subsystems from various participants in academia and industry [11]. The modem selected, provided by Woods Hole Oceanographic Institution (WHOI), used quadrature phase shift keying modulation. Eight receivers with 45 degree beam patterns were mounted on the forward free-flooded section of the 40 foot long vehicle. Two projectors or transmitters with a 30 degree beam pattern were mounted flush to the rear free-flooded section. When the proper transducer geometry existed between the four element receiver on the ship and one of the two projectors on the UUV during deep water testing in depths up to 180 meters, most vehicle status updates were received in the range of 1-2 kilometers at data rates up to 5 kbps. However, return communication to the UUV from the ship was not demonstrated and attributed to poor thermal layer conditions and improper geometries between the ship transmitter and UUV receiver patterns. Additionally, an image was successfully transmitted acoustically with the modem system but in a static post-mission environment [12][13]. The Norwegian UUV Hugin, designed for deep seabed survey and operational since 1996, has multiple acoustic systems. Its acoustic command link provides two way communications at a low bit rate (around 55 bps) using frequency shift keying

modulation. The data link, with bit rates up to 2000 bps, is one way (from the vehicle to the surface) and uses multiple frequency shift keying. The Hugin's tertiary acoustic link, operating in the same frequency band as the command link, uses the acoustic positioning transducer as an emergency command link. The elements on the spherical transducer are normally used strictly for positioning the UUV relative to the ship, but a separate menu is available on the operating console when an emergency communications link is required [14]. Finally, the Sirene, a European vehicle intended to shuttle benthic stations to depths of 4000 meters, reports the capability of a "low data rate" acoustic communication link between the UUV and the support ship [15].

Acoustic control of UUVs in the more adverse shallow water environment has increased in importance with the Navy's recent focus on the littorals and the mine warfare problem. Although communications theory will be discussed in greater detail later in this work, reliable and efficient shallow water acoustic communications are much more difficult to achieve than those in a vertical channel, primarily due to signal interference and reflections. An additional restriction is that littoral UUVs tend to be much smaller in size (under 500 kilograms) and, therefore, suffer the further disadvantages of less space and power to house and operate the communications equipment. However, many of the shallow water and very shallow water UUV platforms either have or intend to integrate an acoustic modem capability to their vehicle.

While details were not available in the literature, the Florida Atlantic University has claimed successful demonstration of in-flight retargeting using their Ocean Explorer (OEX) AUV, the predecessor to their Morpheus vehicle [16]. In an earlier test, OEX sent one way telemetry transmissions in the shallow water channel to a surface ship using multiple frequency shift keying at a 20 bps data rate [17]. The Marine Systems Engineering Laboratory, using two open framed EAVE III vehicles, sent a preprogrammed 1024 byte transmission over a hundred meter range at 600 bps [18]. The REMUS, a WHOI UUV currently in use by the Fleet, does not have a modem installed as standard equipment. However, there are plans to integrate one in both the vehicle and in the PARADIGM tracking buoy system [19]. In a previous exercise demonstration, REMUS was temporarily outfitted with a developmental modem and conducted one way

communications from the AUV at a 58 bps data rate using binary phase shift keying modulation. The modems demonstrated much higher bit rates (up to 5 kbps), but only in controlled tests from surfaced ship-to-ship platforms [20]. In ongoing shallow water experiments, WHOI modems have been used with limited success with a surf zone crawling UUV, demonstrating a tenuous low data rate communications link [21], but performance details are sketchy. Cetus, a shallow water search UUV, has not yet integrated acoustic modems for command but has demonstrated fiber optic tether control [22]. The Odyssey class AUVs, a product of the MIT Sea Grant program, successfully demonstrated one way control (heading commands to the vehicle only) using a spare acoustic channel on its long baseline navigation system [23]. A military derivative of the Odyssey, the Battlespace Preparation AUV (BPAUV), did not include any acoustic control equipment in its initial configuration for operational trials [24]. The Naval Coastal Systems Station (CSS) crawler, a surf zone UUV, has demonstrated data rates up to 1200 bps in experiments, passing compressed imagery back to the operator. Initial acoustic testing with rates up to 600 bps has also been completed with the small EMATT AUV by Sippican, Inc., envisioned as a data transport node for future underwater networks [25].

Having briefly reviewed the history of acoustic communication and control of operational UUVs, it is important to note that acoustic communication became the method of choice after careful consideration of the alternative methods available. Despite obstacles such as low bandwidth, time and frequency spreading, and long delay times inherent to the environment, sound transmission is still the best match for this adverse channel. Opaque and dense, the sea more readily absorbs and substantially weakens other forms of energy transfer such as light or electromagnetic (RF) waves [26]. For example, while laser beams (particularly blue-green) have the advantages of higher data rates with less interference, they have the distinct disadvantages of requiring clear water, accurate alignment, and short transmission distances [27][28]. Therefore, acknowledging that it is and will remain the key communications method for future undersea dominance, the Navy has given full support to and devoted massive resources

towards advances and improvement in all facets of underwater acoustics [29]. New modems are under development at commercial activities and educational institutions such as Benthos, Incorporated and WHOI.

The amount of data accumulated for analysis during a mission scales dramatically as the number and quality of the sensors increase. The UUV must be able to ultimately transfer the critical data collected by its sensors to the evaluators and end-users in a timely manner to be effective. Therefore, the urgency of the transfer or latency of the data is very dependent on the mission or intended use of the information. For example, mine warfare UUV platforms used in the shallow and very shallow water regions (e.g. REMUS and BPAUV) currently require vehicle recovery before post-processing the data. However, this conflicts with the warfare commander's immediate need for the information they can provide. Hence, near real-time data transfer to a server vehicle or other data collection and dissemination point is one major motivation for continuing research work in higher speed, shallow water acoustic communications. In the interim, other methods for data transfer are being considered. Underwater docking platforms, intended to recharge power supplies as well as transfer data, are in various stages of development. Methods to download data to the dock (for further transfer by RF or satellite) have included optical transfer, inductive coupling, and direct electrical connection methods [30][31]. If receipt of the data is not time critical (e.g. long term environmental monitoring or scientific exploration) periodic updates using satellite communications similar to methods used by the UUVs such as ALTEX or the glider vehicles (tritium satellite telephone modems) may be an acceptable alternative [32]-[35].

While not yet demonstrated by dynamic UUV platforms, advances in acoustic control combined with enhanced methods of acoustic high speed data transfer will naturally evolve into the next crucial capability desired in the battlespace: multiple vehicle cooperative behavior. The warfare commander would gain two immediate tactical advantages; the ability to redirect undersea missions as priorites change and a means to collect near real-time uploads of the critical battlespace data. "Search" UUVs such as the REMUS could be sent new mission or track plans to alter their search areas or patterns acoustically by "Master" vehicles. Data server vehicles with both RF and

acoustic communications (e.g. ARIES [36]) could relay new commands from surfaced or submerged sources and perform an "on demand" rendezvous to upload time critical side-scan files or pictures from working vehicles. Initial steps toward successful integration of multi-vehicle systems into the battle space are ongoing (e.g. [25]) but much work remains.

C. SCOPE OF WORK

The objectives of the work described in this dissertation are efforts to:

- Investigate the acoustic transmission characteristics of commercially available modems installed on ARIES.

- Investigate and demonstrate the capabilities of acoustic control of a dynamic, operational underwater vehicle in the Very Shallow Water (VSW) and Shallow Water ocean environment. Given the low data rate link expected, semi-autonomous control is desirable.

- Study the limitations of Acoustic Control and relatively High Data Rate Transfer when using commercial acoustic modems in underwater vehicles.

- Study methods by which short range communications could be utilized in data transfer between multiple vehicles and simulate multi-vehicle behaviors for cooperative tracking while conducting acoustic communications data transfer.

Extensive laboratory and field testing were required to accomplish the majority of these objectives. This work required a "systems engineering" approach with a synthesis of knowledge from muliple fields including mechanical engineering, acoustics, and computer science.Section I has discussed the previously reported work in underwater acoustic communications and control as found in the open literature , highlighting areas of success as well as those areas that have not reached full maturity. Also discussed is the scope of work which is intended to examine a small portion of some of the less evolved areas.

Section II is an explanatory chapter on the basic theory of acoustic underwater communications.

Section III presents the majority of the in-water experimental testing completed in order to characterize modem performance as installed in the ARIES vehicle and to demonstrate the use of acoustic communications to control an operational AUV. The section begins with a chapter describing the particular coding schemes used in the modem. The next two chapters are devoted to hardware and software issues. The hardware chapter includes a discussion of the modem components and the support equipment. The software chapter reviews the current software architecture in the ARIES and ends with the code developments that were required to be implemented to meet research objectives. The next three chapters of this section contain the results of experimental investigations of modem performance and vehicle reponse to modem commands. The final chapter summarizes the results of the acoustic control studies.

Section IV discusses the integration and installation of a high speed, high data rate acoustic modem into the ARIES. Extensive experimental evaluation of its asymmetric data transfer download performance is completed and analyzed.

Section V contains a simulation developed to demonstrate the envisioned use of acoustic communications for both high rate data transfers and multi-vehicle cooperative tracking.

Section VI contains the summary and conclusions to this study. Samples of pertinent code developed and the complete Matlab version of the cooperative vehicle tracking and data transfer simulation are contained in the Appendices.

II. COMMUNICATIONS THEORY REVIEW

Underwater acoustics (UWA) and acoustic communications (acomms) are broad and deep fields of study, subdisciplines that combine knowledge from numerous sources. For example, to fully comprehend underwater acoustic communications, one requires an understanding of elements of physics (acoustics), oceanography (environment), communications (digital signal processing), and materials science (tranducer design). Countless volumes have been written on each subject, including many excellent reference texts on the physics and communications principles involved (see, for example, [26],[37]-[40]). While active research is projected to continue in this field for the foreseeable future, the summary of efforts to date in [4] must be mentioned as important background reading. This section makes no attempt to comprehensively incorporate or encapsulate the entire field of underwater communications. Rather, the focus will be restricted to a discussion of the theory behind and terminology applicable to typical underwater modem systems as potential systems for tactical UUV control. Included will be an explanation of the elements of an underwater communications system, the stages of the communication process, the types of modulation or coding encountered, and environmental considerations. In particular, this chapter supports the discussion of detailed experimental studies described later.

A. GENERAL UNDERWATER COMMUNICATIONS SYSTEM

Communication, in its most general sense, involves an information source transmitting some type of message or signal through a medium (possibly corrupted by noise) to the ultimate recipient. Information sources can be analog (continuous) such as the voice when one speaks or digital (discrete) such as when a Morse Code signal is tapped out on a telegraph line. Transmission of the signal may require conditioning processes to add redundancy or special formatting, depending on the type of medium or channel that must be used. The medium or channel is the path that the communication signal will take between the transmitter and the receiver. Examples of communications channels range from the simple, such as the wire for standard telephones, to the complex, such as the atmosphere for wireless cell phones or the ocean for acoustic modems. While transiting the channel, the signal is susceptible to corruption by noise sources such as,

amongst others, static electricity, mulitipath reflections, biological noise and fading. Finally, the intended receiver must be able to intercept the message or signal sent and possibly perform inverse conditioning processes on it to recover the original message.

Underwater Communications systems, the focus of this discussion, rely almost exclusively on digital technology although there were some early analog systems [4]. Typical digital acoustic communications systems contain most, if not all, of the basic elements in Figure 1. (Adapted from general communications diagrams in [40] and [41]).

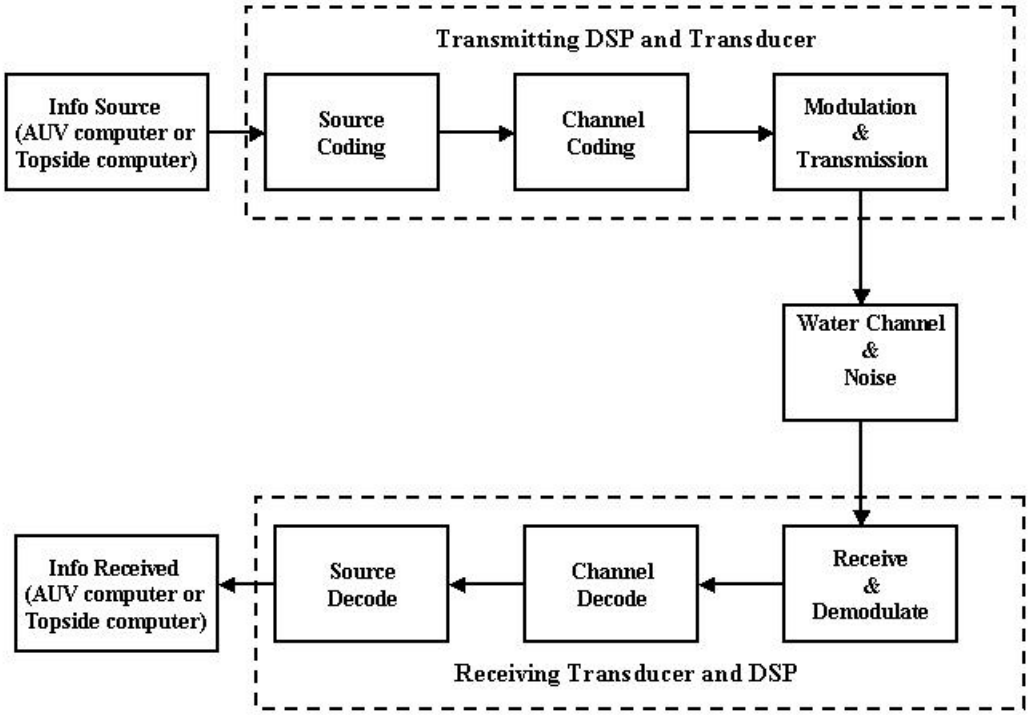


Figure 1. Acoustic Modem Block Diagram (After refs [40] and [41])

Before discussing key elements of the acoustic modem system in some depth, it is important to clarify the meaning of the term "coding" as used throughout communications literature. Coding, according to Webster's Dictionary, is "*to put in or into the form or symbols of a code*". Within the communications literature, coding is often used to generically describe the process that takes place within the Digital Signal

Processor (DSP). In reality, multiple distinct processes are completed in the DSP including source coding, channel coding and modulation. Fundamental to information theory, only the first two are strictly coding processes.

While the ultimate goal is for autonomous vehicles to independently use their onboard computers as both the source and the receiver of information and action messages, many current typical underwater communication systems involve a human operator on at least one end of the communication sequence. The operator has the ability to query or task the AUV via a "topside" computer. As the following discussion of the modem system components progresses, it is assumed that the topside computer is manned by a human operator.

B. INFORMATION SOURCE AND SOURCE ENCODING

The information source originates with an analog thought posed by the human operator. This query or tasking is typed (in a specified format) onto the topside computer where the typewritten ASCII characters now become digitally stored by the computer as binary characters. Transfer of this digital information to the modem can be either by direct serial port link (RS232) or by other means such as a wireless RF modem.

Source encoding is a means to convert information to the most efficient binary representation possible. All modern underwater communications systems transmit information digitally. Therefore, whether the initial source is digital or analog, source encoding generally must be done to ensure the proper binary format. The final result of source encoding is an alphabet of codewords understood by both the source and the ultimate receiver. The source encoding branch of information theory is heavily enmeshed in probability theory. Numerous mathematical models (based on the degree of statistical dependence) and algorithms (Huffman, Lempel-Ziv, etc.) have been developed [40]. In each case, the goal of source encoding remains the same: to develop the most efficient, unique, unambiguous, and instantaneously decodeable table of binary symbols that will retain and convey all of the information content sent by the source.

C. CHANNEL ERROR CORRECTION CODING

Once the source information has been converted into binary digits (bits), the next major issue is to prepare the data for reliable transmission over a noisy channel. This

process is known as error correction coding. Many variations on error detection and correction methods are possible, but the fundamental purpose of any channel encoding scheme is to add redundancy to the digital bits of source data. With added redundancy, the decoding device at the receiver will then be able to detect and correct errors without the need for retransmission or other complications. Many terrestial coding schemes have been used only to detect rather than detect and correct errors [42]. But the nature of underwater communications, including the propagation delays encountered and the real-time requirement, have demonstrated the need for detection and correction, even at the added cost of bandwidth to accommodate redundant digits. The key reason for adding redundancy is that it broadens the field of messages possible while only a limited number in the field are legitimate. Therefore, receipt of an error is readily determined. For example, a encoding a string of two data bits at a ½ code rate (explained later) will result in a four symbol string. The two data bit strings have four possible sequences but the four independent symbols have 16 possible sequences. Therefore, there are twelve sequences (16-4) that would indicate reception of an incorrect message. As the number of redundant digits increases, the ratio of possible sequence combinations to legitimate sequences rises exponentially, increasing the ability to detect signal corruption.

In 1949, Shannon discovered that it was theoretically possible to drive transmission errors to arbitrarily low levels in any channel if the proper encoding was selected [41]. Consequently, there has been much work in the past 50 years to develop practical codes that approach this theoretical ideal. There are two main categories of channel encoding schemes: block and convolutional. While coverage here of this voluminous field will necessarily be brief, an in-depth explanation of channel encoding and decoding methods and theory may be found in many reference texts, such as [40] and [43]. The theoretical study of coding methods presumes familiarity with linear algebra, binary arithmetic, polynomial manipulation, and matrix operations.

1. Block Coding

Block coding derives its name from the fact that the information is divided into blocks containing k information bits, called a "message". These k bits are then transformed into a symbol containing n digits, where $n>k$. The transformed symbol is

known as a "code word." The number of distinct messages possible, or the size of the "alphabet" or "dictionary", is $M = 2^k$. Since they have direct one-to-one correspondence with each message, there are 2^k code words and this set of code words is called an "(n,k) block code." A metric used to determine the information content transmitted per symbol is called the "code rate" (R) and is defined by $R = k/n$. In essence, block coding uses a combination logic circuit to add (n-k) digits of redundancy to each message in an effort to overcome channel noise. Table 1 is an example of a (6,3) binary block code. Each 3 (k) bit message has been encoded into a 6 (n) bit code word. The size of the alphabet or dictionary (M) is 2^3 hence there are 8 code words possible. The code rate, R, is ½ (3/6). Each message has 3 additional digits of redundancy to use when combating the channel noise problem.

Messages	**Code Words**
(0 0 0)	**(0 0 0 0 0 0)**
(1 0 0)	**(0 0 1 1 0 0)**
(0 1 0)	**(0 1 0 0 1 0)**
(1 1 0)	**(0 1 1 1 1 0)**
(0 0 1)	**(1 0 0 0 0 1)**
(1 0 1)	**(1 0 1 1 0 1)**
(0 1 1)	**(1 1 0 0 1 1)**
(1 1 1)	**(1 1 1 1 1 1)**

Table 1. Binary Block Code, $k = 3$ and $n = 6$

Block coding has numerous subclassifications. Widely used are the linear, cyclic, BCH (Bose, Chaudhuri, and Hocquenghem), and Reed-Solomon codes.

a. Linear Block Codes

The basic linear block code produces its codewords through matrix multiplication. Each input message (X) generates a codeword (Y) by multiplication with a generating matrix, G, whose k rows are linearly independent basis vectors of length n. The input is an M by k matrix, G is a k by n matrix, and the output is an M by n matrix.

$$X = \begin{pmatrix} 0\,0\,0 \\ 1\,0\,0 \\ 0\,1\,0 \\ 1\,1\,0 \\ 0\,0\,1 \\ 1\,0\,1 \\ 0\,1\,1 \\ 1\,1\,1 \end{pmatrix} \qquad G = \begin{pmatrix} 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad Y = X * G \quad \Rightarrow \quad Y = \begin{pmatrix} 0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,1\,0\,0 \\ 0\,1\,0\,0\,1\,0 \\ 0\,1\,1\,1\,1\,0 \\ 1\,0\,0\,0\,0\,1 \\ 1\,0\,1\,1\,0\,1 \\ 1\,1\,0\,0\,1\,1 \\ 1\,1\,1\,1\,1\,1 \end{pmatrix}$$

Mathematically, $X * G = Y$. The matrices used to generated the codewords in Table 1 are shown in the equation above:

An (n-k) by n parity check matrix, H, is created from a mathematical derivative of the G matrix. The parity check matrix for the example above is:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}$$

Once the codeword has reached its destination, decoding is accomplished by multiplying the received codeword (R) with the transpose of the parity check matrix. If the product, called the syndrome (S), is identically the zero vector, the codeword has been received without error and the message is recoverable. Mathematically, $S = R * H^T$ where S is 1 by k, R is 1 by n, and H^T is n by (n-k). Continuing with the example above, assume the received codeword was R = [0 1 1 1 1 0]. When multiplied by the transpose of the parity check matrix, the syndrome is S = [0 0 0]. If the syndrome was equal to anything other than the zero vector, comparison with the columns of the parity matrix would make it possible to determine which digit of the received codeword was in error. One fundamental disadvantage of this basic linear block code is that it can correct only a limited number of errors based on Hamming weights and minimum distances. Multiple errors can be detected, but not necessarily corrected.

b. Cyclic Codes

Cyclic codes are a special category of linear code. The distinguishing feature is that cyclic codes have a more defined structure than basic linear block codes.

By definition, a code is cyclic if shifting one symbol in the n length codeword produces another n length codeword within the alphabet. For example, given $n = 4$, if [1 0 0 1] is a codeword and the code is cyclic, then [1 1 0 0] and [0 0 1 1] are also codewords. The mathematical complexity to generate and decode cylic codes exceeds that of the linear block code. Rather than a generating matrix, cyclic codes encode the message using a generating polynomial to represent each member of the alphabet. Codewords are created in a division circuit with feedback loops and the output is essentially a modified form of the message prepended with n-k parity check digits. Decoding is accomplished in a similar division circuit on the receiving end. The received signal is divided by the generating polynomial. The remainder, or syndrome, is used to determine errors and make error corrections by comparison with the legitimate codewords feasible in the alphabet. Easily implemented into logic circuit hardware, the main advantage of a cyclic code is that a greater number of errors can be corrected. However, simple cyclic decoding circuits tend to increase exponentially as the code length, n, grows. Therefore, further variations and improvements on cyclic coding schemes are desirable.

c. Advanced Cyclic Codes

The Bose, Chaudhuri, and Hocquenghem (BCH) and Reed-Solomon codes are two important classes of improved cyclic codes. The BCH code uses a generating polynomial of the lowest degree whose roots are powers of its primitive element. In simpler terms, this means that it is possible to generate all codewords in a set if the proper primitive polynomial is selected. The primary advantage of the BCH cyclic linear block code is that multiple errors can be corrected yet the decoding process is simplified due to generation of the code with the primitive polynomial. Decoding begins with computation of the syndrome from the received codeword, as in the case of other cyclic codes. The difference in BCH codes is that the error pattern is used to create a series of simultaneous equations. Solution of these equations by any algorithm resulting in the least amount of errors provides the solution for the most probable error pattern. Errors caused by transmission through the channel may then be corrected. Reed-Solomon codes are essentially an extension of the binary BCH code principles to nonbinary systems. For example, each code symbol of a 2^4-ary (15,11) Reed-Solomon code would represent 4 binary digits. The block length (n) is 15, the number of data symbols (k) is 11, the

number of parity check symbols (n-k) is 4, and the code would be able to correct 2 symbol errors per block. The primary advantage of the Reed-Solomon code is that it effectively corrects clustered errors produced by "burst" events found in the noisy underwater channel. One disadvantage of Reed-Solomon is that it requires an additional step in the decoding process to evaluate error values as well as determine the position where the error occured.

2. Convolutional Coding

Convolutional coding is the second main category of channel encoding schemes. The advantages of convolutional coding include its ability to perform well even in lower signal to noise (SNR) environments and the receiver does not require block synchronization for the decoder to operate correctly [44]. Although much of the terminology and many of the definitions used in block coding also pertain to convolutional coding, there are some major differences in the two methods. Rather than operate on a single block, convolutional coding operates on a sequence of blocks (or continuously, in the special case where the block size is one symbol long). Similarly to the block code, k bits are transformed into a symbol containing n digits, where $n>k$ and the transformed symbol is known as a "code word." Likewise, the same "code rate" metric ($R = k/n$) is used to determine the amount information content transmitted per symbol. However, the major difference between the two types is that convolutional coding schemes have a property termed "memory" which means that encoding of the data depends on the "m" previous blocks. The extent of the dependence on previous blocks is denoted in the description and the resulting code is referred to as an "(n,k,m) convolutional code." An equally acceptable alternative means to describe a convolutional code in the literature is to specify the code rate (where $R = k/n$) and the constraint length parameter K, which is the number of stages available to produce the output symbols (where $K = m + 1$).

Due to its memory characteristic, convolutional coding is implemented by a sequential logic circuit using linear feedforward shift registers. "Generator sequences" are obtained for the system by observing the impulse response of the n outputs. The

input data is convolved with the generator sequences, each "m + 1" symbols in length. The resulting codeword is a multiplexed sequence derived from the combination of the convolved inputs.

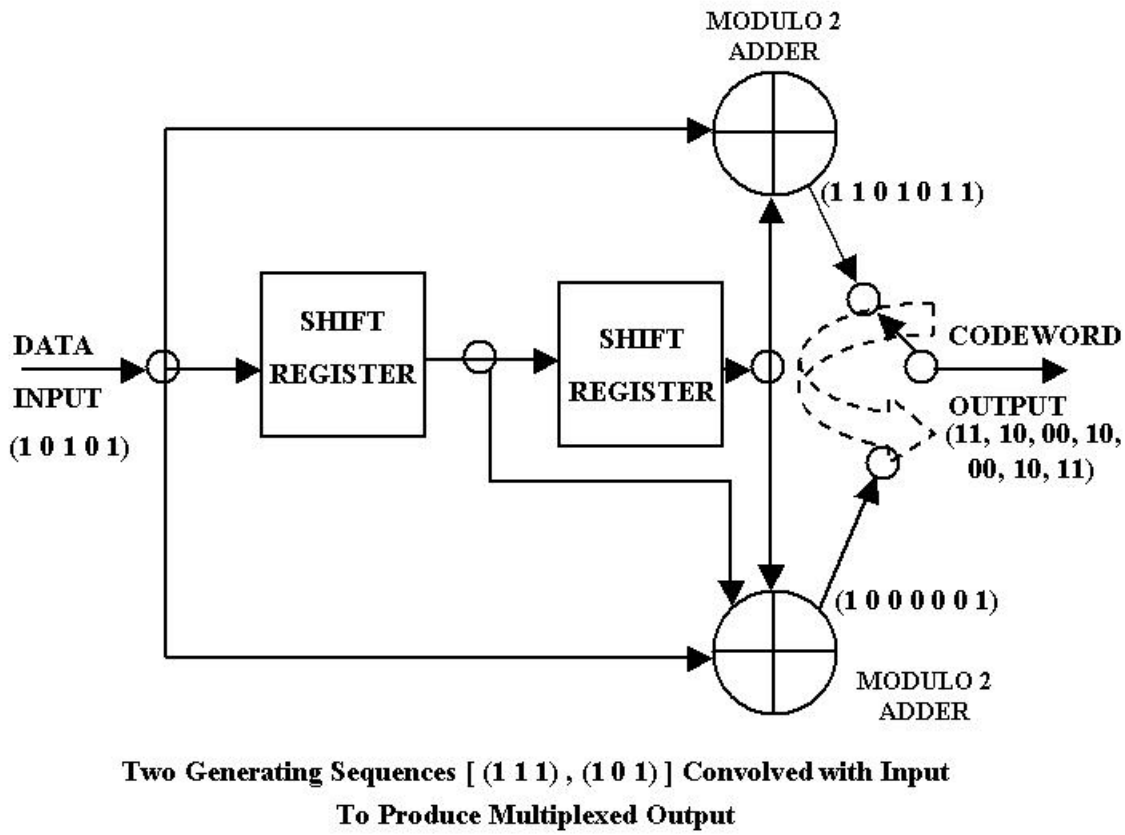


Figure 2. Binary (2,1,2) Convolutional Encoder (After ref [43])

For example, consider the binary (2,1,2) convolutional encoding diagram in Figure 2. This figure could also be referred to as a rate ½ convolutional code with constraint length parameter K = 3. The n = 2 outputs are represented by the modulo 2 adders, k = 1 is the data input, and the m = 2 shift registers are represented by the square elements. There are two impulse responses (corresponding to the number of outputs) and each reponse must be 3 digits in length (corresponding to the number of shift registers plus one which is the constraint length K). From impulse response, the generator sequences are determined to be (1 1 1) and (1 0 1). Let the input data sequence be (1 0 1 0 1). Discrete convolution of the input data sequence with each of the generator

sequences results in the 7 digit output sequences (1 1 0 1 0 1 1) and (1 0 0 0 0 0 1). The output symbols are created by "multiplexing" or grouping successive digits in one output sequence with the corresponding digits in the other. For example, the first digits of both sequences would be multiplexed into the symbol "11", the second digits into "10", and so on. Therefore, the codeword output from this encoder after multiplexing into a single sequence would be (1 1, 1 0 , 0 0, 1 0, 0 0, 1 0, 1 1).

Decoding convolution codes can be done by numerous algorithms but one of the most effective and widely used is the Viterbi algorithm. Viterbi decoding is based on the premise that the symbol sequence is deterministic hence it is possible to estimate or predict the most likely sequence of symbols that had been transmitted from the source to the receiver. Maximum likelihood decoding with Viterbi is a dynamic programming process that has been well recognized as the optimum solution for many noise contaminated channels. The method essentially determines the shortest path through a weighted graph of possibilities. There are three primary tools used to better visualize convolutional coding: the state diagram, tree diagram and trellis diagram. The most compact and simplest representation is the trellis diagram as shown below in Figure 3. Several excellent discussions on using the trellis diagram are available, including those in [40], [43], and [45].

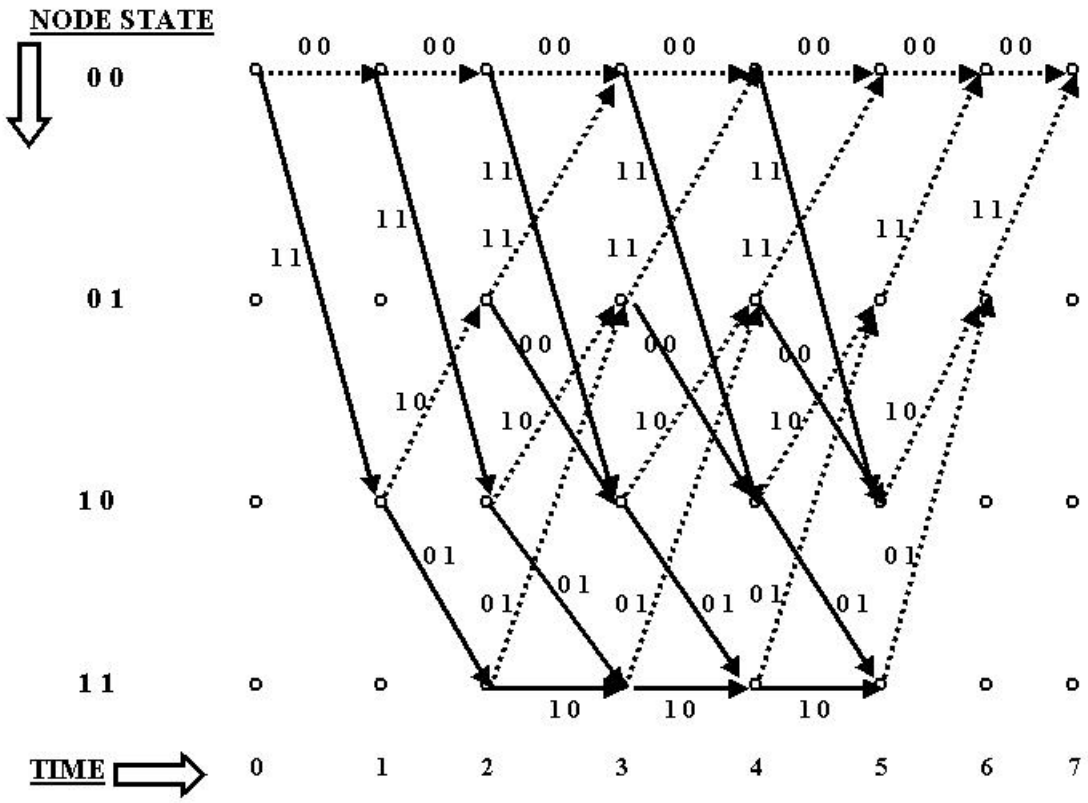


Figure 3. Trellis for Binary (2,1,2) Convolutional Code

Convolutional code trellis diagrams contain all the possible states and all the possible transitions from each state as time progresses as well as the output symbol that will be generated if that transition occurs. In all cases, the encoder is initialized at the beginning and "flushed" at the end of the series so that the state is known (node state "00" in the figure). If the initial and final states are unknown, the data will not be able to be properly decoded. Dark lines in the diagram indicate the path selected if the input digit is a "1" and the dashed lines are the path taken if the input digit is a "0". There are two bits of memory therefore four possible states. The two digit number associated with each line is the encoding symbol of that particular bit as it transitions to the new time and state. The codewords generated by following a path through the trellis are the only codewords feasible; receipt of any other combination indicates that the sequence symbols have been corrupted.

The trellis in Figure 3. directly corresponds to the encoder diagram in Figure 2. Recall that in the previous example, the parameters are $k = 1$ input, $n = 2$ outputs, and a memory order of $m = 2$. The trellis is best explained by going through the encoding steps previously completed for the input sequence (1 0 1 0 1). Referring to Figure 3. at time = 0, the encoder has been reset and is initially at state 00. The first bit input to the register is a "1" so the dark line path is selected, encoding the input bit "1" into the output symbol "1 1" and the state is 10. The next digit in is a "0" so the state transitions from state 10 to state 01 and the bit is encoded as the symbol "0 1." This process continues through the remainder of the trellis with the last two output symbols returning the sequence to state 00. The resulting convolutional codeword is identical to that previously determined, (1 1, 1 0 , 0 0, 1 0, 0 0, 1 0, 1 1). A trellis diagram showing only the states reached during encoding of the 5 digit sequence is shown in Figure 4. Note that the last two symbols are the result of the "flush bits" or zero inputs that returned the sequence to state 00.

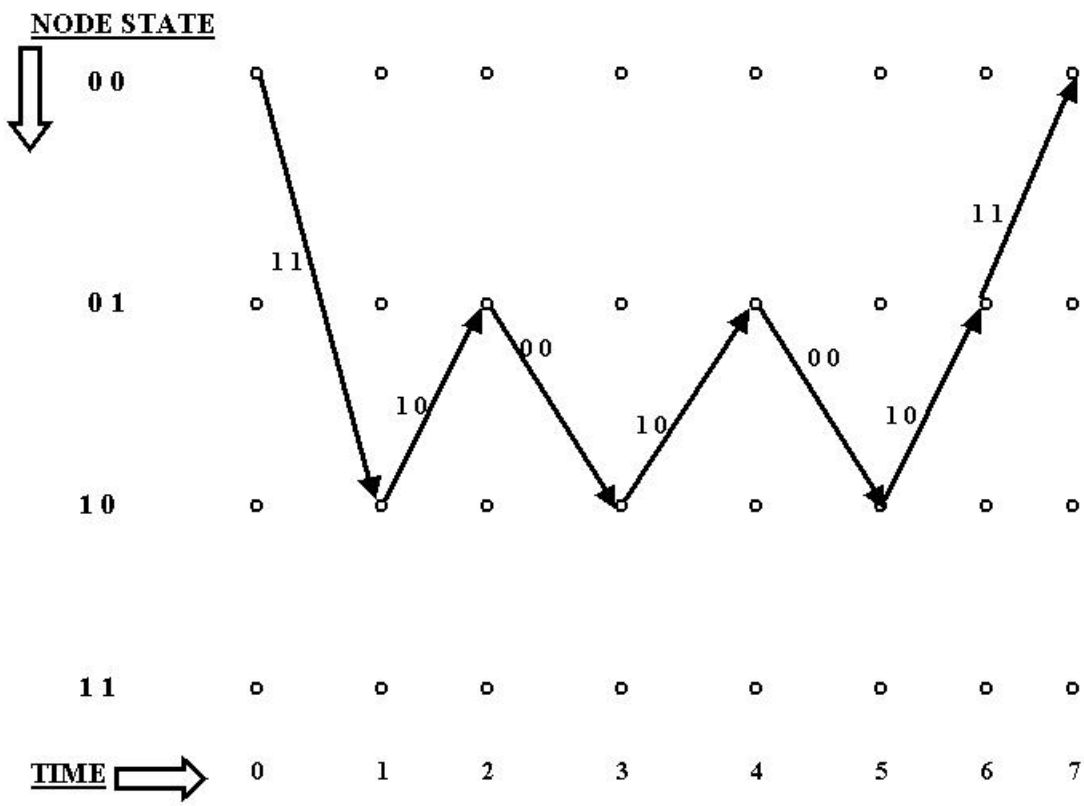


Figure 4. Encoding of Input Sequence (1 0 1 0 1) to (1 1, 1 0 , 0 0, 1 0, 0 0, 1 0, 1 1)

Now that convolutional encoding of data using the trellis has been explained, the next phase is to examine how the Viterbi algorithm uses this information to decode the received codewords and recover the original data. In simplest terms, the received sequence is used to "trace back" or recreate the most likely path through the trellis. This is implemented by assigning weights or a "metric" to each branch of the path. The received sequence is iteratively processed and compares path metrics at each state. The path determined to optimize the metric is stored as the "survivor" and all other paths leading into that state are eliminated. The process continues in this manner until terminated when reaching the known end state (state 00). Depending on the metric selected, the path that survives will either have the largest metric (maximized log-likelihood function) or the smallest metric (minimized Hamming distance). Once the path is reconstructed at the receiver, recovering the original bits that created the symbols is a simple matter.

The figure below illustrates using the Hamming distance as the metric for determining the most likely codeword sent. Hamming distance is calculated by counting the number of digits that differ between the symbol received and the possible symbol pairs on each branch. Continuing with the previous example, it is assumed that the string of symbols received was (1 1, 1 0 , 0 0, 1 0, 0 0, 1 0, 1 1). The received sequence is shown beneath the time line in the figure to aid in the computation of the metrics. When viewing the figure, remember that the algorithm computes branch metrics at each state and saves only the survivor, in this case the minimum value at each state node. The other branch entering the node is eliminated (indicated by the red X). Cumulative path metrics are listed in red above each node. After all branch metrics have been calculated, the most likely received path is selected by connecting the state nodes with the minimum value. This final path is highlighted by solid green lines. With knowledge of the final path, the original information sequence, (1 0 1 0 1), is then easily recoverable.

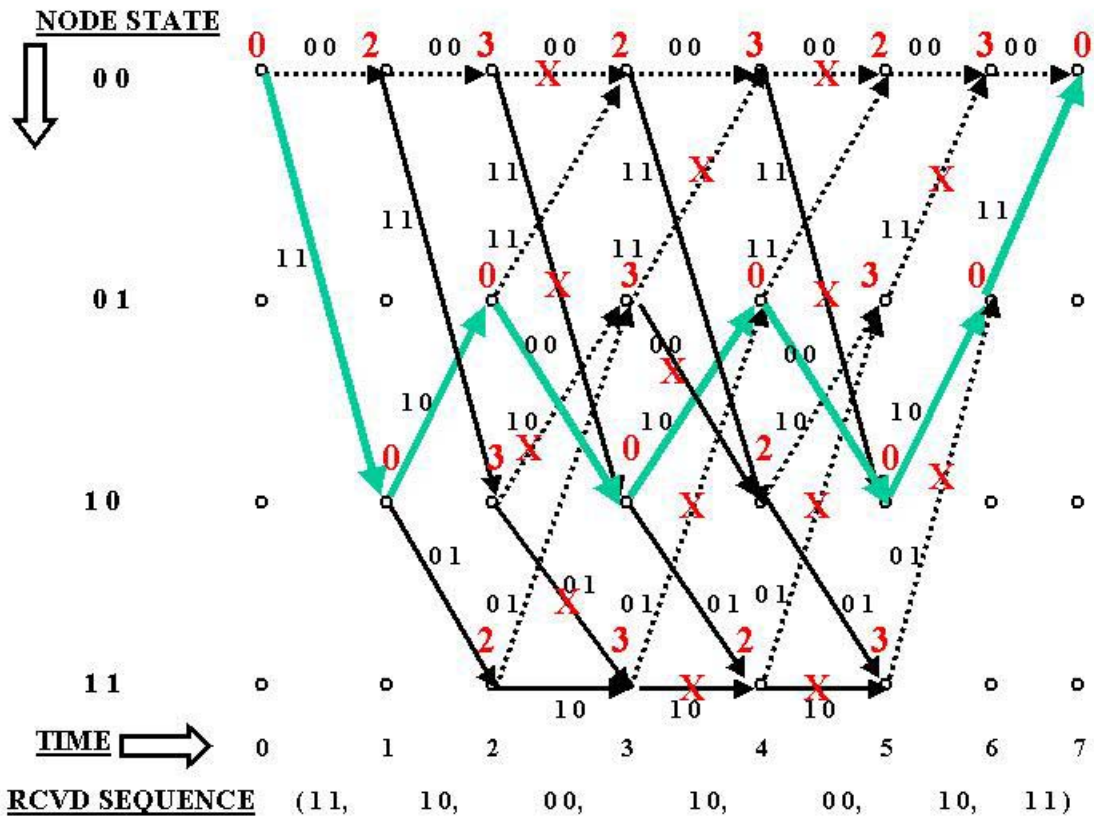


Figure 5. Viterbi Decoding Using Hamming Metric

The primary advantage of convolutional coding with Viterbi decoding is that the system performance is enhanced by "coding gain." Signal to noise (SNR) ratios are generally expressed in decibels (dB) and are usually defined as the energy per bit divided by the one sided noise density (E_b/N_0). "Coding gain" is the dB increase that would be required to achieve the same bit error rate (BER) if coding was not used. Viewed another way, a message that is convolutionally coded may be successfully received despite a lower SNR through the channel. The primary disadvantage to convolutional coding is that it requires a great deal of computational power and storage for processing the receiving sequences. Constraint lengths (K) for practical coding schemes are generally limited to around 9 since the complexity grows exponentially as a function of 2^K.

3. Advanced Coding Methods

Many variations on the two primary categories of channel encoding schemes, block and convolutional, have been developed. Two techniques used to increase code

reliability are "concatenating" and "interleaving." A concatenated code consists of an "inner" and "outer" code, essentially grouping any combination of the two coding methods into one larger code. The "dual" code produced has the advantage of being even more resistant to error when received. However, the cost of the added reliability is a loss of effective data throughput, since each code contributes additional redundancy. An interleaved code is one that alters the order of the codewords before transmitting them through the channel. The primary advantage of interleaving is time diversity in the signal, hence an increased likelihood of codeword reception in a channel susceptible to burst errors (e.g. open ocean). The disadvantages are that additional check digits are generally required and the receiver has the added complexity of deinterleaving prior to decoding.

Having only briefly reviewed the extensive discipline of information coding, it is clear that with every advance in hardware technology, there will be continuing research to produce faster algorithms and improved coding performance.

D. MODULATION AND TRANSMISSION

Once the signal sequence is encoded with redundancy to overcome problems introduced by channel noise, the next step is to map the bits or symbols into the signal waveforms and prepare the information for transmission into the physical channel. Three major and often competing design criterion in any modulation scheme are bandwidth, power, and cost. Typically, simpler modulation schemes are cheaper to design and may require less power but they use a larger portion of the frequency spectrum. On the other hand, complex modulation schemes require more complex and expensive equipment but are more bandwidth efficient.

Propagation in the physical channel requires mapping the digital information onto an analog signal waveform. There are numerous digital modulation methods but all consist of a basic carrier signal modified ("modulated") in some manner that must be detectable at the receiver. Carrier signals can be modulated either in amplitude, frequency, or phase. Modifications to the carrier characteristics contain the information to be transmitted. Choice of the best modulation scheme for a particular application is largely determined by the method of demodulation that will be used. The two primary

classes of demodulation are called "noncoherent" ("incoherent") and "coherent." Specific knowledge of the carrier phase is the basic difference between the two types.

1. Noncoherent

Noncoherent or incoherent demodulation methods (usually based on frequency modulation) are premised on the fact that only sufficient power level of the received signal is needed to determine the information sent. When a signal is received and demodulated, the frequencies with high energy levels indicate which symbols were transmitted. The main advantages of noncoherent demodulation are that it tends to require less SNR for good performance, generally costs less to implement, and it works well even in relatively noisy channel conditions. The primary disadvantage of noncoherent demodulation is that a larger amount of bandwidth is required to ensure the different signals are distinguishable. This disadvantage also leads to lower symbol rates or decreased data rate through the channel when compared to coherent methods. Examples of common noncoherent modulation techniques are frequency shift keying (FSK), multi-level frequency shift keying (MFSK), and amplitude shift keying (ASK), to name but a few. It is instructive to examine frequency shift keying and the terminology involved in more detail.

Binary FSK uses two separate signal frequencies, typically sine waves of constant amplitude. One frequency represents the binary digit "0" and the other (conventionally higher frequency) maps into the binary digit "1". The carrier representing the appropriate digit is transmitted for a time period referred to alternatively in the literature as the "chip duration," or "baud time" or "element length" or "pulse length." Regardless of the nomenclature used, the differential frequency between the high and low frequencies must be greater than or equal to the reciprocal of the transmission time period. Ensuring that the time divisions are separate will eliminate the possibility of "inter-symbol interference," a phenomenon that degrades data rates when the energy from the frequency representing one bit or symbol overlaps into a neighboring time slot. Consider the figure below where the message to be modulated is the binary representation of the letter "N" (0 1 0 0 1 1 1 0).

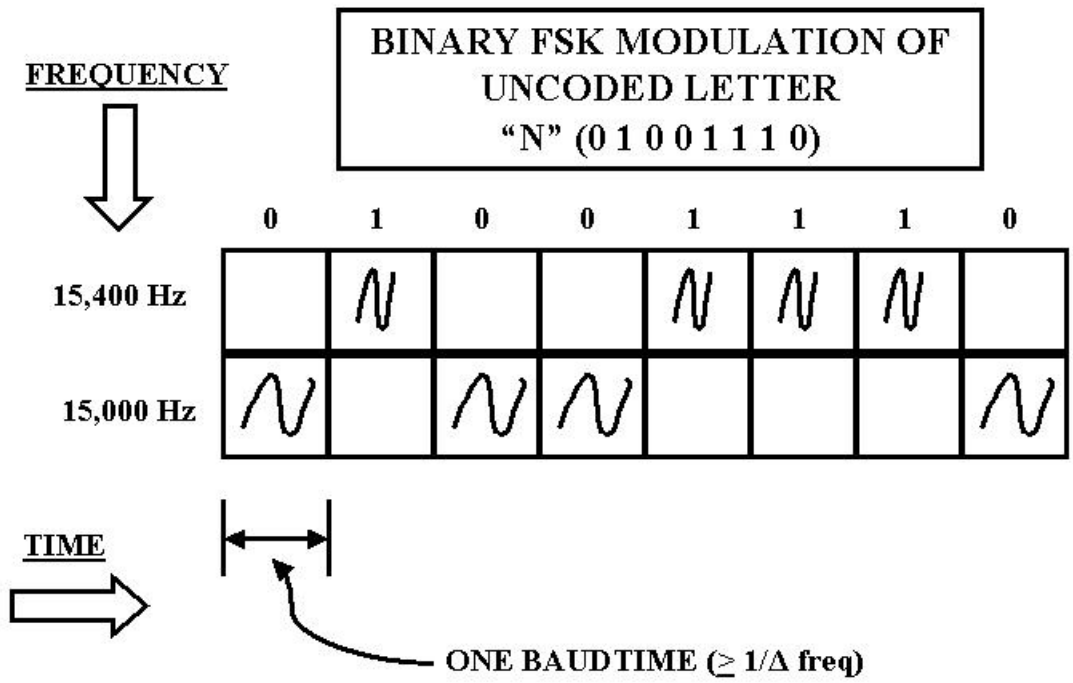


Figure 6. Binary FSK, Two Signal Frequencies (After ref [46])

In this binary FSK system, the 15.4 KHz sinusoid is "on" for the baud period used to convey the digit "1" and the 15.0 KHz sinusoid is present in the period when "0" is the intended message. The baud time is the time allowed for one change of state event. In the case of binary FSK, there is one frequency change (state event) to denote one message bit. Therefore, for binary FSK, baud rate is equal to bit rate. The minimum time between between events must be greater than or equal to the reciprocal of the difference between frequencies to allow detection and prevent inter-symbol interference. In this example, each baud time must be at least 1/400 or 2.5 milliseconds.

In an M-ary frequency shift keying, more signal frequencies are required but the amount of bits sent per frequency change (state event) also increases. The standard means of defining an M-ary communication system is $M = 2^q$ where q is an integer. M is the number of total frequencies available and q is the number of digits that will be in the

symbol corresponding to each frequency. For example, binary FSK is defined by q =1; there are two frequencies and each change in frequency denotes one digit. When q =2, M = 4 so there are four frequencies possible, each of which represent a two digit symbol. Similarly, there are three digit symbols and eight frequencies when q =3, four digit symbols and 16 frequencies when q =4, and so on. The advantage of M-ary frequency shift keying is that a greater number of information bits can be sent in less time. But the cost is a larger bandwidth requirement. Below is an example of 4-ary frequency shift keyed modulation. Note that the same amount of data sent binary in Figure 6. has been modulated in half of the time, but the frequency bandwidth required has increased.

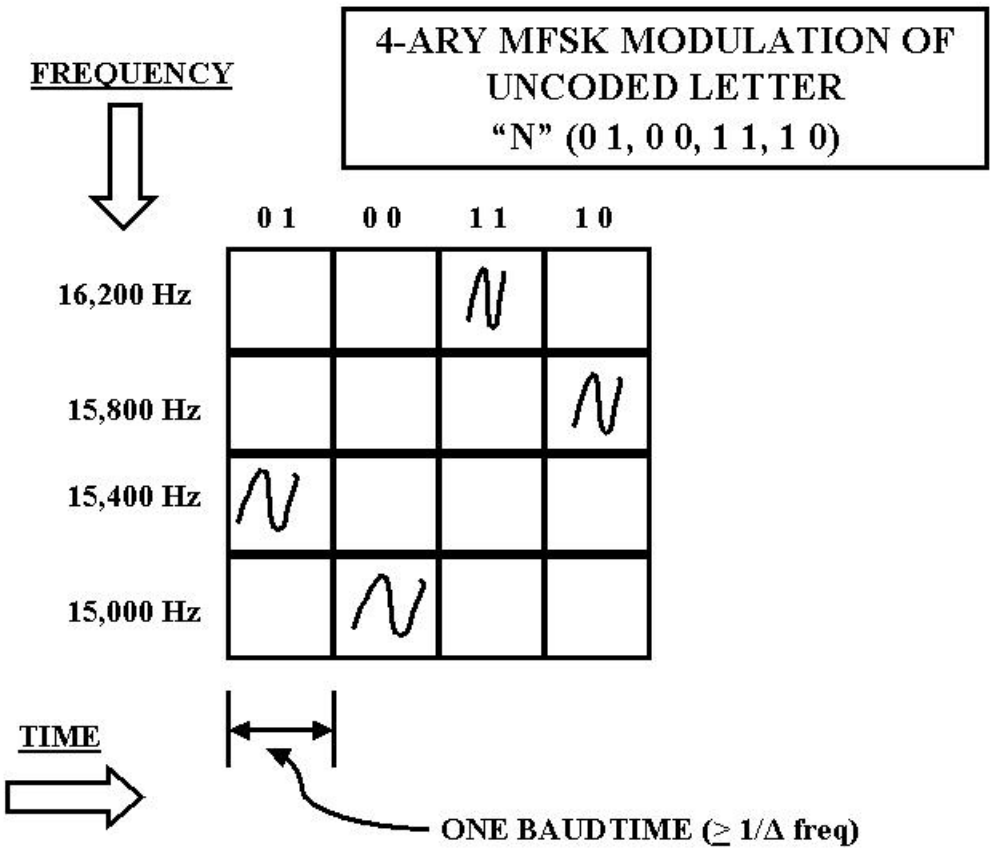


Figure 7. 4-ary FSK, Four Signal Frequencies (After ref [46])

Signals are processed during demodulation by passing them through a bank of bandpass filters for each frequency used. Using DSP techniques, the time history is

sampled and processed using a DFT to produce the frequency components. The magnitude squared output of time synchronized filters reveals which frequencies, hence symbols, were transmitted.

2. Coherent

Coherent modulation uses shifts in the signal phase to transfer the information bits or symbols. Common coherent modulation techniques include binary phase shift keying (BPSK) and quadrature phase shift keying (QPSK). Higher level phase shift schemes such as 8PSK, 16PSK, 32PSK, and 64PSK form "constellations" of possible states. Each state for a 2^q PSK will be matched with a symbol containing q bits of information. For example, 8PSK will have eight separate possible states, each with a symbol that represents 3 bits, 16PSK will have sixteen states with 4 bit symbols, and so on. Shown below is a schematic showing the possible states for the simple BPSK and QPSK modulation schemes.

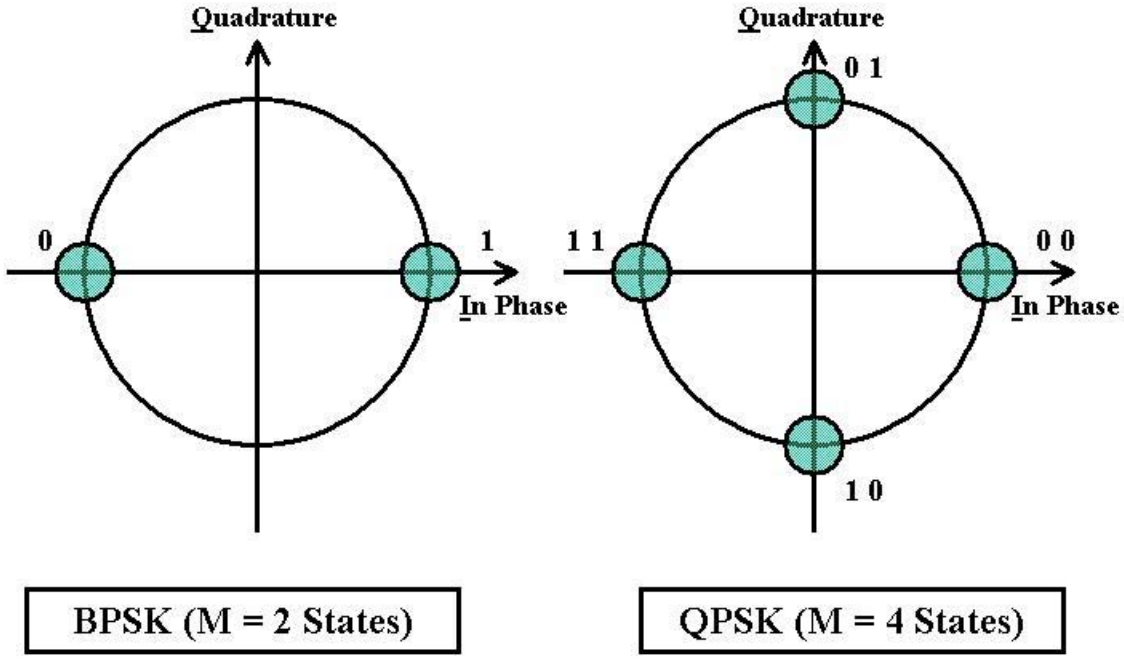


Figure 8. Phase Shift Key Mapping (BPSK and QPSK)

The greatest advantage to coherent modulation is that it is spectrally efficient compared to noncoherent methods, thus very attractive for use in a bandlimited environment. However, detection of the phase at the receiver requires much more complex filtering circuitry and integrated channel equalization algorithms, particularly in an inherently noisy channel such as the ocean. Additionally, in the higher order modulation schemes, symbols are closer together. Therefore, more power may be required during transmission in noisy environments to effectively differentiate them.

3. Advanced Modulation Techniques

Many advanced and hybrid methods to improve modulation performance and increase data rates are available to the system designer. Two particular noteworthy areas to discuss are frequency hopping (FH) and multiplexing or multiple access techniques.

a. Frequency Hopping (FH)

"Spread spectrum" techniques are methods that allow for multiple users in the same frequency band. The two main categories are direct sequence spread spectrum (DSSS) and frequency hopping (FH). While much research continues toward refining DSSS and coherent signaling [47], frequency hopping for noncoherent MFSK signals has been widely used. Frequency hopping essentially divides the entire spectrum available into a large number of contiguous frequency slots. A pseudorandom or pseudonoise (PN) generator is used to generate the hopping pattern and translate the M-ary FSK signal into one of the frequency slots just prior to transmission. The receiver has an identical pseudorandom generator that must be synchronized in order to properly remove the frequency translation and recover the original signal for further processing.

Additional advantages of FH spread spectrum communication are that, by its randomized nature, it can overcome noise in a particular slice of the bandwidth and is more difficult for an adversary to detect or jam. The disadvantages are the added computational complexity and hardware costs.

b. Multiple Access

The spread spectrum techniques belong to a class of multiplexing or multiple access called code division multiple access (CDMA). The multiple access methods allow many users to share the same operational area. There are two other multiplexing methods that may be used in underwater communications. The first is

called frequency division multiple access (FDMA). In FDMA, users are each allocated a portion of the overall bandwidth for their exclusive use. Time division multiple access (TDMA) is similar in the exclusive use aspect but assigns each user a particular time period rather than a frequency band. Further discussion of multiple access methods and current research trends may be found in [40], [48], and [49]-[51]. In particular, [49] discusses multiple access methods and concludes that CDMA and spread spectrum techniques seem to be the best solution for shallow water multi-vehicle networks.

4. Transmission

Prior to physical transmission of the data, the protocol for inter-modem communications must be considered. Protocols determine system characteristics like the packet or frame size, how the data will be sent, and any specific handling instructions. While there are various protocols currently in use (see, for example, [52]-[54]), many common elements exist.

a. Frames or Packets

Every underwater modem has a specific maximum amount of data that can be held in a buffer for transmission. Dividing up and sending this information requires an efficient format or protocol that will be understood by both the transmitter and intended receiver. This is done by segmenting the data into pieces referred to in the literature variously as a "packet" or a "frame." The primary value of frames is that they provide a method of timing control or synchronization to enable proper message processing at the receiver. While the structure for packets is not standardarized, most packets are constructed with a small number of header and/or trailer bytes and a large number of data or information bytes, where a byte is defined as 8 bits. An example of packet structure is shown below for nominal 32 byte (256 bit) frame.

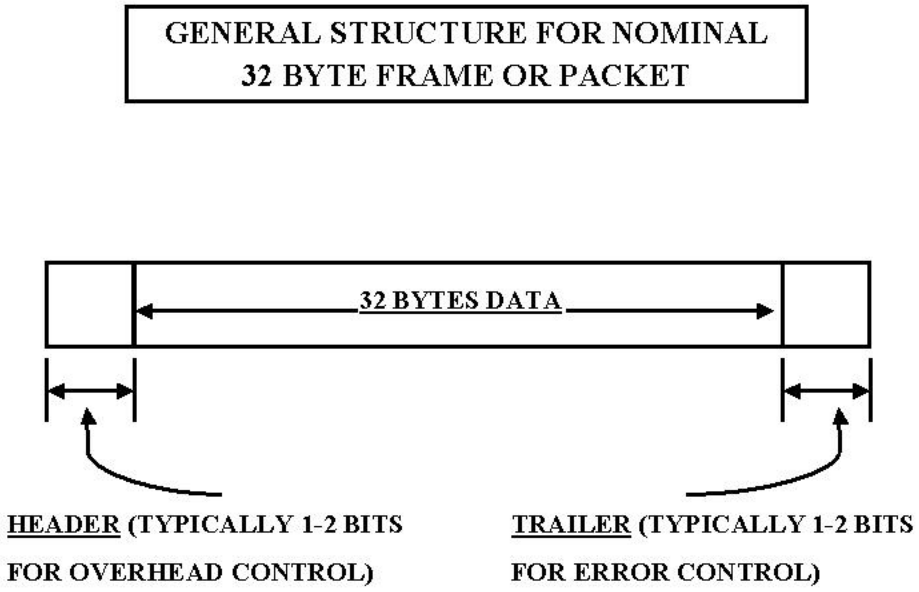


Figure 9. Typical Frame or Packet Structure

The header and trailer bytes represent overhead bits that can be used for control purposes. Some typical uses of the header bits include synchronization, assignment of modem addresses, and information about the amount of actual data in the frame. The trailer bits typically are used for some method of error checking, such as a check digit or a cyclic redundancy check sequence (CRC). Most protocols allow for multiple frames to be concatenated or grouped together into one large packet, thereby reducing the number of bytes expended on overhead error control. Typically, frames that do not have the required minimum bytes of data will be "padded" with zeroes to maintain frame size. The excess zeroes are removed during processing. A very important point to note is that the entire frame (or group of frames) is processed as a unit through the channel encoder and digital modulator. This means that there are effectively layered coding schemes: CRC for the entire packet and block or convolutional coding for the data symbols.

b. Physical Transmission and Reception

Physical transmission of the information packets or frames generally requires several further actions after completion of digital modulation. An inverse FFT

of the output spectrum of modulated signals is completed. The time domain data is then sent through a digital-to-analog (D/A) converter to produce the analog waveform. An acquisition waveform may be prepended to the signal at this point as a means to "wake up" the receiving unit. The combined signal is then put through a power amplifier and sent to the transducer element. The final stage of the transmission process occurs when the transducer converts the signal into an acoustic pressure wave. The pressure wave emanates depending on the beam pattern of the transducer: hemispherically if the transducer is omnidirectional or concentrated into a reduced spatial sector if the transducer is directional. When the pressure wave is detected at the receiver, all operations that had been previously performed at the transmitter are now done in reverse order to extract the message. The pressure wave is converted back into a signal, preamplified, and sent through an analog-to-digital (A/D) converter. The signal is synchronized and the prepended acquisition waveform is removed. An FFT is used to convert the digital signal back into the frequency domain. The packets are demodulated, decoded and the overhead information (header, check digits) is removed. Finally, the message is output to the recipient.

E. THE PHYSICAL CHANNEL AND ENVIRONMENTAL FACTORS

The acoustic channel is well known and well documented as an adverse environment with multiple impediments to underwater communications. However, as stated earlier, it is the best of the limited alternatives available. Amplifying information on the rejected alternatives is in Appendix A and detailed discussions of the acoustic environmental properties may be found in many excellent texts such as [26], [37], and [38]. Major impediments to underwater communications include the sound speed, transmission losses, multipath or reflection , and transient noise sources.

The speed of sound in seawater is nominally approximated as 1500 meters per second. Propagation speed is more than five orders of magnitude slower than RF so delay times are inherent in the ocean channel. Additionally, the sound speed is not a static value but depends upon water density which, in turn, depends on the local values of salinity, temperature, and depth. Sound waves bend and are reflected from property boundaries. Fluctuations in the sound speed profile between the transmitter and receiver

are nearly guaranteed due to the spacial and temporal distances. These fluctuations will cause the signal energy path to bend in a manner depending on local conditions. The result could be either lost signals and dead zones, or a requirement to transmit at a higher power.

Transmission losses are another barrier to underwater communications. These losses include the effects of spreading and attenuation. Spreading loss is the fading in the power of the signal as it radiates from the source. For omnidirectional transducers, signal fading is significant; the loss due to spreading increases as the square of the range (spherical spreading). Attenuation losses are primarily due to signal energy absorption by the seawater. The total transmission loss is commonly measured in decibels (dB) and is given by the equation $TL = 20 \log R + \alpha R$ where TL is transmission loss in dB, R is the range in meters, and α is the absorption loss per meter. The figure below, reproduced from [56], shows absorption dependence on frequency in both fresh water (curve a) and seawater (curve b).

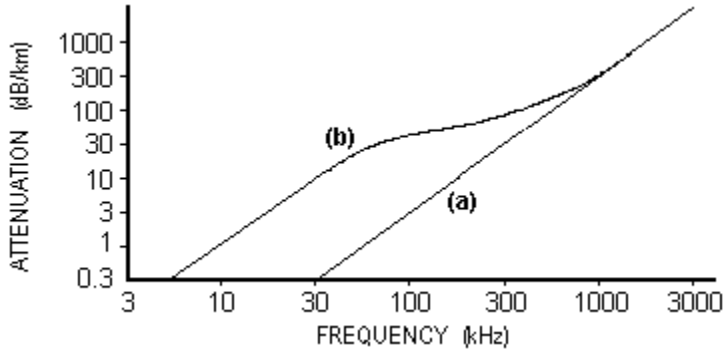


Figure 10. Acoustic Absorption as a Function of Frequency (From ref [56])

One of the worst characteristics of absorption is that it rapidly increases with increased frequency. The usuable bandwidth in the underwater environment is effectively limited by this property of absorption. Increased ranges require lower frequency transmission but lower frequency transmission equates to smaller bandwidth. Reduced operating bandwidths, in turn, decrease data transmission rates.

Multipath is defined as multiple propagation paths for the signal between the source and receiver. The primary source of multipath for shallow water communications arises from reflections from the surface and ocean floor, but other obstructions between

the source and receiver may contribute. The undesirable result is severe signal attenuation and interference, particularly in a shallow water channel where the operating depth of the water is substantially less than the range to the receiver. In multipath transmission, many different reflections of the signal may reach the receiver at nearly the same instant, causing distortion to the amplitude and phase of the transmitted signal. Additionally, multipath conditions are not stationary. Relative motion on either end of the channel introduces a Doppler shift and frequency broadening which further reduces the likelihood of receiving and correctly decoding the message. Finally, multipath may also contribute to signal fading by a destructive interference combination of two paths. An example of the severe multipath encountered in very shallow water is reproduced below from [57]. Note the numerous ray trace propagation paths evident even at the relatively short range of 0.5 kilometers.

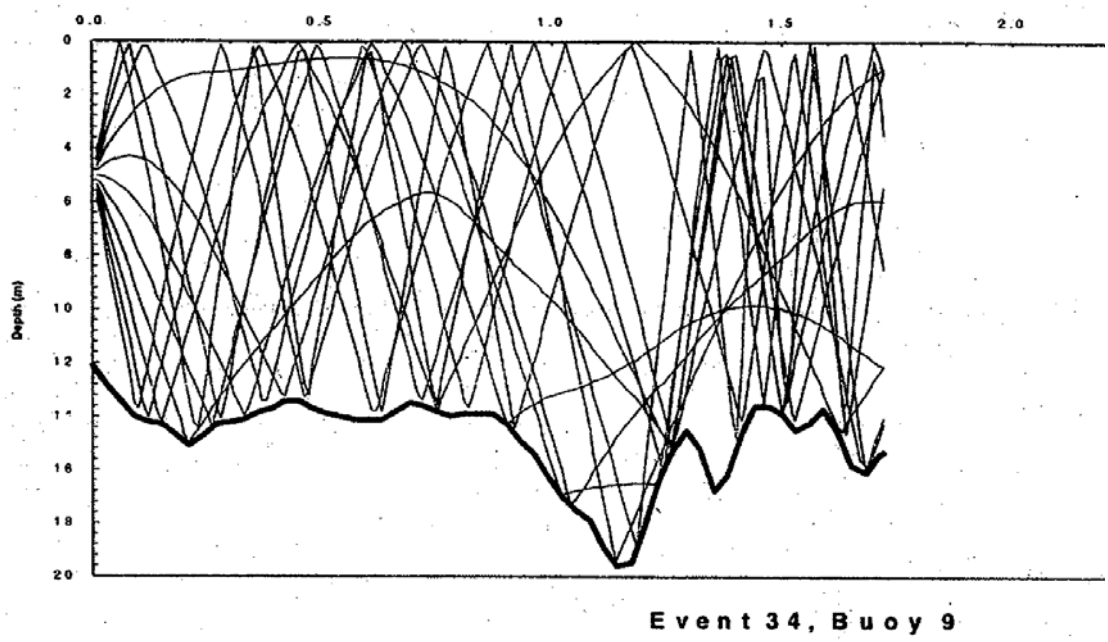


Figure 11. Propagation Ray Trace, Very Shallow Water (From ref [57])

The last category of major impediments to underwater communications includes all transient noise sources, natural and man-made. Marine life in an operating area will contribute to the ambient noise levels and may directly interfere with the transmission path. Shipping and other human activities may also introduce noise into the marine environment. The effect of both is a decrease in the SNR at the receiver.

THIS PAGE INTENTIONALLY LEFT BLANK

III. ACOUSTIC CONTROL

Command and control refers to the process whereby an operator or tactical controller has the lines of communication, authority, and the ability to alter operating parameters of a deployed system, whether human or hardware. Command and control implies a hierarchical two-way communication; commands and data may be sent to the deployed system from the tactical controller but usually the deployed system only sends data and acknowledgements in return. The autonomous system ofter sends prompts for action. In general, the tactical controller for any autonomous system is a human operator. A communication flow diagram of the control process is shown below. As previously discussed, operation of autonomous platforms such as AUVs in the unique underwater environment demands use of the acoustic channel as its line of communication.

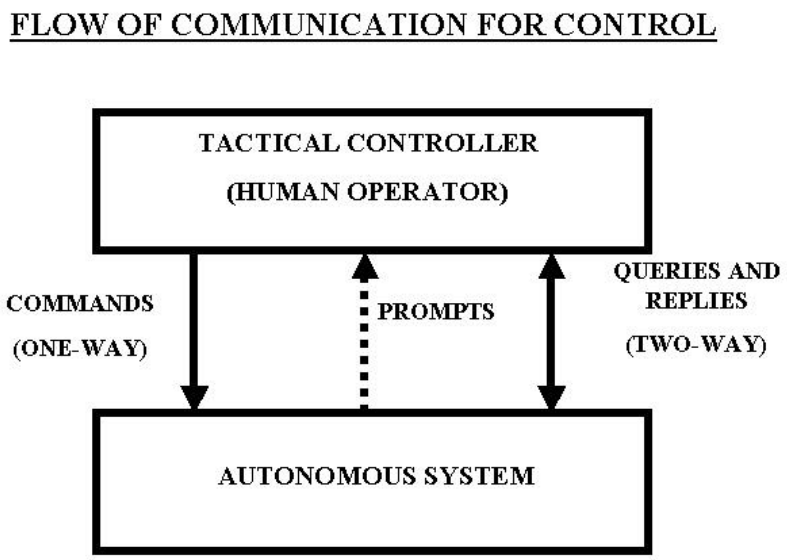


Figure 12. Control Communications Flow

When considering an AUV system, there are several layers of "control" occurring simultaneously. On the basic level, the vehicle has an operating system that completes programmed instructions in a prescribed manner. Sensors monitor variables that may

lead to alteration of the plan and subsequent servo level actions to control surfaces or thrusters. Tactical control of an AUV by a human operator is generally limited to either changes to a condition or input, or the monitoring of vehicle status. One-way commands to the vehicle may be of a directive nature, such as to change operating modes and set point values, or a two-way query that causes the vehicle to return a reply on the current state of vehicle systems. A schematic of the control block diagram is shown below.

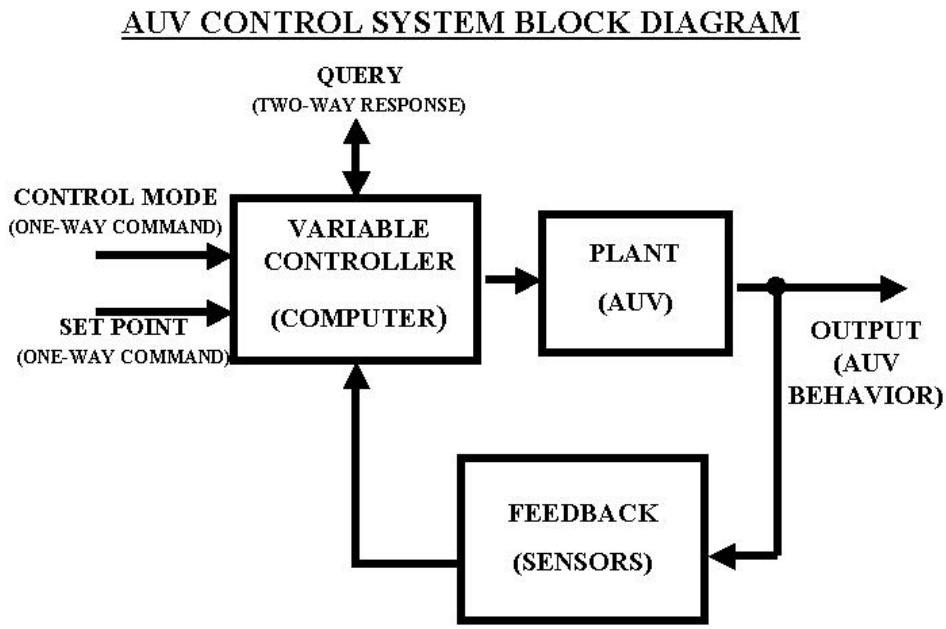


Figure 13. Autonomous Control System Block Diagram

This section deals with the development and validation of a tactical controller using the NPS ARIES as the testbed AUV. An acoustic link capability was added to the vehicle and software was developed to enable one way and two way data transfer. Software was written so that new servo-level set point values could be accepted by the existing controller operating system. The software design has incorporated a separate control process for reading and interpreting data from the acoustic modem, and new software for reading new command values. Also, inter-process communication of vehicle state information was designed using shared memory structures. This allowed for

vehicle state data to be written into strings that were sent to the ARIES modem for communication out to the tactical controller.

The majority of the experimental work on command and control was completed using the Florida Atlantic University (FAU) modem installed in the ARIES AUV. This section begins with two chapters describing modem operating parameters and all hardware used during the subsequent experimental investigations. The next four chapters are devoted to the results of the control with an acoustic modem study. Chapter C. discusses the ARIES computer code development necessary for this work. Chapters D. through F. examine the performance characteristics of the installed system and the results of behavior control experiments. Chapter G. is a summary of the control work in this section.

A. FAU MODEM OPERATING PARAMETERS

Having completed communications theory review in the last section, this chapter examines the operational parameters and the encoding and modulation schemes specific to the modem system installed in the ARIES vehicle. Detailed information on the development and operating parameters of the Acoustic Modem can be found in references [54], [58]-[61].

1. Basic Specifications

The FAU modem is a general purpose modem that was designed to use variations of multi-frequency shift keying (MFSK) modulation to transmit encoded data at nominal coded rates from 200-1200 bits per second. Operating in the nominal frequency range from 16–32 KHz, the system uses an omnidirectional transducer and a 40-60 volts supply voltage. Signals are sent out at a maximum source level of 192 dB with peak power consumption up to 250 watts rms. Frames are composed of encoded data pulses to provide 256 bits (32 bytes or characters) of information plus header information to synchronize the transmission and specify the modulation method. Additionally, there is a cyclic redundancy check (CRC-8) which adds 8 bits to each message. As previously discussed, the CRC is used to ensure the integrity of the entire message, not the individually encoded symbols. The modem can be configured to select any of three data encoding schemes with any of four modulation methods.

2. Encoding Schemes

The data may be encoded either with block, convolutional, or dual encoding schemes. Recall that the fundamental purpose of any error correcting encoding method is to prepare the data for reliable transmission over a noisy channel. The added redundancy enhances successful decoding at the receiver but introduces overhead costs that reduce the channel throughput. The block encoding scheme used in the FAU modem is the one half rate Bose, Chaudhuri, and Hocquenghem (BCH) linear, cyclic block code. Recalling the definition of code rate, this means that there will be twice as many bits to transmit as there are data bits. The second coding scheme is the $K = 7$, rate ½ convolutional code. Recall that K is the "memory" or number of stages available to produce the output symbols. It is also the size of the trellis when the message is decoded using Viterbi decoding at the receiver. As in the case of solely encoding with BCH, there are twice as many bits as data bits to transmit. The final encoding method is called "dual," a concatenation of the two individual types. First the data bit is block encoded with BCH then convolutionally encoded. The process is reversed on the receiving end, decoding with Viterbi followed by BCH. The dual encoding method has the advantage of being the most robust but at double the overhead cost. Four bits must be sent for each bit of actual information data.

3. Modulation Methods

The modem can be configured for any of four possible modulation methods for transmitting the data. In all cases, some form of frequency hopping MFSK is used. The total available bandwidth from 15.6 KHz-32.2 KHz is divided in 56 bins, each slot about 295 Hz wide. Only 32 slots (bins 10-25 and 27-43) are used for data transmission with the remainder for control purposes such as synchronization. The modulation methods are referred to as "modes 1-4." The number of bits per pulse (also referred to as bits per signal event or bits per baud time) range from the greatest in mode 1 (16 bits/pulse) down to the least in mode 4 (3 bits/pulse). The pulse width is 13.54 milliseconds and the number of pulses per frame varies based on the encoding and modulation combination selected. Modes 4 and 3 are the more robust modulation schemes but subsequently suffer from lower data rates and, therefore, longer transmission times.

The 32 frequency bins are divided into four groups of eight frequencies for modes 2, 3, and 4. The symbol size (q) for each of these modes is 3 bits, which comes from the fact that there are eight possible frequencies (M) for each chip or symbol and $M = 2^q$. These three modes differ primarily in the established hopping patterns and the level of "packing," or number of bits sent per pulse. Mode 4 transmits one frequency from only one of the four groups of eight frequencies during each pulse and "hops" between groups on subsequent pulses. Simply stated, mode 4 transmits a frequency mapped to three bits of the message in each pulse period. Mode 3 hops between two of the four groups of eight frequencies during each pulse, transmitting one frequency in each of the two groups. Mode 3 modulation results in the transmission of six bits per pulse. Mode 2 does not hop between groups but uses all four groups simultaneously, transmitting four frequencies so twelve bits per pulse. Diagrams of all three modes are shown below.

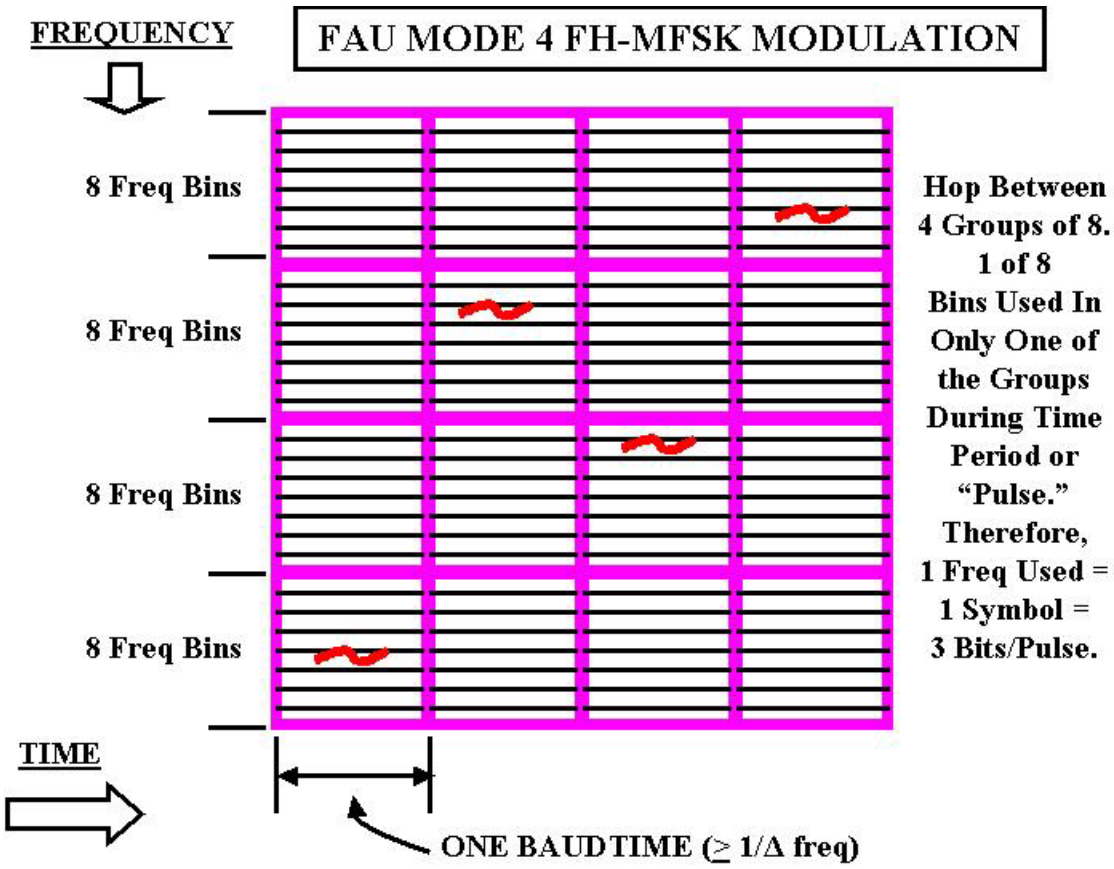


Figure 14. Mode 4 Modulation, FAU Modem

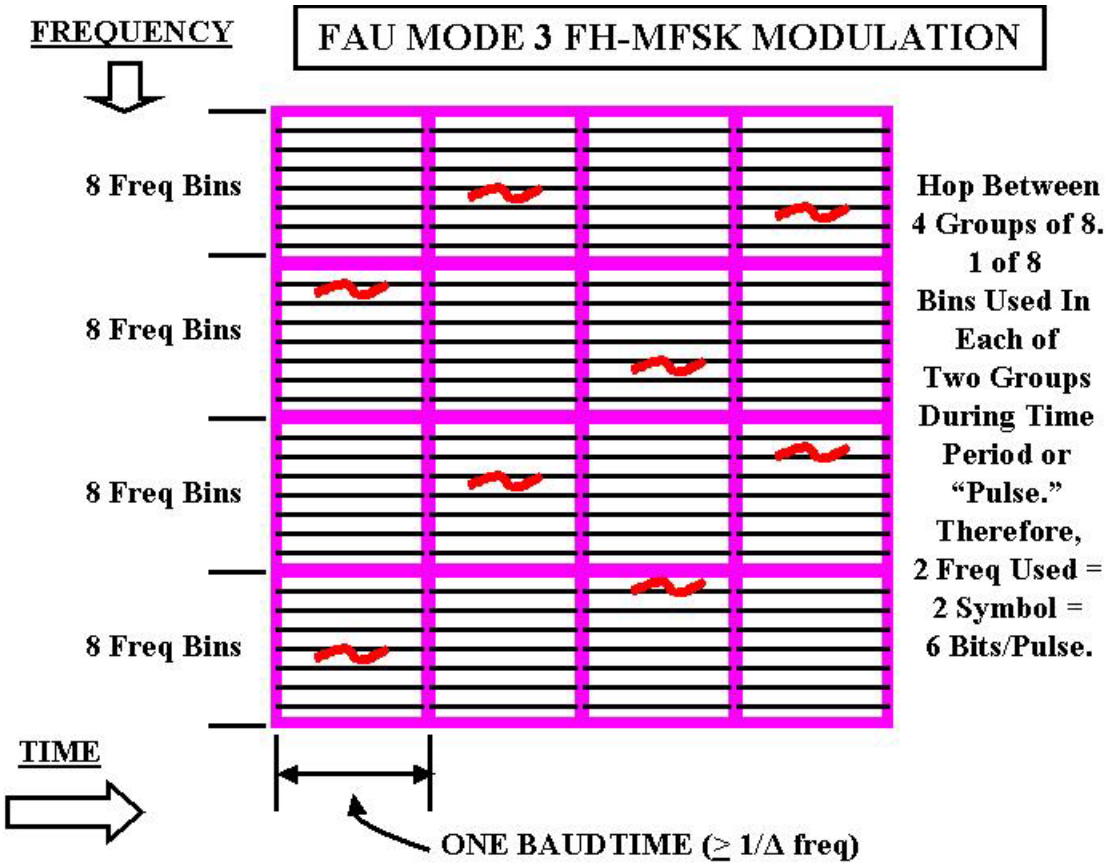


Figure 15. Mode 3 Modulation, FAU Modem

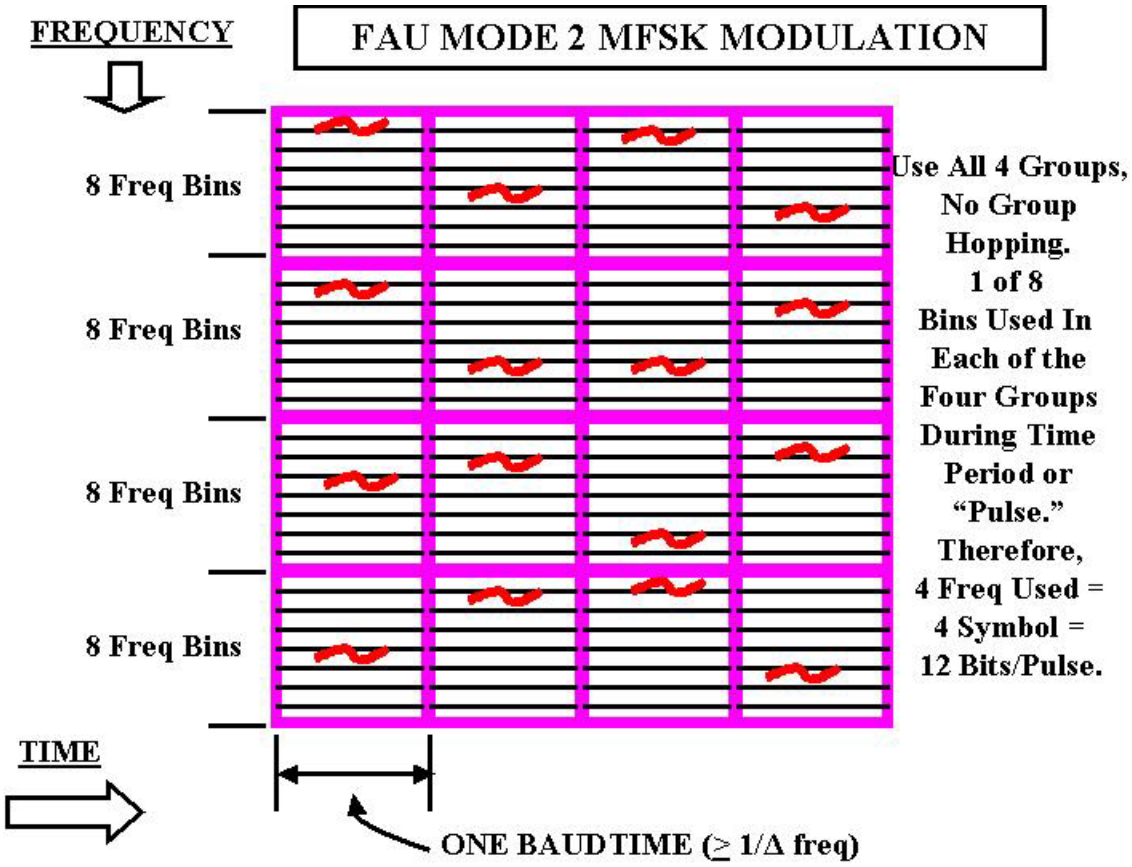


Figure 16. Mode 2 Modulation, FAU Modem

The greatest number of bits per pulse are transmitted using mode 1 modulation. The symbol size (q) for this mode differs from the other three modulation methods. There are four possible frequencies (M) for each chip or symbol and $M = 2^q$. Therefore, only 2 bits per pulse can be represented by each frequency sent. Mode 1, similar to mode 2 does not hop between groups. Unlike mode 2, mode 1 uses eight groups of four simultaneously, transmitting eight frequencies equating to sixteen bits per pulse. A diagram of mode 1 transmission is shown below.

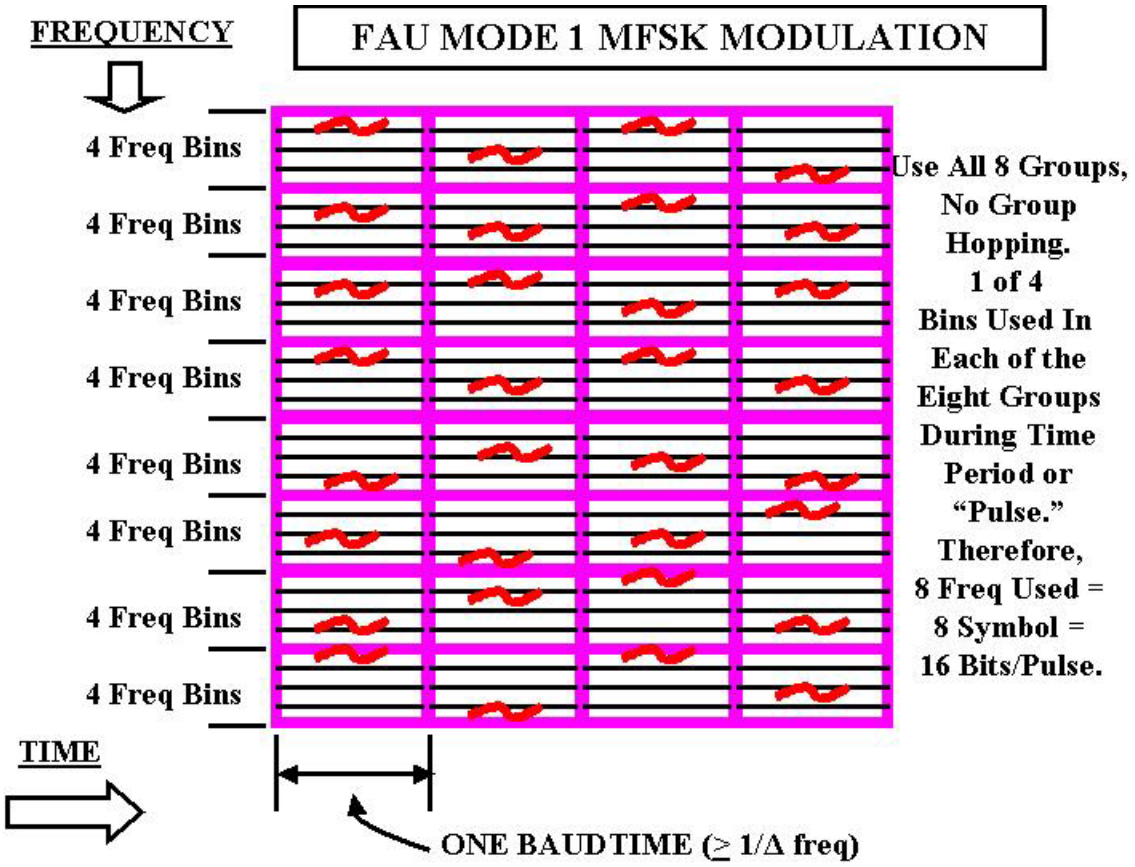


Figure 17. Mode 1 Modulation, FAU Modem

In summary, as the mode decreases from mode 4 to mode 1, a greater amount of data is sent per pulse so the amount of time to send a message is decreased. However, the disadvantage is that the increased packing of the waveforms makes the signals more difficult to detect and process, resulting in less reliable reception.

4. Summary of Theoretical Performance

Theoretical performance for the twelve combinations of encoding and modulation methods are reproduced in the table below from [61]. BCH refers to block encoding, Viterbi refers to convolution coding, and dual is a concatenated combination of the two. Two items of particular note are the relationship between data rates and range and the effects of coding. In general, as the rate of transmission increases, the effective range decreases. Also observe that additional coding improves the range for reliable transmission, but at a significant loss of data rate. Range data is assumed to be estimated

from a relatively noise-free open ocean transmission loss model without consideration of the very shallow water surf zone background noise and multipath problems.

Mode	1-BCH	1-Viterbi	1-Dual	2-BCH	2-Viterbi	2-Dual	3-BCH	3-Viterbi	3-Dual	4-BCH	4-Viterbi	4-Dual
Coded Rate (cps)	1172	1172	1172	880	880	880	440	440	440	220	220	220
Data Rate (bps)	586	586	293	440	440	220	220	220	110	110	110	55
Range (m)	200	200	500	500	500	1000	2500	2500	3000	4000	4000	5000
Bit/J at 192 dB	4.69	4.69	2.34	3.52	3.52	1.76	1.76	1.76	0.88	0.88	0.88	0.44

Table 2. Theoretical FAU Modem Performance (from [61])

B. HARDWARE ISSUES

1. Configuration of Modem

a. In the ARIES Autonomous Underwater Vehicle

The Naval Postgraduate School (NPS) has been involved in underwater research and control technology since the inception of the Center for AUV Research in 1987. Currently on a third generation vehicle, the Acoustic Radio Interoperative Exploratory Server (ARIES) AUV is an ideal testbed platform for reconfiguration and the testing and evaluation of new hardware and software. The ARIES includes a large suite of integrated sensors for navigation, command, and control [36]. A schematic of the major hardware arrangement is shown below and operating procedures are in [62].

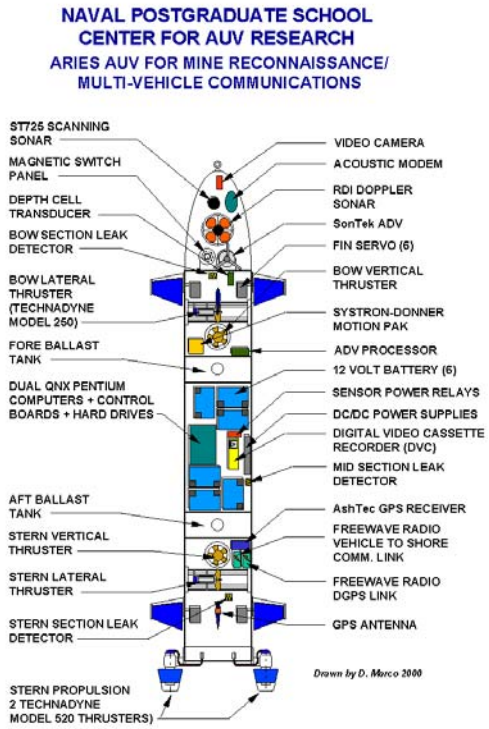


Figure 18. ARIES Hardware Schematic (From ref [36])

The hardware for the Florida Atlantic University (FAU) general purpose acoustic modem has been under development for several years. An overview of its development and operating characteristics are well documented in [58] and [59] with the component wiring diagrams in [60]. In order to properly integrate the acoustic modem set into the ARIES vehicle, the system required separation into three primary subcomponents: an amplifier/power supply section, a signal processing section, and the transducer.

The Amplifier and Power Supply were built on a chassis 4 inches in diameter. The chassis was located in the center of the forward compartment. As will be discussed later, a casing not shown in the figure below was fabricated to overcome Electromagnetic Interference (EMI) problems. The Digital Signal Processor (DSP) and its associated electronics were on the separate rectangular board shown in the figure

beneath the amplifier and power supply. The board was placed in the forward compartment directly beneath the servo motor for the starboard bow fin.

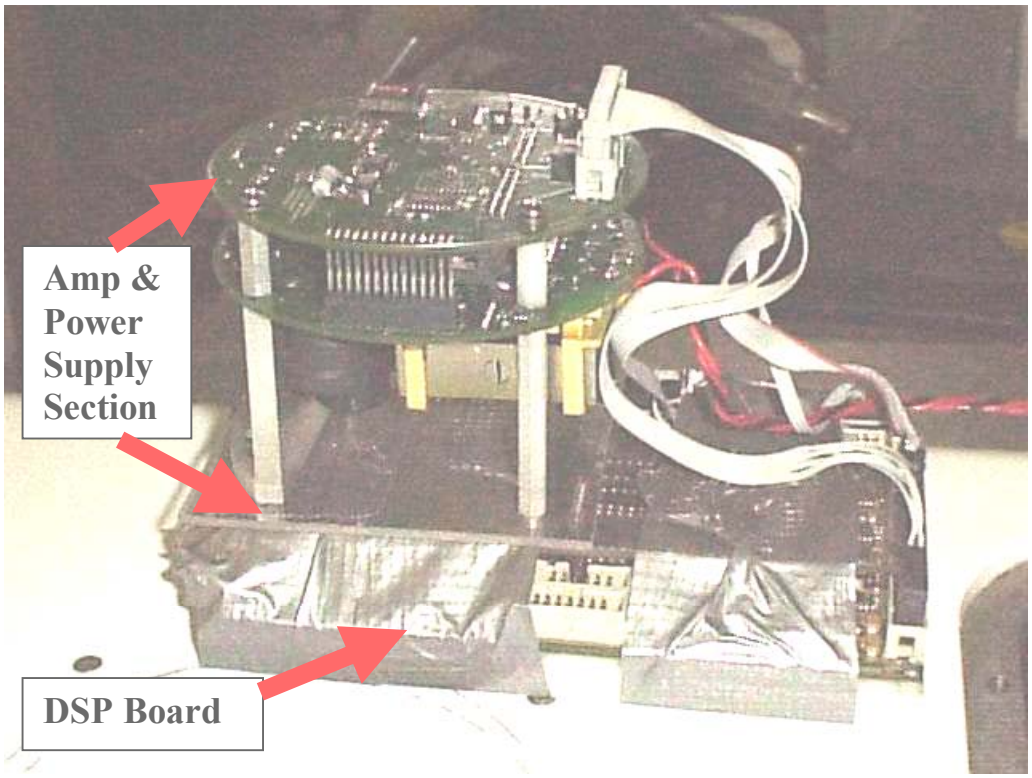


Figure 19. Installed FAU Modem Board Set: Power Supply/Amp (cylindrical top) and DSP Board (rectangular bottom)

The third primary component was a transducer molded into a hydrodynamic housing then mounted on the lower fairing of the free-flooded nose section of the ARIES, as shown in the figure below. Connection to the modem board set was through a waterproof SeaCon pie connector in the forward vehicle bulkhead. The transducer operated in the nominal 16-32 KHz band.

The transducer was connected to the lower side of the vehicle nose. In this manner, acoustic communication with the vehicle was maintained, even when surfaced. At no time was it considered that ARIES would be used as a deep water node communicating to the surface, but rather the other way around.

Figure 20. FAU Modem Transducer Mounted on ARIES Fairing

b. Topside Modem

The modem shown below is referred to as the "topside" or local modem. The topside modem was transported into and around the operations area onboard a 22 foot Boston Whaler support boat. Modem communication and control was carried out via a Freewave RF link from the host laptop computer physically located at the pierside Command Post.

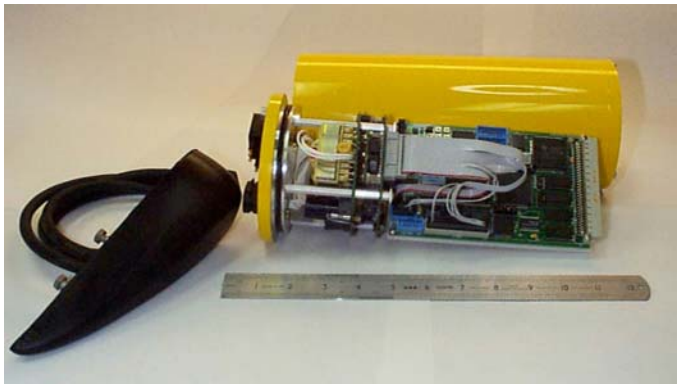


Figure 21. FAU "Topside" Modem Configuration (from [61])

2. Ancillary Equipment Used

In addition to the modems and the ARIES, many additional items were essential to the experimental investigation. The primary equipage discussed below was used either to monitor and assess environment factors or provide general support to the operations.

a. DiveTracker™

Background noise levels were taken during experimental testing with the DiveTracker™ developed by Desert Star Systems. A multifaceted underwater tool [63] , the DiveTracker™ can be preloaded with software for versatile applications ranging from diver applications to signal analysis. The SonarScope™ V1.00 application software allowed investigation of the noise in the range of interest. SonarScope™ operates in three different modes: receive, transmit, or spectrum scope. The Spectrum Scope Mode was used to collect readings in the 19-28KHz band to ascertain background noise levels in the frequency range of modem data pulses.

b. Sea-Bird CTD

Channel conditions were sampled and recorded during the long range modem performance tests using the SEACAT SBE 19 CTD Profiler made by Sea-Bird Electronics, Inc. The CTD profiler is an oceanographic instrument used to measure the conductivity, temperature, and depth. These measurements were used to produce representative temperature and sound velocity profiles of the channels investigated. Further details on the hardware and sensitivity settings are contained in [64].

c. *Support Equipment*

Numerous items of miscellaneous support equipment were necessary to complete experimental operations. The figure below illustrates both the current communications configuration of ARIES and the envisioned concept of operations during possible future work with Remus or Crawler vehicles [25].

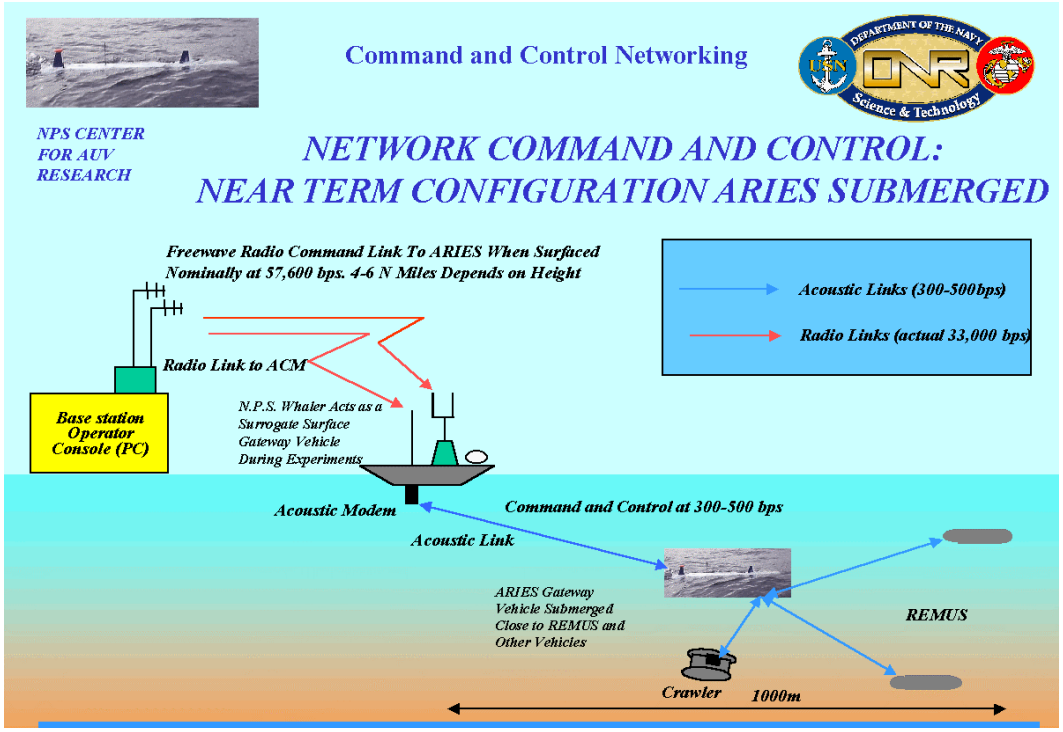


Figure 22. Typical Operating Configuration

All open water experiments were completed in the Monterey Bay. The Command Post trailer was located on the Monterey municipal wharf. The trailer houses computers and communications equipment. Two laptop computers were used, one for AUV command and control and the other for communication with the topside modem. Each computer sent its signals from the Command Post to the antenna relay system deployed aboard the 22 foot Boston Whaler support boat via RF Freewave modem. The Boston Whaler served as the relay for direct RF control of the ARIES (to communicate when surfaced) and the support platform for the topside acoustic modem (to communicate when submerged).

C. SOFTWARE DEVELOPMENT

Communication between the ARIES and topside modems necessarily involved development of computer code to interpret the information received and perform the subsequent action required. This section begins with a brief explanation of the overall computer architecture in the ARIES and then discusses the creation of and modifications to portions of the computer code specifically related to the modem and modem processes.

1. Aries Computer Architecture

Specifications for the dual computer system hardware and software used in ARIES are given in [36] and the operating procedures to conduct missions are well documented in [62]. To recap pertinent highlights, the hardware system consists of two 166 MHz Pentium computers, each with a 2.5 GB hard drive. The computers communicate with each other and an exterior LAN through TCP/IP sockets. One of the computers is designated QNXE (Execution) and controls autopilot execution and the modem process. The other computer is designated QNXT (Tactical) and runs all of the sensor processes critical to navigation.

The distibution of the software architecture is shown in Figure 23. Both computers use the QNX (Pentium II processor) real time operating system. The dual computer architecture allows equitable balancing of the computational workload. The sensor processes that run on the "Tactical" computer (QNXT) are asynchronous and collect data at the rate appropriate to the sensor concerned. The values are then integrated into the navigation process at 8 Hz, transferred to QNXE at 16 Hz, and ultimately are used to effect control on the "Execution" computer (QNXE). In summary, QNXT gathers and synthesizes pertinent sensor based data and QNXE uses the information to command autopilots directly for the servo control level. All processes are written in C language. Intraprocessor communication in each processor is accomplished with specially configured shared memory structures. Inter-processor communication is accomplished using thernet socket links. All processes run to meet guaranteed time lines which is an essential ingredient guaranteeing stability for real-time control.

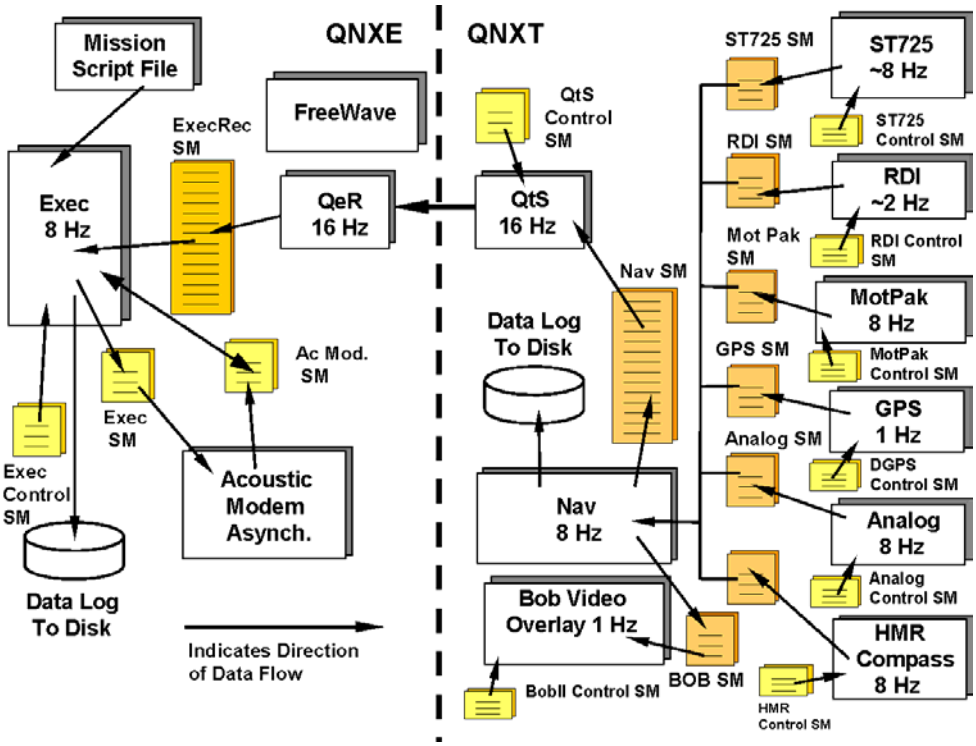


Figure 23. Dual Computer Software Architecture (SM is Shared Memory)

Although minor code alterations were implemented in some of the QNXT processes (including *Analog*, *HMR* compass, and *Nav*), the majority of the modem code development was done in QNXE. Two processes of special interest in QNXE are the mission execution process *Exec* and the modem process *fm*.

2. Changes to Execution Code

The mission plan is a series of commands formatted in a preprogrammed script file, *script.d*. The execution process *Exec* parses mission script commands sequentially, taking the desired control action required by each command. The control actions generated by each line of the script file can vary from simply setting an operational parameter (such as depth) to completing a complicated multi-legged search mission (with multiple parameter changes). A sample script file is contained in Appendix B.

One of the most general and useful mission profile types run on ARIES is invoked by the script line "USE_CTE_WAYPOINT_CONTROL". This command allows for

multiple segments during a mission, up to a maximum mission length of ninety-nine separate tracks or legs. In an input file to the *Exec* process named *Track.out*, each leg is pre-defined by waypoints and set points for speed, depth, and other control parameters. The *Track.out* file is read one time at the beginning of each Waypoint Control mission and stored into memory for subsequent use by the *Exec* process. A sample *Track.out* file and a description of the parameters is in Appendix C.

The executable object (*Exec* process) is created by compiling three interrelated source programs written in C language: Exec.h, Execf.c, and Exec.c. This modular approach to code development was used to create the executables for most of the processes and certainly for the more complex among them. Modular programming enhances future code development efforts by being clear, concise, logical, and efficient. Exec.h is a standard header program containing library function declarations specific to the execution process. Execf.c serves as the repository for most of the functions used in the compiled program. Also in Execf.c are the bit level assignments for the digital-to-analog cards (DAC) and the structure assignments for shared memories related to the execution process. Exec.c is the "top level" part of the combined code, the piece that identifies the actions to be taken when a script command is parsed.

Exec interfaces with the modem control software (discussed in the next section) through shared memory segments. The execution process reads the FAU shared memory segment at the control rate of 8 Hz and checks for a flag that indicates a new command has been received by the modem. If true, *Exec* sequentially compares the new command string with those recognized to be valid and selects the relevant option. The process then changes the desired parameter or initiates the required action. Finally, *Exec* resets the data flag prior to leaving this segment of the code. An excerpt of the Exec.c code that interfaces with the modem is in Appendix D.

Parameters that can currently be altered at any time during the mission are the commanded depth, commanded altitude, and the control mode used (depth or altitude). Changes to these parameters will override any previous segment leg guidance and will remain in effect until changed via a new modem command. Parameter alterations

determine which control actions will be taken at the servomotor level. For example, consider the 3rd mission segment or track leg excerpted below from Appendix C:

580.00 330.00 2.75 2.75 0 8.0 1.5 0 25.0 10.00 45.0

Refering to the excerpt above, the control mode (0 in the 5th column) is set for depth control. Therefore, ARIES will use flight depth control and control to the set point of 1.5 meters below the surface (1.5 in the 7th column). If a modem command was sent to change the control mode to altitude control (1 in the 5th column), ARIES would immediately begin using the flight altitude control mode rather than depth control and control to the set point of 8.0 meters above the bottom (8.0 in the 6th column). Similarly, the control mode could remain at its current setting but be required to control to a different set point if a new depth or altitude value was sent via modem command. Samples of this behaviorial modification and ARIES' reponse to commands will be examined in a later chapter.

Code changes were also made to effect an appropriate change to the mission parameters for all subsequent legs once the control mode change was accepted. In this way, a one time mission change to go to altitude control would be maintained in all subsequent tracks of the mission.

While altitude and depth control changes were regularly used in the experimental validations, it should be noted that changes to any mission specification parameter could also be effected. For instance, advancing the mission waypoint track number could also be done. Missions could be advanced or repeated, if necessary.

The final command that can be received from the modem and recognized by the *Exec* process differs from the three previous in that there are no alterations or overwriting of mission segment parameters before action is taken. An "abort" sent via modem is immediately acted upon once received and acknowledged by ARIES, effectively terminating the mission in an orderly manner.

3. Development of Modem Code

The acoustic modem serves as the only means of communication with ARIES once it has submerged. The *fm* process provides the onboard interface necessary for all modem operations. The capability to effectively communicate with a working AUV is of

paramount importance if tactical considerations require alteration of the mission parameters (i.e. mission re-directs). Monitoring vehicle status is an added benefit of having a reliable acoustic modem installed.

Similarly to the *Exec* process, the *fm* process was designed modularly and consisted of two compiled source C language programs. Higher level actions and many functions are delineated in fm.c and the majority of the data structures for subfunctions are resident in fmf.c. A schematic of the pseudo control code for the compiled *fm* process is found in Figure 24. An excerpt of the fm.c source code is in Appendix E and message format rules unique to the FAU acoustic modem are in [54].

a. The fm Process

When the *fm* process is invoked, the onboard acoustic modem receives instructions from QNXE to set intermodem communications configuration parameters (e.g. levels for power output, receive gain, etc.) Also delineated are the bit packing rate per transmitted pulse and the error correction coding method. Once set, these parameters may not be altered until mission completion. Changes to the configuration require source code editing and recompilation.

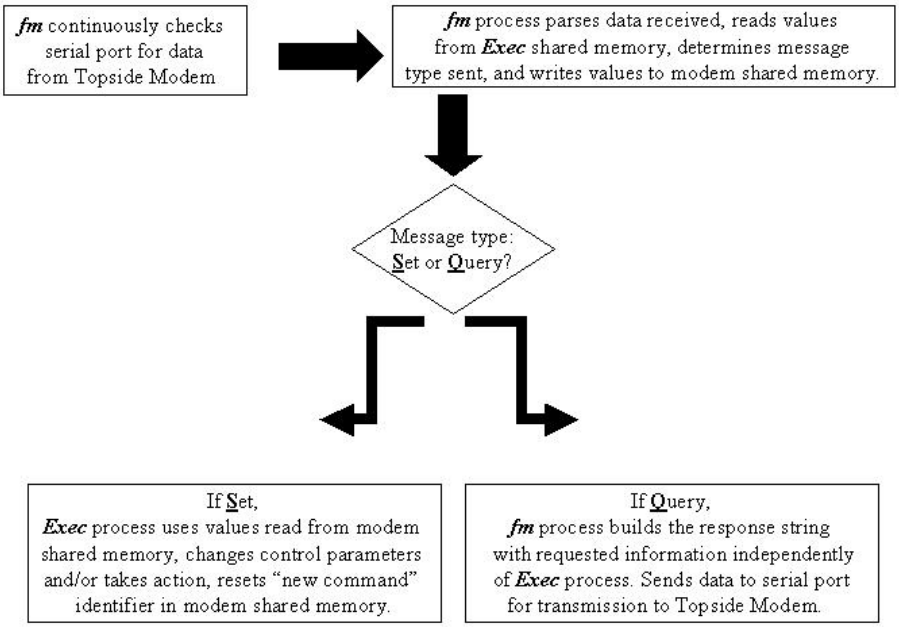


Figure 24. Pseudo Control Code for Modem Process *fm*

After initialization, the ARIES modem process continuously monitors the RS232 serial port for incoming messages from the external Topside Modem. When input is received on the serial port, the process parses the information. Prior to acting on the information received, the process retrieves the current state of ARIES by accessing the execution shared memory location. Key variables accessed include the vehicle's position, active control mode, actual and commanded depth, actual and commanded altitude, and heading. Current variable values may be required in the next part of the process, depending on the type of message received.

Only two message types are recognized as valid by the *fm* process: "Set" messages (S) and "Query" messages (Q), the fifth character after the "$". A typical message sent from the Topside Modem has the format:

$FAUM**S**, COMMAND TYPE, VALUE

OR

$FAUM**Q**, COMMAND TYPE

Once message type is determined, the process then completes a series of string comparisons to determine the validity of the command type. If the command type is valid, the subsequent actions taken by the *fm* process diverge for "Set" and "Query" messages. "Set" messages, except for the abort procedure described in more detail in a later section, cause the new value received to be written to the modem shared memory for ultimate use by the *Exec* process. "Query" messages initiate the building of formatted strings within the *fm* process which are then sent to the Topside Modem through the serial connection.

b. Description of Command Types

There are currently four command types associated with the "Set" message type and each must be exactly five characters in length. The actively used command types are CMODE, DEPTH, ALTIT, and ABORT. As briefly examined in a previous section, these command types directly effect the *Exec* process and will cause a change to the vehicle behavior. The CMODE command type defines the control mode for the mission, either depth control if the value is 0 or altitude control if the value is 1. The DEPTH command type allows the operator to alter the commanded control depth

value for the vehicle at any time during the mission. Similarly, the ALTIT command type allows alteration of the value for commanded control altitude above sea bottom. ABORT, the final command type, differs slightly from the previous three commands. No value is passed by ABORT but the vehicle proceeds directly into abort procedures and ends the mission.

The primary usefulness of the "Query" message type is as a means for the Topside operator to assess vehicle status and monitor key parameters in near real-time during the mission. An important difference between "Query" and "Set" message types is that "Query" messages are completely independent of the *Exec* process. Actions taken by the vehicle to answer the query are done entirely within the *fm* process using the most recent vehicle states read from shared memory. While the "Query" message type currently has six recognized command types, only three will be discussed: POSIT, WAYPT, and CMODE. The remaining three query message command types (SMALL, MIDDL, and LARGE) were developed primarily for testing. As was true for the "Set" message type, each command type must be exactly five characters in length.

The POSIT command type requires the vehicle's *fm* process to build and return, to the external operator, a response string with the current vehicle position, heading, depth, and altitude. Although the string size sent via modem is relatively long (about 30 characters in length), knowledge of the returned parameters enables the operator to have a complete three dimensional visualization of the vehicle's progress in the mission.

The WAYPT command type generates the shortest returned string, about 15 characters in length. The response from the vehicle gives only the position and the current waypoint or leg of the mission. While brief, this information allows the operator to visualize a plan view of the mission and to have a good estimate of the remaining mission duration.

The CMODE command type used in "Query" message types differs from the CMODE command type used in "Set" messages. In a query, CMODE returns information on the current control mode (altitude or depth) and also includes values for commanded and actual altitude as well as commanded and actual depth. The returned

string is about 20 characters in length and has two important uses. Primarily, it is essential to verify the current operational control mode (depth or altitude) prior to sending a new set point value. For example, it would be ineffective to send ARIES a "Set" message commanding a new depth if the vehicle was currently operating in the altitude control mode. In that particular mode, only a new commanded altitude would elicit a vehicle response. A secondary use of the CMODE command is as a diagnostic tool to measure ARIES' control performance. Depending on the active control mode (altitude or depth), comparisons can be made between the commanded and actual altitude or depth. System failure can be assumed if the measured value of altitude or depth exceeds a reasonable bound around the commanded altitude or depth. System failure would require invoking abort procedures.

c. Emergency Abort Command

Addition of an emergency mission abort procedure was an extremely important revision to the *fm* process. As previously discussed, sending ARIES a "Set" message type with the "ABORT" command type causes the mission to terminate in an orderly manner. However, this presumes that operating conditions are normal and that the *Exec* process is still functioning. If the *Exec* process becomes corrupted for any reason, it is unable to properly read the set command from the modem shared memory or carry out the actions directed. Furthermore, the thrusters and all control surfaces become unresponsive to safeguard behaviors inherent to the code and remain "locked" in the state that existed immediately prior to the process corruption. This condition is extremely hazardous and could result in severe damage to or loss of the vehicle. For example, if the process failed while the planes were in a dive position and the rudders were locked in a seaward direction, the ARIES would continue to dive until it exceeded its crush depth. Therefore, the emergency abort procedure was developed to serve as a fail-safe method to save the vehicle.

Simply stated, the emergency abort procedure took direct control of and stopped all power to the thrusters. While other emergency schemes were discussed and investigated, this method was selected for its simplicity and effectiveness. The ARIES has slightly positive buoyancy so once the thrust is removed, the vehicle will rise to the surface regardless of the plane and rudder orientation. Source code changes to fm.c were

required to implement the emergency abort procedures in the compiled *fm* process. The new functions added to the *fm* process directly addressed the digital-to-analog card and controlled the thrusters, a capability previously only possible in the *Exec* process. Effectiveness of the new software was validated during field testing. ARIES performed normal ABORT procedures when the *Exec* process was running and activated emergency procedures during a test where the *Exec* process was corrupted. An excerpt of the emergency abort procedure from the fm.c source code is in Appendix F.

D. CHARACTERIZATION OF TRANSMISSION DELAY TIMES

Often times, the expected theoretical performance of hardware and sensors that are developed and individually tested in a relatively benign environment does not directly correlate to actual performance of those that become part of a dynamic operating system. Therefore, once the computer code for all testing was developed, it was necessary to begin performance tests with the modem as a part of the AUV system to determine the operating characteristics and quantify the bounds of parameters such as the delay times that could be expected for various combinations of encoding and modulation. Because one of the key performance parameters of any control data link is latency (time delay) to send the data, this parameter was studied first.

1. Experimental Procedure in Laboratory and Bay

The objective of the first series of experiments was to characterize the message delay times that could be expected due to hardware constraints. Therefore, initial testing was done in the laboratory with the topside and ARIES modem transducers placed in a 1 meter fresh water test tank. Since the nominal speed of sound in seawater is approximated as 1500 meters per second, the reduced distance between transducers made any time delay contribution due to the acoustic sound speed negligible. Multiple versions of the fm.c program for the *fm* process were created, each to appropriately configure the coding and modulation combination in the ARIES modem prior to the corresponding test. Query messages were sent from the topside modem to the Aries modem and the response times were logged and tabulated. In the laboratory, it was determined that testing of the coding schemes would only be possible using slower modulation, modes 3 and 4. Responses from the ARIES with the higher rate, tightly packed waveforms used in

modulation modes 1 and 2 were not successful. Mode 2 was sporadic, even at low transmit power levels, and mode 1 was not able to be decoded successfully.

The series of experiments was continued in the seawater at the Monterey Harbor piers. ARIES was launched and securely lashed to the dock. The topside modem head was placed in direct line-of-sight with the ARIES transducer at a range of no more than 5 meters. The various configurations of modulation and coding were tested and, as determined in the laboratory, modulation modes 1 and 2 were not successful, probably due to the higher bit packing rate in each baud period. Four combinations were selected for extensive investigation using one consistent message, a position query. Mode 4, the slowest but expected to be the most reliable modulation method, was tested using block coding and concatenated block and convolutional coding. Mode 3 modulation was also tested with both block and concatenated coding and was anticipated to perform at a higher speed.

2. Results

More than 1300 query reponses were received during seawater testing on the four combinations of the two coding and two modulation methods selected. Results were logged, tabulated, and processed to produce bar charts of normalized occurrences versus delay times for successfully received messages. The results are graphically displayed below. Note the trend in all histograms for the preponderance of transmission times to occur early then taper with a relatively steep decay. The additional occurrences, somewhat periodic in nature, are the result of software instigated retransmissions after an initial modem communications failure.

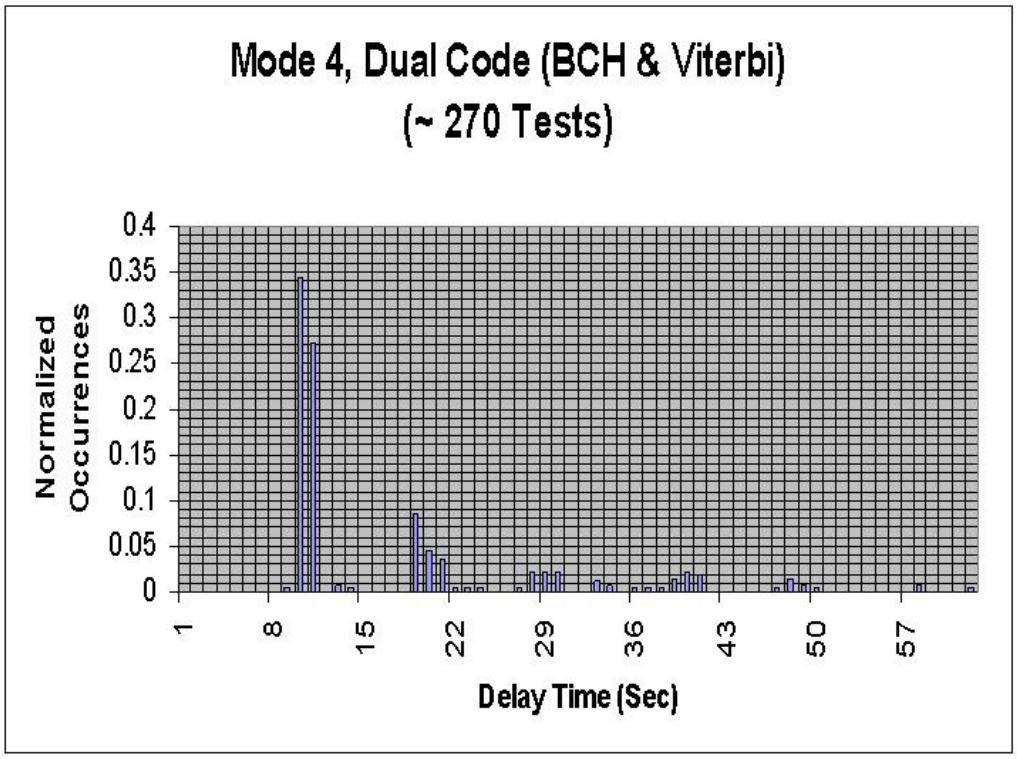


Figure 25. Time Delay, Mode 4 Dual Code

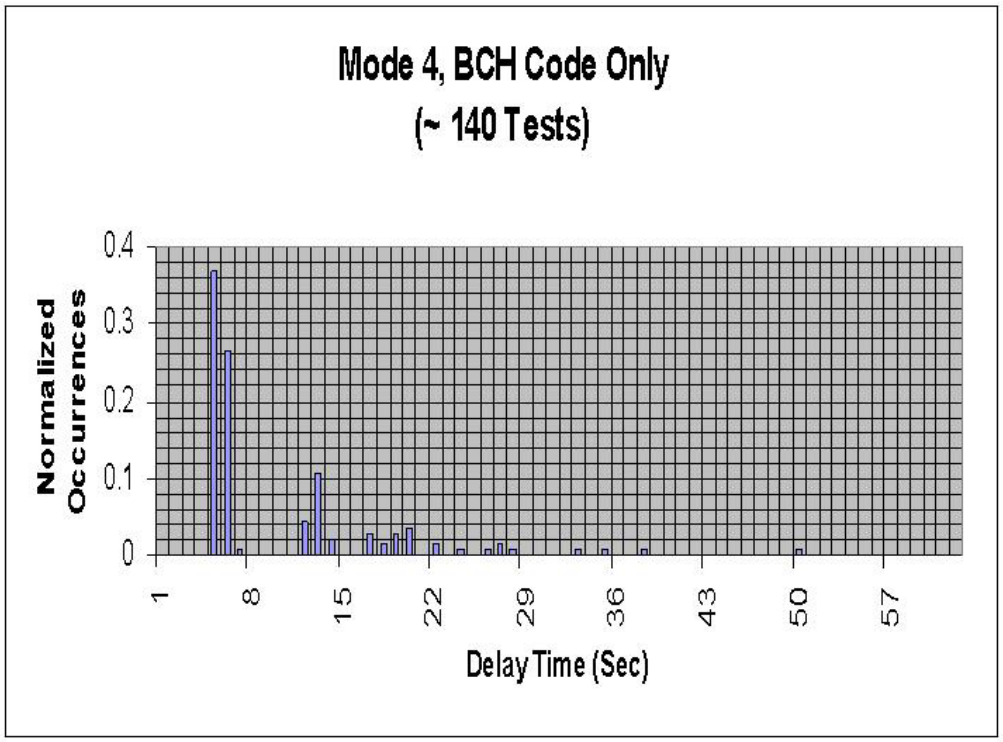


Figure 26. Time Delay, Mode 4 BCH Block Code

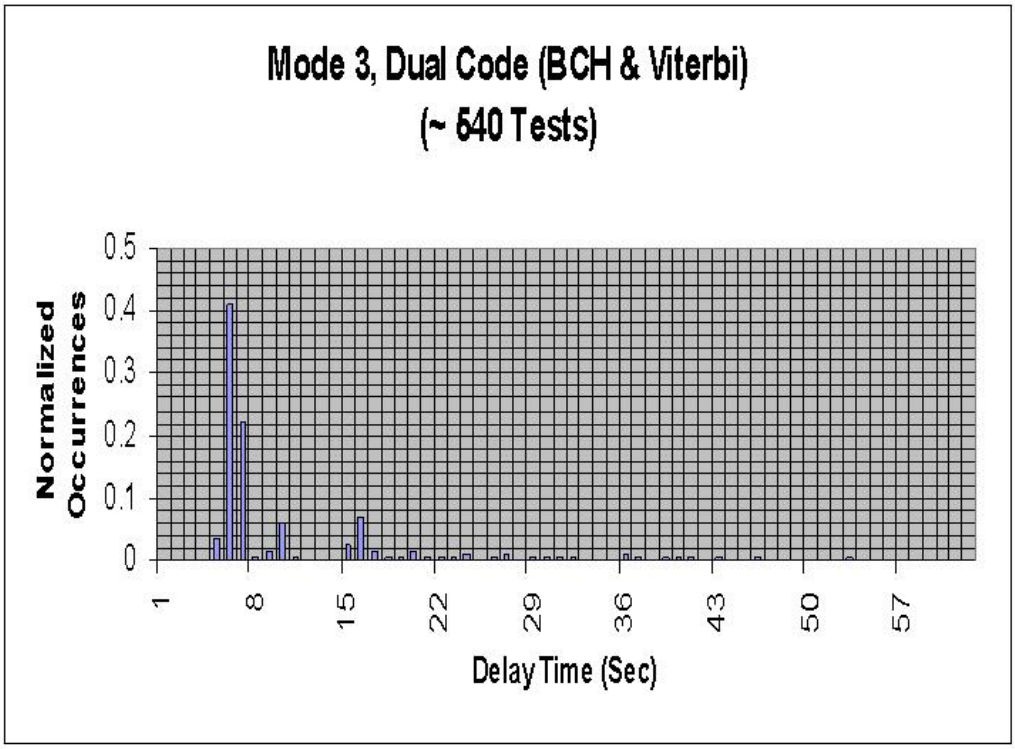


Figure 27. Time Delay, Mode 3 Dual Code

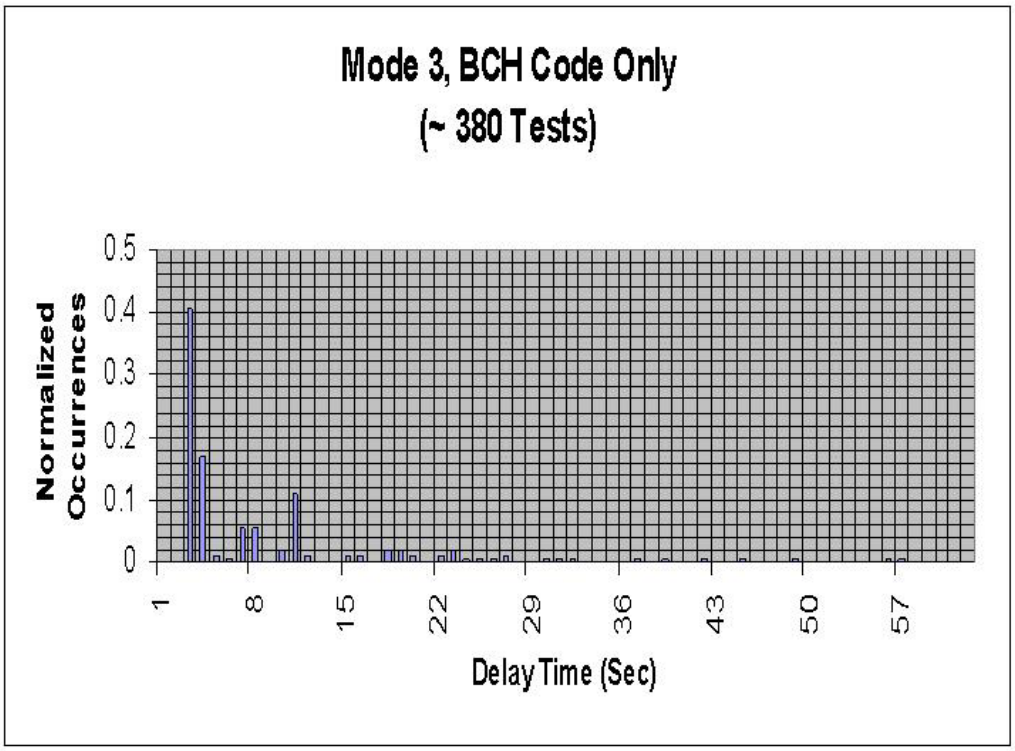


Figure 28. Time Delay, Mode 3 BCH Block Code

3. Conclusions

Several interesting observations can be made from the data, including quantification of actual versus theoretical delay times expected and assessment of the relative reliability of successful communication when operating in any of the four modulation/coding combinations. Theoretical delay times were calculated in each case based on the fact that the query message used was 40 characters long, the equivalent of 320 bits. Since there are 256 information data bits per frame and incomplete frames are padded with zeroes, each received message was two full frames long. Additionally, there is a 0.677 second channel access delay built into the firmware for detection of an idle channel prior to transmission for collision avoidance. Weighted average overall delays were calculated for each configuration by summing the products of individual delay times multiplied by the number of occurrences then dividing that sum by the total number of occurrences.

Messages received using dual encoded mode 4 modulation had the longest inherent delay, as expected, with a weighted average overall delay of 17.41 seconds for a series of more than 270 tests. About 61.5% of the successful replies were received within 10.44 seconds, as can be seen from Figure 25. Also note from the figure that about 16% more of the total replies were received in the first retransmission attempt (clustered around 20 seconds). The theoretical transmission time is 9.98 seconds based on:

$$\frac{256 \text{ data bits/frame}}{55 \text{ data bits/second}} = 4.65 \text{ seconds/frame}$$

$$40 \text{ X } 8 = 320 \text{ data bits in message } \therefore \text{ 2 frames required}$$

$$2 \text{ frames X } 4.65 \text{ seconds/frame} = 9.30 \text{ seconds}$$

$$9.30 \text{ seconds} + 0.677 \text{ seconds (wait time)} = 9.98 \text{ seconds}$$

The results of mode 4 modulation with only block (BCH) coding are shown in Figure 26. In a series of over 140 tests, about 63.3 % of the messages were received in an average time of 5.42 seconds with an additional 17% received after the first

retransmission. The weighted average delay for the entire series of responses was 10.11 seconds. The theoretical transmission time was 5.34 seconds, computed similarly to the mode 4 dual encoded case. The major difference was that 110 data bits/second are sent in this configuration so there are only 2.33 seconds/frame required for each of the two frames.

Mode 3 modulation with dual block and convolutional encoding also has an effective data rate of 110 bit/second, so its theoretical transmission time calculation is identical to the mode 4 block encoded case, 5.34 seconds. However, the actual average time for 63.5% of the 540 tests was about 6.35 seconds with an overall weighted average delay of 11.03 seconds. The long overall average can be explained by the wide dispersion of reponse occurrences as seen in Figure 27.

Finally, we have the configuration expected to have the shortest transmission delay, mode 3 modulation with block (BCH) encoding, shown in Figure 28. 57.9% of the responses were received within an average delay of 3.29 seconds during over 380 tests. The weighted overall delay time was 8.51 seconds. Note that about 83% of the responses had been received within 11 seconds. The theoretical transmission delay was 3.00 seconds, based on a data rate of 220 bits/second hence 1.16 seconds/frame.

The chart below summarizes the data from time delay characterization tests on the modem installed in ARIES.

	Mode 4, Dual	Mode 4, BCH	Mode 3, Dual	Mode 3, BCH
Average Delay (sec) 1st Transmission	10.44	5.42	6.35	3.29
Theoretical Delay (sec) 1st Transmission	9.98	5.34	5.34	3.00
% Received 1st Transmission	61.54	63.31	63.54	57.85
% Received By End of 1st Retransmission	80.58	81.29	75.32	69.90
Weighted Overall Average Delay (sec)	17.41	10.11	11.03	8.51
Total Tests	~ 270	~ 140	~ 540	~ 380

Table 3. Summary of Modem Delay Time Characterization

In summary, modulation mode 4 has longer delay times yet better reliability than mode 3 modulation with the same encoding scheme, where reliability can be measured by the percentage of messages received by the end of the first retransmission time. Actual transmission delays closely mirror the performance anticipated from theory with less than 10% error except in the case of dual encoded mode 3 modulation, which varies from theory by about 16%. It should be noted that the shortest delay time possible, even assuming a one frame message, would be 1.84 seconds when using mode 3 modulation and block encoding. Finally, for the best compromise between delay time and reliability, mode 4 modulation with block encoding appears to be the method of choice. Its performance exceeds that of the dual encoded, higher bit rate mode 3 modulation; the delay time is less and the reliability of receipt is greater.

E. CHARACTERIZATION OF EFFECTIVE RANGE

In addition to analyzing the delay times that could be expected with the FAU modem installed in the ARIES, determination of the effective transmission distances was an important parameter to quantify. As previously discussed, the shallow water environment is well documented as an adverse channel for acoustic communications.

However, acoustic control of the AUV and exchange of data between modems is predicated on a successful and open line of communication. Therefore, it was necessary to experimentally validate the effective range of modem configurations in typical operational environments. In the very shallow waters (15 meters), multiple reflections from surface and bottom limit transmit distance. Rough weather further deteriorates range through increased surface loss.

1. Discussion

Since range prediction is not possible with the present state of acoustic modeling, three separate series of experiments were needed to characterize modem effective range performance when integrated into the AUV system running all sensors and processes. The objective of the first series, at close range, was to investigate ARIES ability to receive and reply to messages at different altitudes above the sea bottom. The second and third series were investigations to bound the limits of the communications range between the topside controller and the AUV in different shallow water channel geometries. In all tests, background noise readings were taken and recorded. Additionally, in the channel geometry studies, representative CTD casts were collected to measure the sound speed profile that can be expected in the shallow water acoustic channel.

2. Short Range Response (Tranducer Depth Effects)

An AUV may be tasked to perform missions in any portion of the water column during shallow water operations. To ensure that this capability would be possible despite the adverse multi-path expected, it was necessary to investigate and characterize topside-AUV communications success rates at different operating depths with various combinations of modulation and encoding.

As was previously done in the time delay characterization study, the modem configurations selected for evaluation were modulation modes 4 and 3, each with block (BCH) and concatenated (BCH and convolutional) coding schemes. Recall that mode 4 modulation with dual coding was the slowest configuration while mode 3 modulation with only block coding had shorter delay times. During each data collection period, the ARIES, with all integrated sensors active, was placed on a multi-leg mission using the vertices of a 30 meter square as waypoints at a horizontal distance of no more than 150 meters from the submerged topside transducer. The topside tranducer head depth was

held constant at 1 meter while the ARIES transducer depth was varied. Depending on the stage of the tide, total water depth was approximately 15 meters. To negate the tidal variations, altitude above bottom was used as the reference rather than depth below the surface. A minimum of 10 communications attempts for each of the four configurations were made at altitudes above bottom from 2-12 meters in two meter increments. Daily assessments of the background noise in the signal frequency spectrum (19-28 KHz) were recorded, with values ranging from 90-111 dB. Wave heights over the course of testing varied from 1-3 feet.

a. Results

The figures below summarize the more than 240 queries that were attempted during seawater testing on the four combinations of the two coding and two modulation methods selected, each configuration tested at six different altitudes, ranging from 2-12 meters above bottom. Message attempts and receipts were logged, tabulated, and processed to produce bar charts of communications success as a function of altitude above bottom and modem configuration. In each case, results are first presented showing the specific tabulated data then, for ease of comparison, are displayed as percentages.

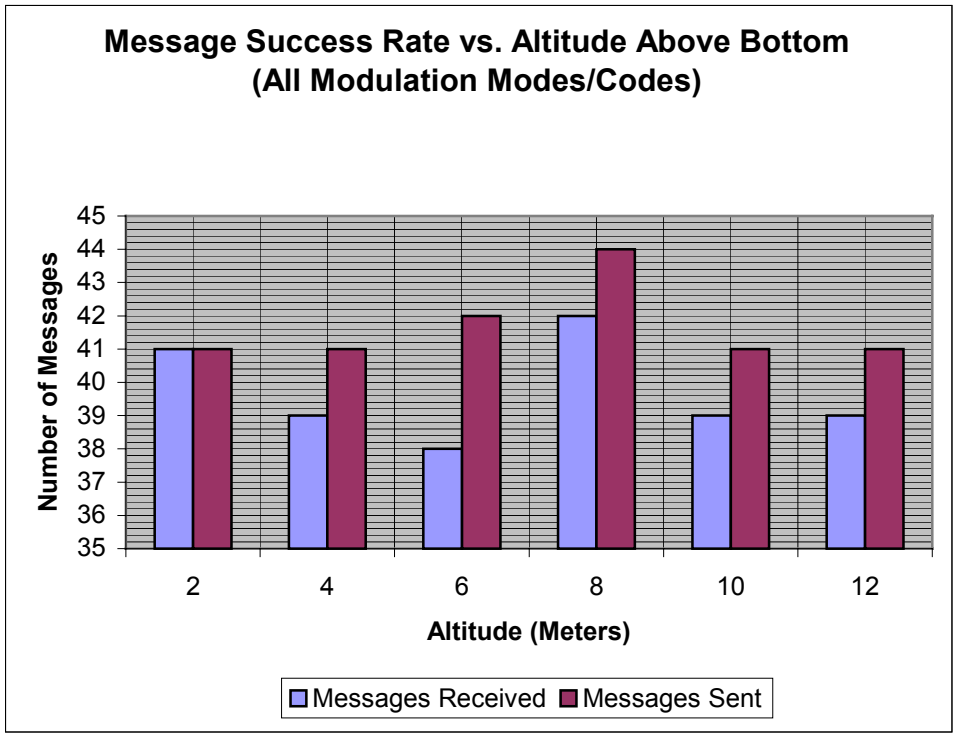


Figure 29. Message Success Data (Altitude Effects)

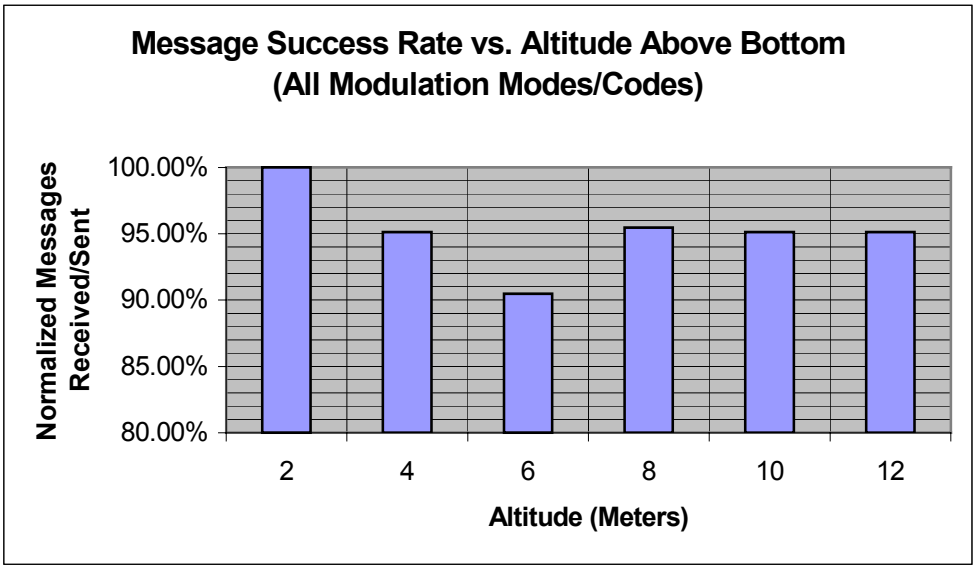


Figure 30. Message Success Rate (Altitude Effects)

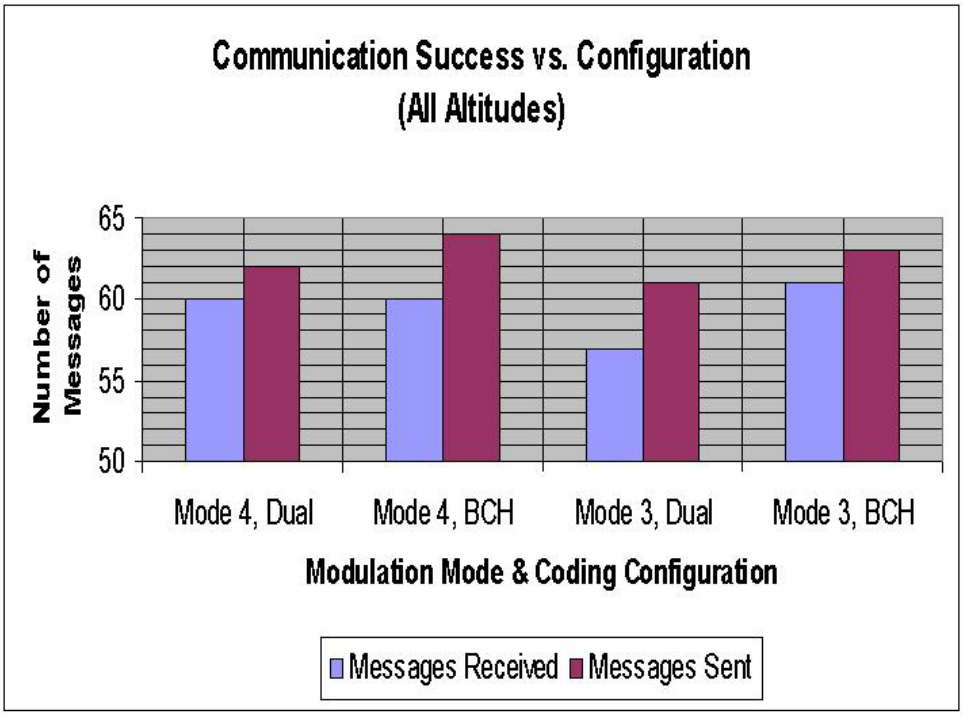


Figure 31. Message Success Data (Configuration Effects)

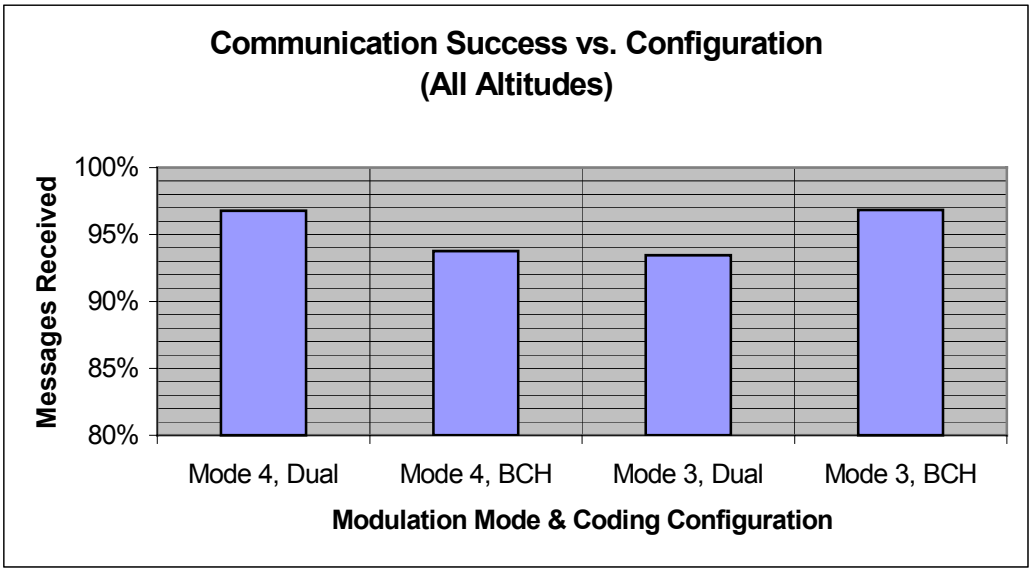


Figure 32. Message Success Rate (Configuration Effects)

b. Conclusions

The data in this series of tests quantified vehicle communications performance at different altitudes above bottom for various modem configurations. Unlike the time delay characterization study, this (and all further experimental characterizations) involved dynamic testing of the AUV with all sensors and processes active. The AUV mission plan, essentially a 30 meter circle, was selected to present a broad range of aspect ratios between the AUV and topside modem transducers, thereby providing a "worst case" scenario for the evaluation.

One of the more interesting observations that can be made from Figures 29 and 30 is that the messages were most successfully received when the vehicle was operating 2 meters above the bottom (100%). The trend from Figure 30 shows that the least success, regardless of the configuration, occurred near the mid-water depth at a six meter altitude. Multipath reflections may explain this trend. Despite the relatively short horizontal range tested, mid-water column depths would still get reflections from both the surface and the bottom, leading to greater signal interference and a lower success rate. However, when operating near the bottom, the vehicle transducer is shielded or baffled somewhat by the AUV body, thereby reducing the strength of surface reflections.

Figures 31 and 32 depict the variations over all altitudes for the four combinations of modulation and encoding. All configurations performed reasonably well with over 90% success. However, mode 4 modulation with dual coding and mode 3 modulation with only block (BCH) coding had the best results over all altitudes. For this reason, among others, dual coded mode 4 modulation and block coded mode 3 modulation were selected as the configurations for the remaining characterization studies. The chart below summarizes the data from altitude effect characterization tests.

	Mode 4, Dual	Mode 4, BCH	Mode 3, Dual	Mode 3, BCH	Total	% Success
Alt=2 Meters (Received/Sent)	10/10	10/10	10/10	11/11	41/41	100%
Alt=4 Meters (Received/Sent)	10/10	10/10	9/10	10/11	39/41	95%
Alt=6 Meters (Received/Sent)	9/10	10/11	9/10	10/11	38/42	90%
Alt=8 Meters (Received/Sent)	11/12	11/12	10/10	10/10	42/44	95%
Alt=10 Meters (Received/Sent)	10/10	9/11	10/10	10/10	39/41	95%
Alt=12 Meters (Received/Sent)	10/10	10/10	9/11	10/10	39/41	95%
Total	60/62	60/64	57/61	61/63		
% Success	97%	94%	93%	97%		

Table 4. Summary of Modem Altitude Effects Characterization

3. Parallel Geometry (Constant Depth Channel)

In addition to the capability to successfully communicate anywhere in the vertical water column, successful acoustic mission control in the shallow water environment is very dependent on the limits or bounds of the horizontal communication range.

Expecting an adverse channel with large multi-path transmission losses, investigation of two common underwater geometries was completed to characterize the limiting range and the associated success rates of horizontal topside-AUV communications using the two extreme combinations of modulation and encoding. The first channel studied was a parallel geometry or constant depth channel.

Demonstrating the highest success rates in altitude characterization tests, the modem configurations selected for evaluation of the constant depth channel were modulation mode 4 with concatenated (BCH and convolutional) and mode 3 with block (BCH) coding schemes. Additionally, the two configurations chosen covered the broadest spectrum of speeds or bit rates; mode 4, dual code, operated at the lowest (55 bits/second) and mode 3, BCH code, at the highest (220 bits/second). Similar to the altitude tests, ARIES, with all integrated sensors active, was placed on a multi-leg mission using the vertices of a 30 meter square as waypoints. However, unlike the altitude tests, the topside transducer, operating from a support boat, varied the horizontal distance at 500 meter increments out to the maximum range communications could be received and acknowledged. Upon reaching a mark where communication completely failed, the support boat transitted back toward the last good mark in smaller increments until a good link was re-established. The parallel channel water depth was held constant at approximately 15 meters. The topside and ARIES tranducer heads operated in the mid-water column and, for consistency, were at a depth of about 8 meters. Using the maximum tranducer power setting (192 dB), a minimum of 30 communications attempts were made for each of the configurations at each horizontal distance to assess the failure rates and maximum effective range possible. Daily assessments of the background noise in the signal frequency spectrum (19-28 KHz) were recorded, with values ranging from 98-110 dB. Wave heights over the course of testing varied from 1-3 feet. CTD casts were taken. Matlab programs were developed to process and plot the CTD data. A representative plot of the sound-velocity profiles generated during parallel channel testing is shown below.

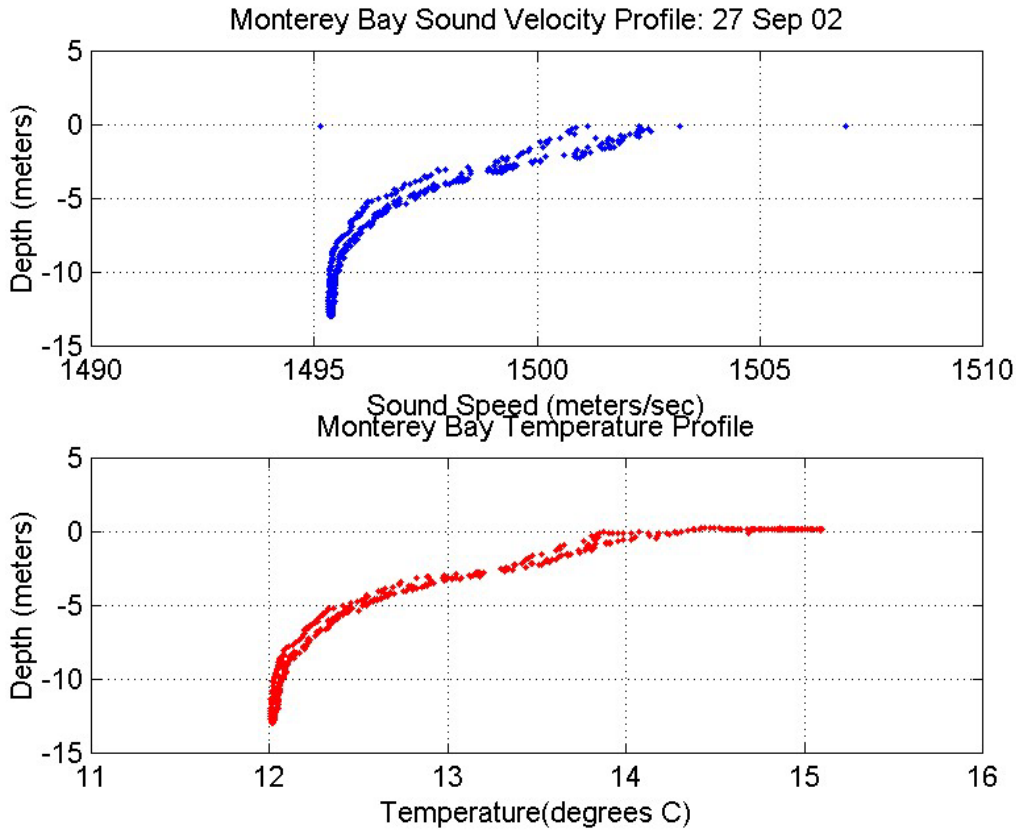


Figure 33. Representative Monterey Bay Sound Velocity Profile

The hysterisis evident in both graphs is a result of slight variations in the values as the instrument is first lowered then raised back through the vertical water column. Temperature has the greatest effect on sound velocity in the shallow water channel. Observe the decrease in temperature at greater depths in this particular profile. The temperature drop translates to decreased sound speed which, in turn, indicates that the sound waves would have the tendency to propagate downward at the time and location of this sample.

a. Results

The figures below summarize the query attempts during seawater testing on the two combinations of coding and modulation methods selected, each configuration tested out to the maximum horizontal range in a constant depth (parallel) channel.

Message attempts and receipts were logged, tabulated, and processed to produce bar charts of communications success as a function of range and modem configuration. The results are first presented showing the specific tabulated data then, for ease of comparison, are displayed as percentages.

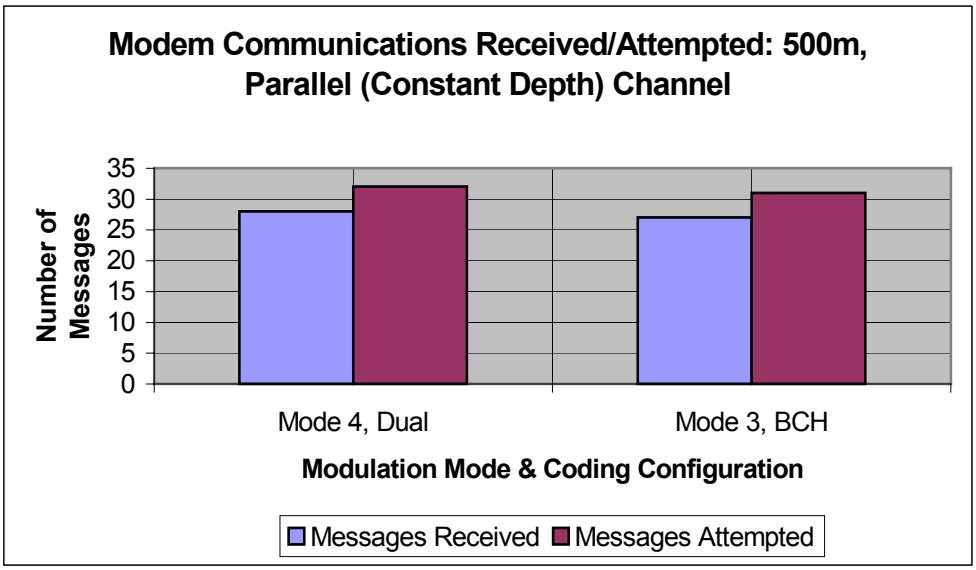


Figure 34. Message Success Data (500 Meter, Constant Depth Channel)

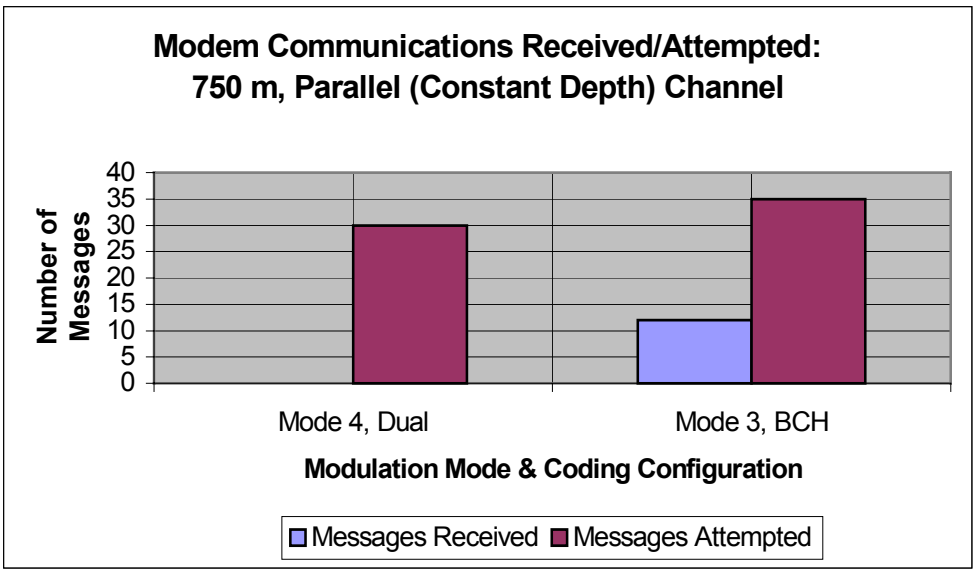


Figure 35. Message Success Data (750 Meter, Constant Depth Channel)

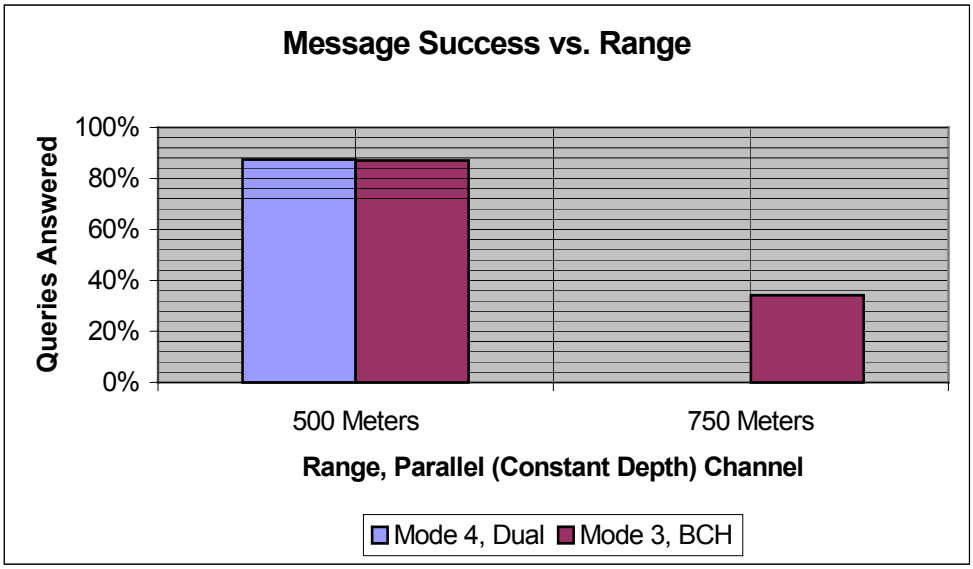


Figure 36. Message Success Rate (All Ranges, Constant Depth Channel)

b. Conclusions

The data in this series of tests quantified the maximum range or bounds of vehicle communications performance in a constant depth, shallow water channel for a dynamic AUV with all processes and sensors active. As in the previous (and all further) characterization studies, the AUV mission plan was selected to evaluate the worst case condition by ensuring there was a broad range of aspect ratios between the AUV and topside modem transducers. Query messages sent to ARIES were either received 100% correctly or not at all, thus no replies were possible for incorrectly received messages. This "all-or-none" feature was a safeguard built into the ARIES software design to minimize the possibility that erroneous commands could alter mission plans or produce unintended consequences.

The most important conclusion from the constant depth tests of the two modem configurations is that the maximum range for effective communication between a tactical controller and an AUV is only about 500 meters in this particular shallow water geometry. Referring to Figures 34-36, it can be seen that message success rates for both the 55 bits/second mode 4, dual and 220 bits/second mode 3, BCH configurations were relatively high at a 500 meter horizontal range (> 85%). However, at the 750 meter

range, the higher speed mode 3 was only about 34% successful and the lower speed mode 4 was completely unsuccessful in responding to any of 30 attempted queries. Comparing results to Table 2. (the theoretical range), mode 4 modulation with dual coding is an order of magnitude lower than the expected 5 kilometers. Mode 3 modulation with block (BCH) coding performed better than mode 4 modulation, but still reached only 30% of its theoretical range of 2500 meters. The table below summarizes the data from parallel (constant depth) channel maximum range characterization tests.

Parallel (Constant Depth) Channel	Mode 4, Dual	Mode 3, BCH	% Success
Range = 500 Meters (Received/Sent)	28/32	27/31	87%
Configuration % Success	88%	87%	
Range = 750 Meters (Received/Sent)	0/30 (Unable to Receive)	12/35	18%
% Success	0%	34%	
Total Received at All Ranges	28/62	37/66	51%
Configuration Total % Success	45%	56%	

Table 5. Parallel (Constant Depth) Channel Range Limit Characterization

4. Perpendicular Geometry (Converging Channel)

The second geometry studied to determine the bounds of horizontal communication range was an acoustic channel perpendicular to the shore, also called a converging channel. Although all evaluation was done in shallow water, the converging channel is defined as any point where the water column height was deeper or exceeded

that in which the AUV is operating. As in the parallel (constant depth) channel study, the goal in this test series was to characterize the limiting range and the associated success rates of converging topside-AUV communications using the two extreme combinations of modulation and encoding.

All test procedures and parameters used to characterize the maximum range limits in the constant depth channel were used in the converging channel study to ensure consistency and make comparison meaningful. Modulation mode 4 with concatenated (BCH and convolutional) and mode 3 with block (BCH) coding schemes were the configurations; the ARIES was placed on an identical multi-leg mission with all systems and sensors active; and the topside transducer was moved by support boat at 500 meter horizontal distance increments out to the maximum range communications, moving back in smaller increments until a good link was re-established when reaching a mark where communication completely failed. The ARIES transducer depth was held constant at 8 meters in a total water depth of approximately 15 meters. The topside tranducer head was also held at about 8 meters below the surface but in overall water column depths up to 30 meters. A minimum of 30 communications attempts were made for each of the configurations at each horizontal distance (except at 1000 meters) to assess the failure rates and the maximum effective range possible. The transducers were operated at maximum power level (192 dB) and background noise in the signal frequency spectrum (19-28 KHz) was no more than 100 dB. Wave heights were under two feet and, once again, CTD casts were taken to sample the temporal and spatial variation of the acoustic channel, indicating a downward reflecting pattern.

a. Results

The figures below summarize the query attempts during seawater testing on the two coding/modulation methods selected, each configuration tested out to the maximum horizontal range in a converging (perdendicular) channel. Message attempts and receipts were logged, tabulated, and processed to produce bar charts of communications success as a function of range and modem configuration. The results are presented in the same format as the constant depth (parallel) channel results: the specific tabulated data followed by an overall figure with percentages of successful communication.

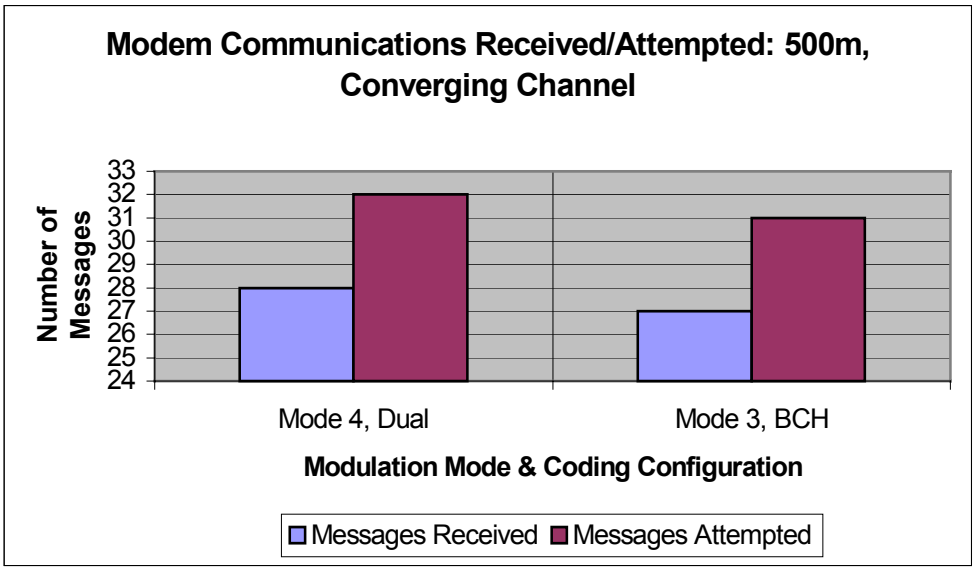


Figure 37. Message Success Data (500 Meter, Converging Channel)

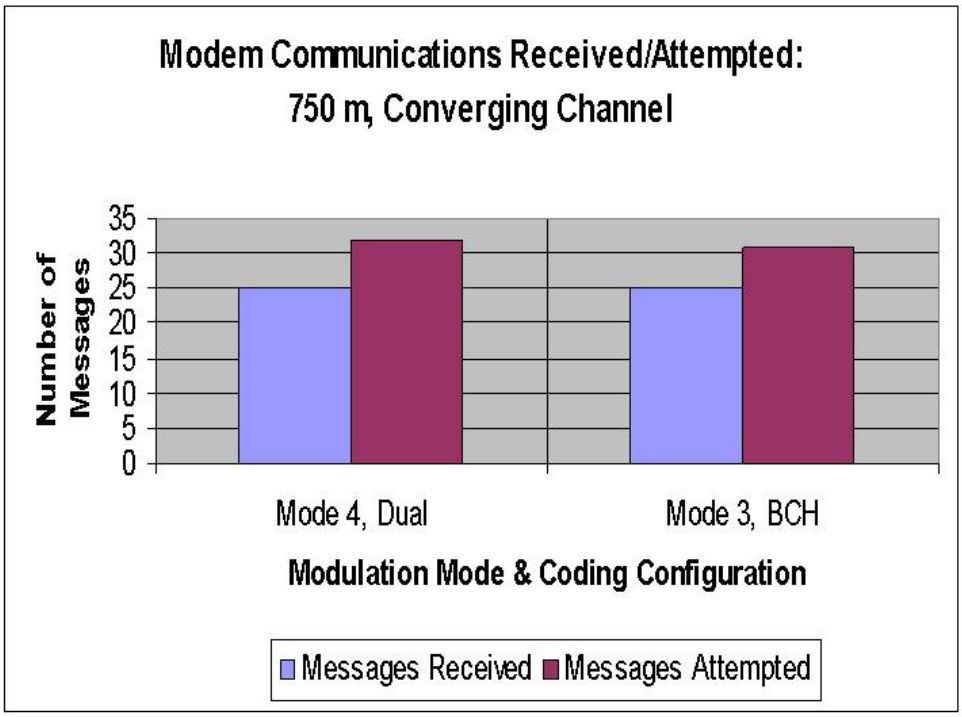


Figure 38. Message Success Data (750 Meter, Converging Channel)

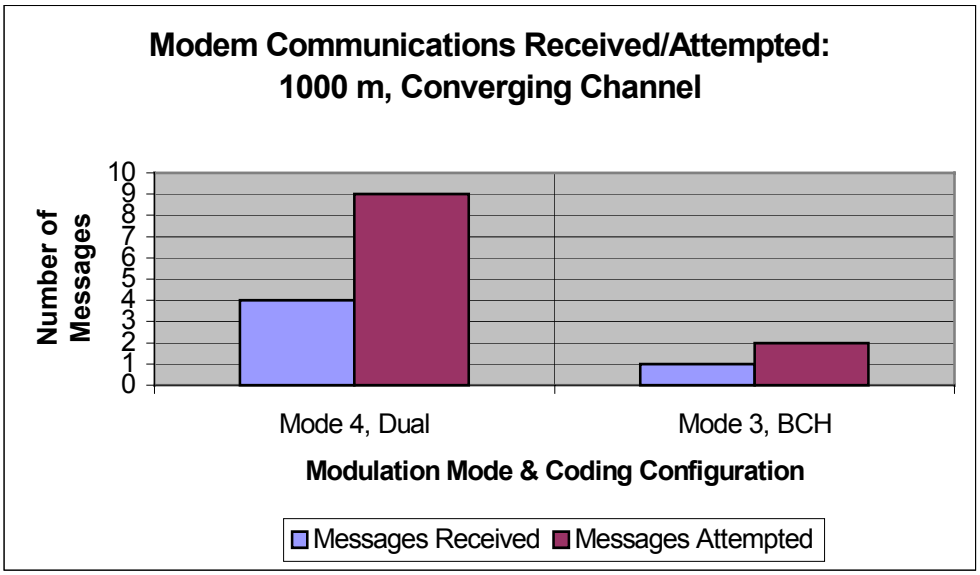


Figure 39. Message Success Data (1000 Meter, Converging Channel)

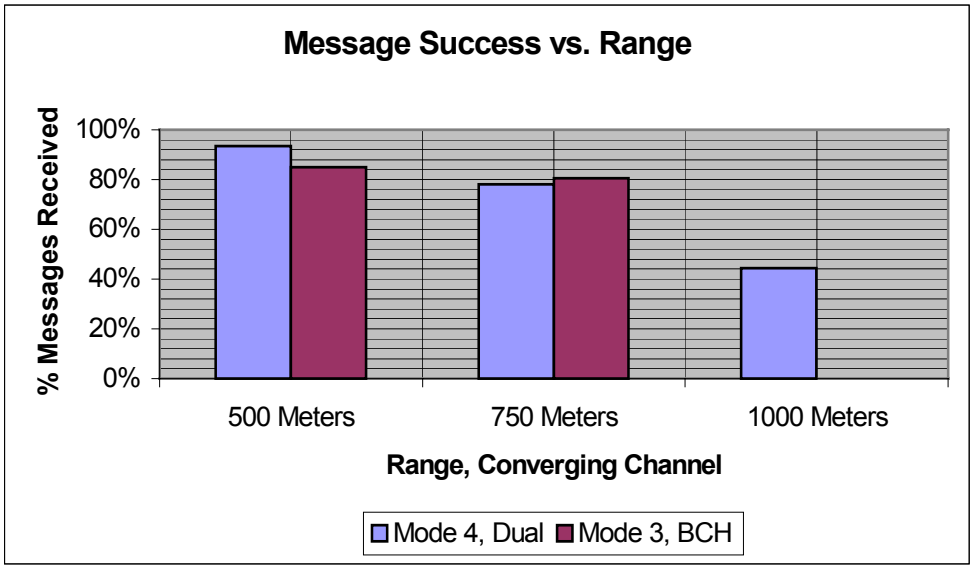


Figure 40. Message Success Rate (All Ranges, Converging Channel)

b. Conclusions

The data in this series of tests quantified the maximum range or bounds of vehicle communications performance in a converging, shallow water channel for a dynamic AUV with all processes and sensors active. As in the other characterization studies, the AUV mission plan evaluated the worst case condition by ensuring there was a broad range of aspect ratios between the AUV and topside modem transducers.

The converging channel tests of the two modem configurations proved that the maximum range for effective communication in this geometry is about 750 meters. Referring to Figures 37-40, it can be seen that message success rates for both configurations were relatively high at a 500 meter horizontal range, 94% for mode 4 modulation with dual coding and 84% for mode 3 with block (BCH) coding. Even at the 750 meter range, communication was still relatively good with about an 80% success rate. However, reliability rapidly deteriorated at the 1000 meter point. The slow but heavily encoded mode 4 modulation dropped below 50% success and the faster mode 3 modulation did not have enough data to be statistically significant. Comparing results to the theoretical ranges in Table 2. once again, mode 4 modulation with dual coding reached only about 20% of expected 5 kilometers and mode 3 modulation with block (BCH) coding still achieved only about 30% of its theoretical range of 2500 meters. The table below summarizes the data from perpendicular (converging) channel maximum range characterization tests.

Converging Channel	Mode 4, Dual	Mode 3, BCH	% Success
Range = 500 Meters (Received/Sent)	29/31	28/33	89%
Configuration % Success	94%	85%	
Range = 750 Meters (Received/Sent)	25/32	25/31	79%
% Success	78%	81%	
Range = 1000 Meters (Received/Sent)	4/9	1/2 *(Not Enough Tests to Be Significant)	*
% Success	44%	*	
Total Received at All Ranges	58/72	54/65	82%
Configuration Total % Success	81%	83%	

Table 6. Perpendicular (Converging) Channel Range Limit Characterization

Recall that successful acoustic tactical mission control in the shallow water environment depends on the limits or bounds of the horizontal communication range, as well as transducer depth in the water column. The results of all three characterization studies were analyzed to determine the overall bounds on acoustic communication using the installed modem system. In the vertical water column, communication was successfully established in over 90% of the attempts despite transducer altitude above bottom. Findings from the two horizontal channel geometries were compared with the conclusion that the constant depth channel was the most restrictive. In summary, it can be concluded that the maximum operating range for effective acoustic control (using this modem system) of a dynamic AUV with all integrated systems active is around 500 meters in the horizontal plane with ~85% reliability, assuming a very shallow water environment, arbitrary transducer altitude, and uncertain channel geometry.

F. VEHICLE BEHAVIOR CONTROL USING MODEM

Characterization of operating parameters such as time delays and effective ranges for the different combinations of encoding and modulation established the performance bounds and ensured that the limits on behavior control with the installed modem system were known with confidence. Using the code developed earlier for the *fm* and *Exec* processes, numerous open water tactical control tests were completed to demonstrate alteration of vehicle behaviors through acoustic transmission of new control modes and set points.

1. Discussion

Initial tactical control experiments, undertaken simultaneously with the short range transducer altitude characterization, had identical test conditions. The ARIES, with all integrated sensors active, was placed on a multi-leg mission using the vertices of a 30 meter square as waypoints at a horizontal distance of no more than 150 meters from the submerged topside transducer. The topside tranducer head depth was held constant at 1 meter while the ARIES transducer depth was varied in the water column through a total water depth of about 15 meters. Once proven, control commands were used throughout the remainder of characterization investigations.

Initial modem testing of the AUV with all sensors active was unsuccessful, in all studies, due to repeated mission aborts. As will be shown in the results later, it was determined (after a rigorous series of investigations) that in this instance, hardware, not software, was the source of the problem. Key design considerations in AUV development include minimizing weight and volume of the vehicle while maximizing the endurance times and numbers of sensors carried. Often times, these competing requirements lead to difficulties when integrating AUV sensor packages. Space constraints may lead to interference problems between sensors, either through operation in the same frequency spectrum or direct electromagnetic interference (EMI). Acoustic modem electronics were housed in the same forward compartment as the depth cell. Responses from the ARIES, even simple acknowledgements of acoustic message receipts from the topside modem, created enough of an electric field to cause erroneous “raw depth” readings on the depth cell. These erroneous values were the input to the “filtered

depth" solution and the rapid, erratic variations made the filter unstable. Unstable filter values triggered safety behaviors in the *Exec* process, aborting the mission. Therefore, to eliminate the problem and begin dynamic AUV experiments, commercial-off-the-shelf (COTS) shielding materials were procured and installed around the amplifier and power supply sections of the modem, pictured in Figure 19. Additionally, shielding was placed around all modem electrical wiring. Once EMI shielding was completed, all dynamic AUV testing resumed with no further depth cell failures.

2. Test Results

Data for the tactical control studies was collected for each mission by vehicle sensors and saved to a file. The data was processed in MatLab and figures were created for representative missions. First shown are figures of the vehicle response prior to modifying the modem with EMI protection to prevent depth cell failure. Later figures are examples of the tactical control tests completed once the EMI problem was resolved, including vehicle response to commands to change the control mode from depth control to altitude control and changes to the commanded depth or altitude operating value.

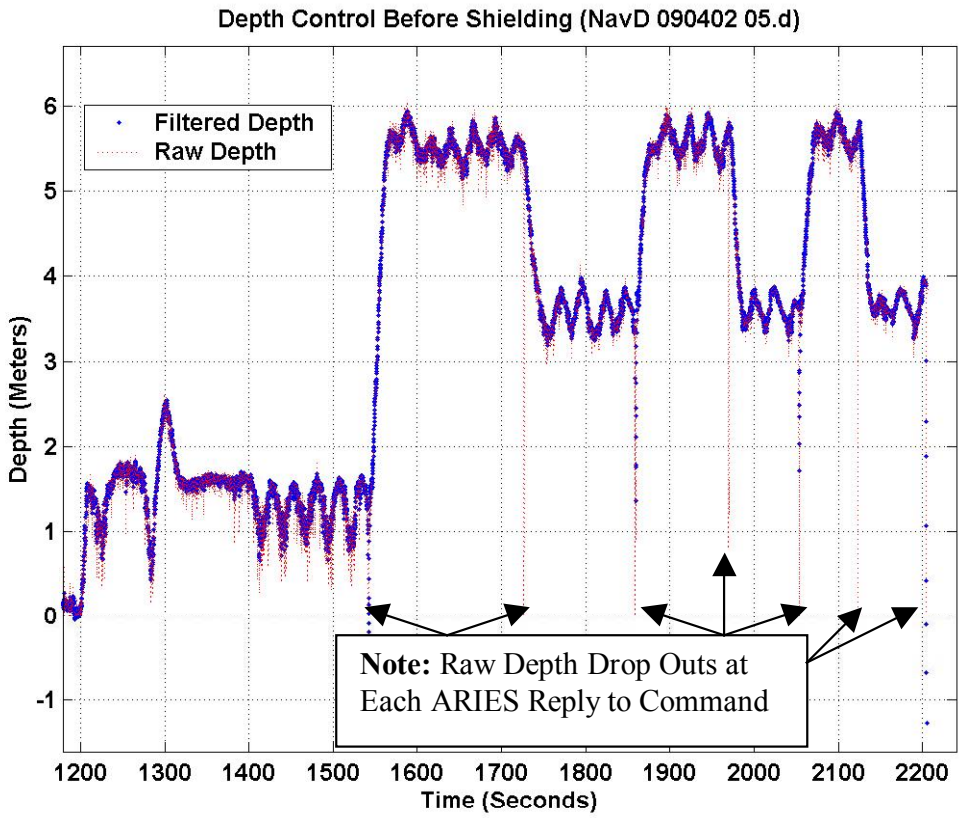


Figure 41. Depth Control, No EMI Shielding, Mission Abort

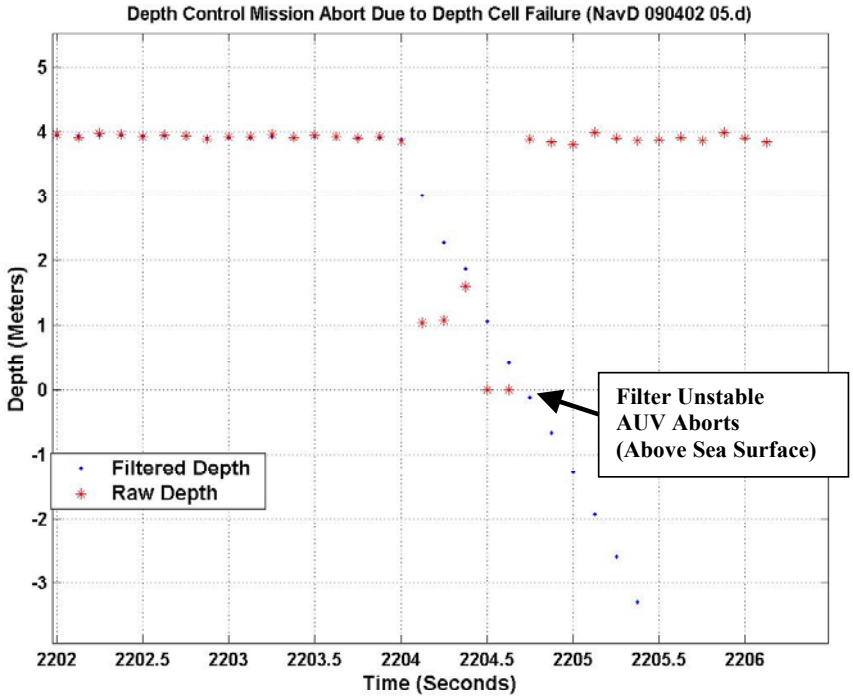


Figure 42. Depth Control, No EMI Shielding, Point of Mission Abort

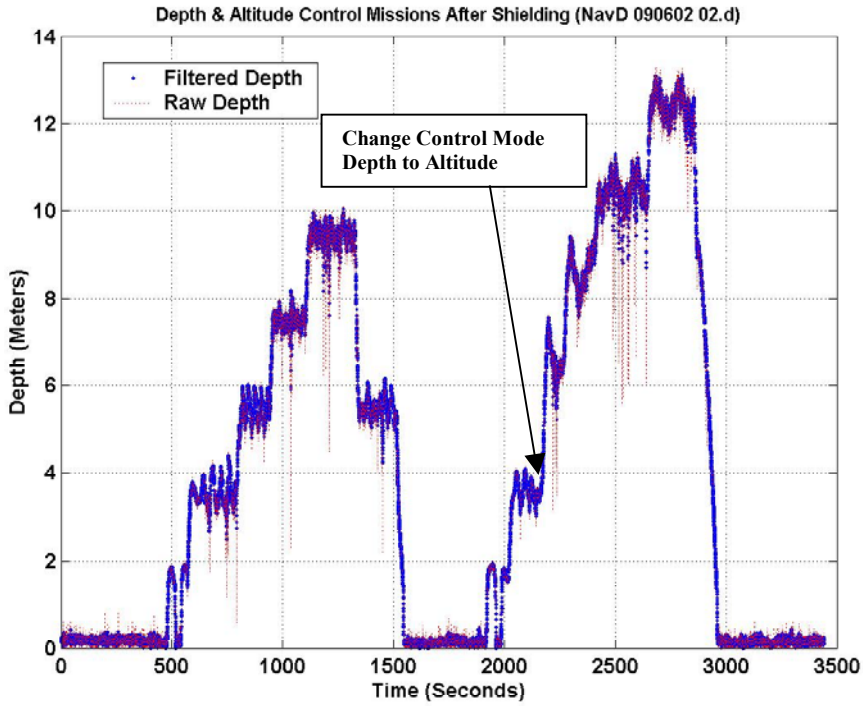


Figure 43. Depth & Altitude Control Mission, After Shielding

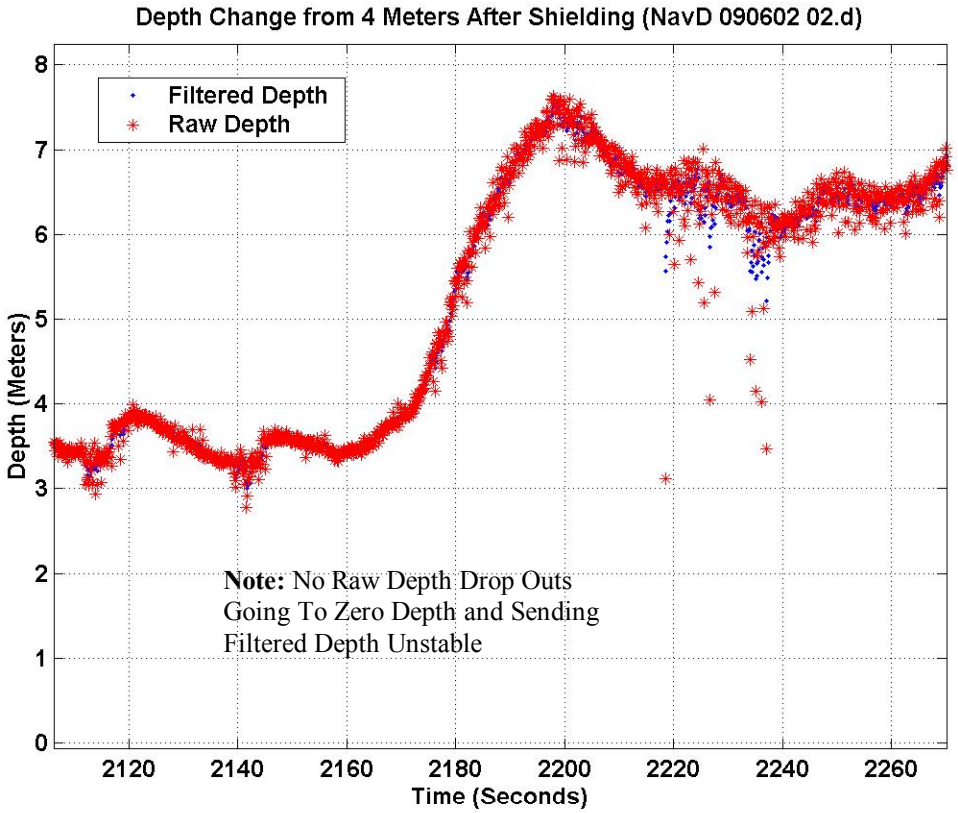


Figure 44. Zoom View of Depth Change After Shielding

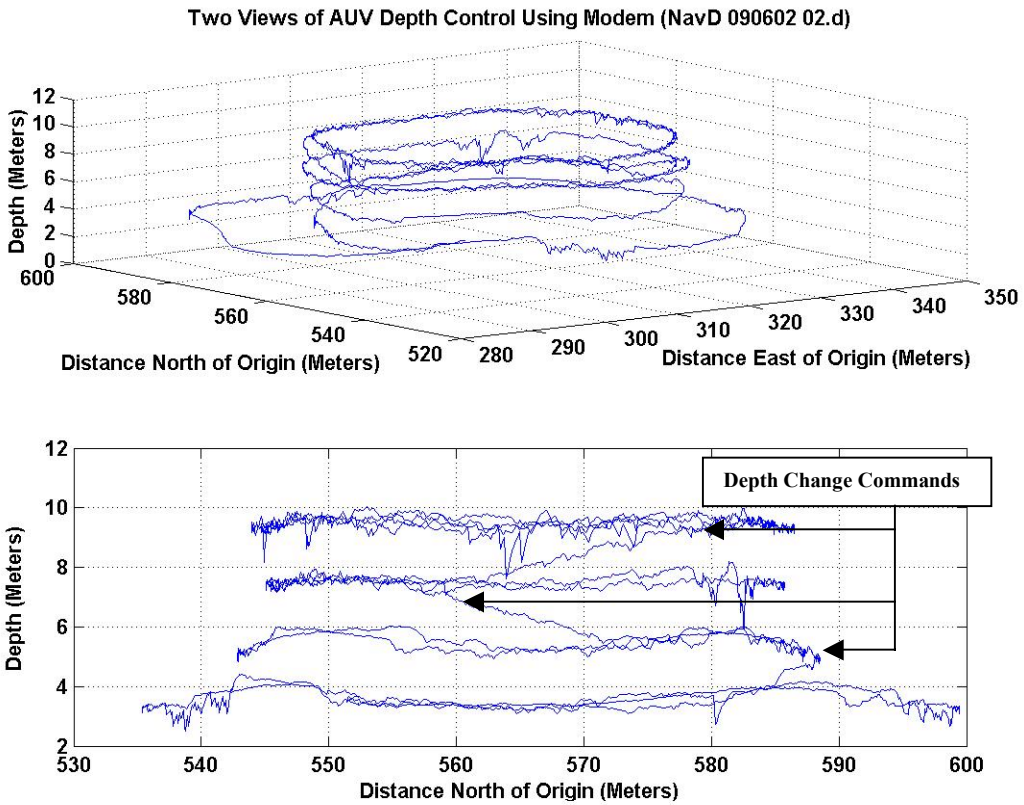


Figure 45. Representative Depth Control Run Using Acoustic Commands

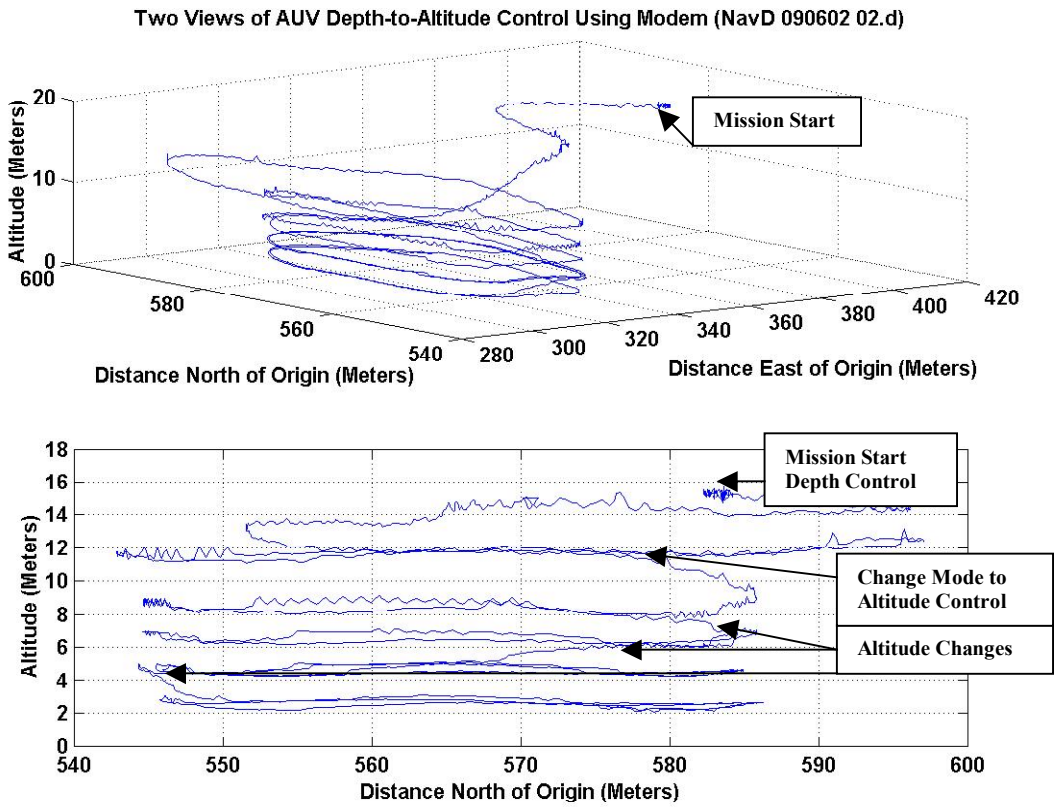


Figure 46. Representative Depth-to-Altitude Control Run Using Acoustic Commands

3. Conclusions

After isolating and rectifying hardware issues with EMI, the data in this series of tests demonstrated the ability to tactically control an AUV using acoustic communications. Control modes were reliably commanded to switch from depth control mode to altitude control mode or vice versa. The set point or operating point for the current control mode was also arbitrarily alterable at tactical controller discretion. Additionally, though not shown in the figures, the emergency abort command developed in the ARIES software was fully tested and performed flawlessly. As previously noted, software design prevented unintended consequences by allowing control modes and set points to be changed only if ARIES received the message completely correct and in the proper format; incorrectly received messages did not elicit action.

Some interesting observations can be made from the representative figures selected. Referring to Figures 41 and 42, it can be seen that missions aborted due to depth cell instability prior to solving the EMI problem whenever the ARIES modem transmitted even a short response such as an acknowledgement to the topside modem. Figure 41 was a mission where the vehicle started in the depth control mode at an operating depth of 1.5 meters. Control messages were sent to alternatively change depth set points from 6 to 4 meters. At each acknowledgement of command point, the raw depth (from the depth cell) is observed to be erroneous and tending toward a zero depth. The vehicle uses the raw depth signal as an input and filters it to determine the estimated vehicle state. If the estimated vehicle state passes zero, it indicates that the vehicle is above the surface of the water. Therefore, the mission will terminate. Note that at about 1865 seconds and 2060 seconds into the mission the filtered value began to approach zero but recovered. However, just after 2200 seconds, the depth filter became unstable and the mission aborted. Figure 42 is a magnified view at the point of mission failure.

Figures 43 and 44 show missions and a magnified view of depth cell response after EMI shielding was added to the hardware. Figure 43 shows two complete missions. The first, from about 500-1500 seconds, was a mission using only the depth control mode. The operating or set points were acoustically altered in 2 meter stages down to a depth of 10 meters. In addition to command acknowledgements, queries sent from the topside modem were replied to by ARIES. Although some erroneous raw depth values was still noted (primarily due to the long query reponses from the AUV), the improvement was readily apparent; no raw depth reading ever reached a zero depth value or drove the depth filter unstable. The second mission in Figure 43, from about 2000-3000 seconds, began in the depth control mode. After changing the depth set point to 4 meters, the vehicle control mode was acoustically changed to the altitude control mode. The remaining portion of the mission was completed by successively ordering the altitude set point to decrease by 2 meter increments down to an altitude of 2 meters above the bottom. As in the previous run, queries as well as commands were sent to the ARIES. Again, raw depth never reached zero or made the filter unstable. A magnified view of the raw depth values during a depth change from 4 meters is shown in Figure 44.

Figures 45 and 46 are graphic displays of mission performance is response to acoustic tactical control. Figure 45 was a mission using the depth control mode only and Figure 46 was a mission using both the depth and altitude control modes.

G. ACOUSTIC CONTROL CONCLUSIONS

All four modem configurations evaluated (modulation modes 4 and 3, each with block (BCH) and concatenated (BCH and convolutional) coding schemes) successfully demonstrated the ability to tactically control the AUV, through both one way and two way acoustic communication, at rates from 55 bits/second up to 220 bits/second. AUV computer software was developed to successfully alter missions and monitor vehicle status. One-way control modes and operating set points were sent to, received by, and acted on by the ARIES. Two way queries sent by the topside tactical controller modem were responded to by the AUV.

Summarizing the characterization and control studies in this entire section, tactical acoustic control in the shallow water environment with the FAU modem system installed in the ARIES was demonstrated but is limited in its utility to operational forces. Since transducer orientation and channel geometry are mission-dependent, it must be conservatively concluded that the maximum operating range for effective acoustic control of a dynamic AUV with all integrated systems active (using this modem system) is around 500 meters in the horizontal plane with ~85% reliability, at bit rates up to a maximum of 220 bits/second. When communicating at the maximum bit rate (mode3 modulation with block (BCH) coding), a transmission time of about 3 seconds plus the channel delay due to sound velocity will be required to complete each one-way transmission.

It is felt that the primary limitation to increasing range of communication lies in the loss of signal strength coming from multiple bottom and surface reflections and is especially apparent as sea state increases. For water depths around 15 meters, in the open ocean Bay conditions, the range represents about 25 water depths. Whether or not that would also follow in deeper channels is not known since complete acoustic modeling of VSW channels is not available.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ACOUSTIC DATA TRANSFER

A. INTRODUCTION/BACKGROUND

Global connectivity of air, surface, and subsurface sensor platforms (hence information superiority) dramatically increases the Warfare Commander's combat effectiveness. The Navy has established a program called ForceNet to "implement the theory of network-centric warfare."[65] Implicit in ForceNet is the requirement to amass near real-time data from a variety of underwater platforms using a high quality, relatively high data rate communication link. As previously discussed, acoustic communication is the primary means of underwater communication and much research has been completed with modems transmitting collected data at rates up to 20 kilobits/second in stationary shallow water testing. However, the highest published data transfer rate from systems integrated into a dynamic vehicle in the shallow water environment is currently limited to about 1200 bits/second at ranges in the hundreds of meters [25]. The purpose of this series of investigations was to integrate a relatively high speed commercially available modem system into the ARIES vehicle and assess the maximum ranges and data rates feasible for information transfer in the shallow water environment.

The experimental work on acoustic data transfer was completed using the Benthos, Incorporated ATM 890 series modem system installed in the ARIES AUV. This section begins with a chapter describing modem operating parameters, hardware used, and software development. The next chapter examines the performance characteristics of the installed system and the results of acoustic range and data transfer rate experiments in four common shallow water geometries. The final chapter summarizes conclusions of the studies in this section.

B. INTEGRATION INTO THE ARIES AUV

Once investigation of AUV tactical control using an acoustic modem was complete and the characteristics of the previous commercial system were well established, it was determined that data transfer experimentation would require a more advanced modem system with greater than a 220 bit/second data rate and 500 meter range

limitation. Subseqently, the Florida Atlantic University modem system was removed and the Benthos, Inc model ATM 890 series modem was selected and installed in the ARIES.

1. Benthos Modem Operating Parameters

The Benthos modem system (originally a Datasonics product) was an outgrowth of a Navy and industry partnership to develop an acoustic telemetry and ranging (telesonar) modem. Initially designed for use in "SeaWeb," an underwater network intended to link deployable autonomous distributed systems (DADS), the modem system has evolved and improved through several generations to its current versions, the ATM-880 and ATM-890 series. This chapter briefly examines the operational parameters and the encoding and modulation schemes specific to the system installed in the ARIES. Detailed information on the historical development and operating parameters of the Benthos 8XX series Acoustic Telemetry Modems can be found in references [52], [66], and [67].

a. Basic Specifications

Two types of Benthos modems are currently in use and each is asymmetric in nature (i.e. allows unequal acoustic transmit and receive baud rates). The ATM-880 series modem, without an additional coprocessing board, can transmit at rates up to 15,360 bits/second but receive only as high as 2400 bits/second while the ATM-890 series can theoretically transmit and receive up to the highest rate of 15,360 bps. Variations of noncoherent multi-frequency shift keying (MFSK) modulation are used to transmit encoded data at nominal coded rates from 150-2400 bits per second and coherent phase shift keying is used for higher speed communication from 2560-15,360 bps. Operating in the nominal low frequency range from 9-14 KHz, the system uses the AT-408 omnidirectional transducer and a 14-28 volt supply voltage. Signals are sent out at a maximum source level of 178 dB with average power of 21 watts and a peak power consumption up to 63 watts.

Each modem can use three distinct modes of operation when interpreting data and commands: either the "Online," "Command," or "Datalogger" mode. Typically, the host computer must use some type of terminal program such as ProComm or Hyperterminal to communicate with its local modem, or alternately, can use the manufacturer supplied graphic user interface (GUI) called Telesonar PC.

The Online mode permits free exchange of all characters between host computer/modem pairs; whatever characters are sent from the local host computer will be transmitted exactly as written to the remote host computer via the acoustic link. The Command mode is used to issue commands from the local host computer to the attached local modem or from the host computer to the remote host computer via acoustic link for subsequent execution by the remote modem. Either the Command mode or the GUI must be used to change operating parameters of the local modem. Changing the majority of the operating parameters of the remote modem, once deployed, may be done acoustically primarily with the GUI: the Command mode may query all operational settings but only alter certain settings such as the remote power and baud rates. In the Datalogger mode, all input from the local host computer is sent into flash memory of the local modem for storage until either read or cleared from memory by local or remote modem command.

Data input into the local modem from the host serial port is organized into packets consisting of up to four 4 kbyte transmit buffers for further transmission. Once a packet is filled, or a user-set forwarding delay time is reached, the data is transmitted to the remote modem over the acoustic channel. Packets include encoded data pulses plus header information to synchronize the transmission and specify the modulation method. Additionally, a cyclic redundancy check (CRC) checksum is calculated and transmitted with each packet to ensure the integrity of the entire message. The modem can be configured to select any of twelve acoustic baud rates.

b. Encoding Schemes

Recall that the fundamental purpose of any error correcting encoding method is to prepare the data for reliable transmission over a noisy channel. The increase in reliability and enhanced probablility of successful decoding is achieved at the cost of reduced channel throughput. At the lower baud rates where encoding is used, the data are primarily encoded with convolutional encoding schemes, with the exception that a linear block Hadamard code is used with the lowest baud rate (150 bps). One half rate convolutional coding is used for the five baud rates from 300-1200 bits/sec and the three baud rates from 2560-7680 bits/sec. No encoding scheme is used for the highest noncoherent baud rate (2400 bps) or the two highest coherent baud rates (10,240 and 15,360 bps).

Additionally, to improve reliability performance, data redundancy and multipath guard periods are featured in many of the lower baud rates. Recall that in general, data redundancy is assumed when any form of coding is used. However, data redundancy in the current context is defined as transmission of the same data bits two or more times in the same frame. The two lowest baud rates, 150 and 300 bits/sec, use data redundancy. Multipath guard periods are used only for the noncoherently modulated baud rates from 300-1066 bits/sec. Multipath guard periods are simply an increase in the duration between data frames or baud periods in an effort to avoid signal overlap in channels prone to high multipath.

c. Modulation Methods

The modem can be configured for two types of modulation methods for transmitting the data: either some form of spread spectrum frequency hopping MFSK for baud rates up to 2400 bits/sec or phase shift keying (PSK) for baud rates from 2560-15,360 bits/sec. The total available bandwidth of 5120 Hz from a nominal low frequency operating range of 9-14 KHz is divided into 128 bins, each slot about 40 Hz wide. One hundred twenty slots are used for data transmission with the remaining eight used for control purposes such as signal tracking.

The MFSK modulation method divides the 120 available frequencies bins or tones into thirty groups of four frequencies. The symbol size (q) for MFSK modulation is 2 bits, which comes from the fact that there are four possible frequencies (M) for each chip or symbol and $M = 2^q$. The maximum number of bits per pulse (also referred to as bits per signal event or bits per baud time) is 60: (2 bits/pulse/group of four frequencies) X (30 groups of four frequencies). Since the pulse width is 25 milliseconds, the maximum baud rate using MFSK modulation is 2400 bits/sec and the number of pulses per frame varies based on the encoding and modulation combination selected. The lower baud rates are generally more robust but subsequently require longer transmission times for a given amount of data bits. The figure below shows MFSK modulation at the maximum baud or bit packing rate of 2400 bits/sec. All higher baud rates require phase shift keying (PSK) modulation which increases the amount of data sent in a time period but makes the signals more difficult to detect and process, resulting in less reliable reception.

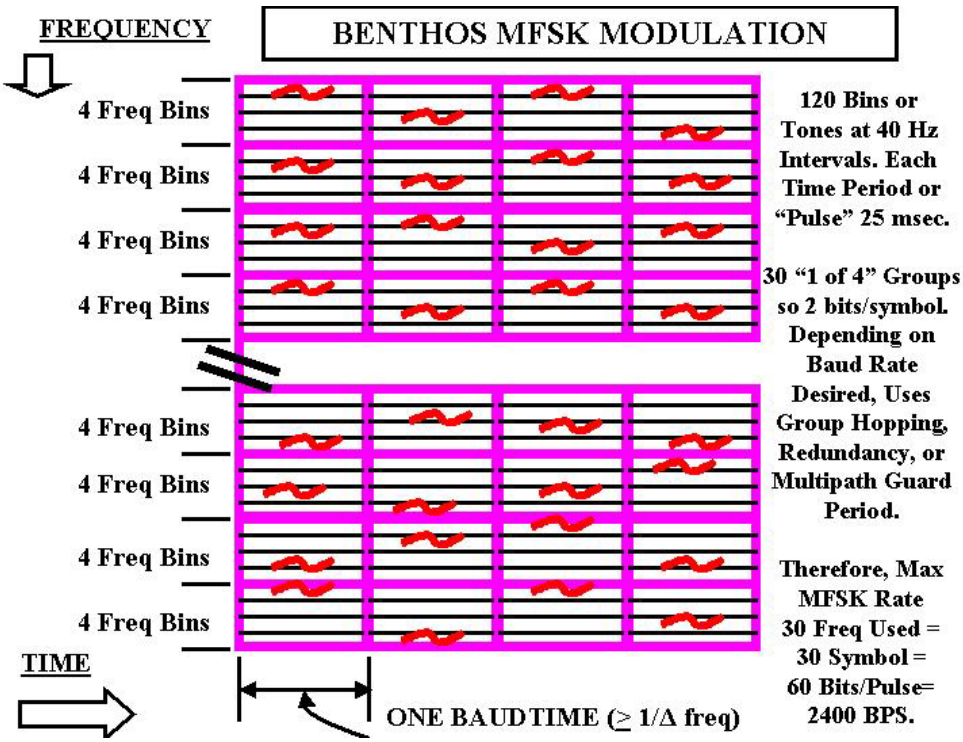


Figure 47. MFSK Modulation, Benthos Modem

2. Hardware Installation

Modem installation in the ARIES primarily involved rearrangement of components in the forward water-tight compartment near the free-flooded nose section. Based on previous experience, electromagnetic interference issues were eliminated by the construction of a shielded container for the modem electronics boards. The power input line, computer serial cable, and transducer connection line were attached, with mating connectors soldered where required. The modem board set shown below, measured in inches, was 8.75 x 2.81 x 2.60. The modem transducer head is also shown below. It was mounted in the free-flooded nose compartment and had a 3.90 inch diameter and 2.25 inch height.

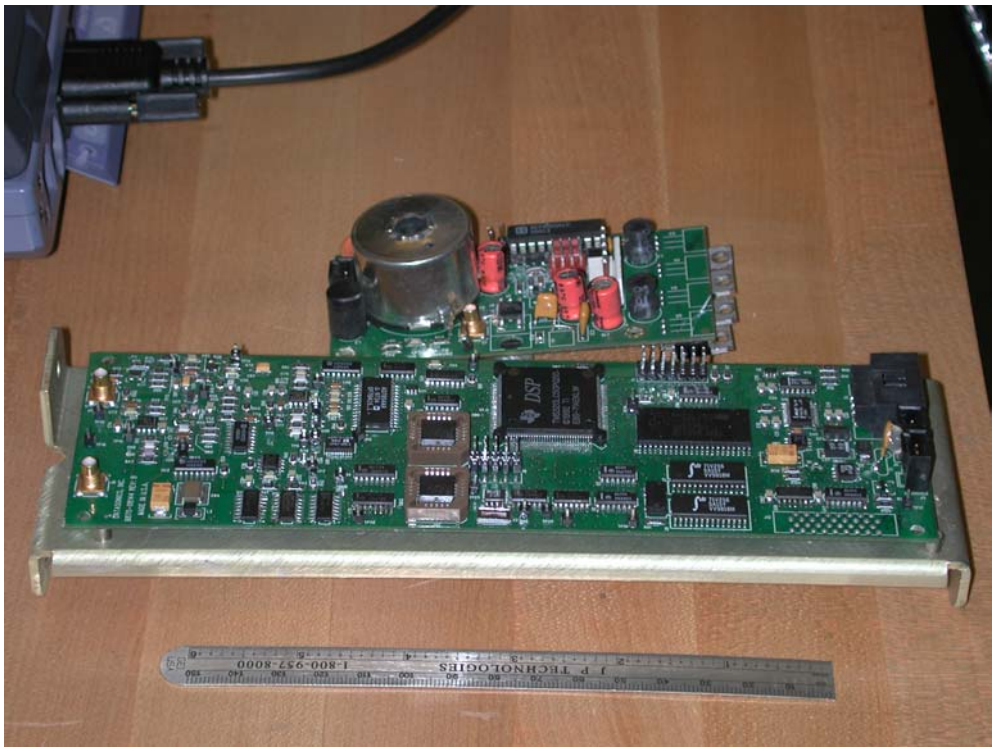


Figure 48. Benthos 89X Modem Board Set

Figure 49. Benthos Modem Transducer Mounted on ARIES

The “deck box” or local modem shown below contained a modem board set identical to that installed in ARIES. The deck box modem was transported throughout the operations area onboard a 22 foot Boston Whaler support boat. As with the FAU system, local modem communication and control was carried out via an RF link from the host laptop computer physically located at the pierside Command Post. The deck box was attached to its transducer by a 25 meter cable.

Figure 50. Benthos Deck Box Modem

Figure 51. Benthos LF Transducer and 25 Meter Cable

3. Software Development

The modem *fm* process in ARIES and its associated C language source code files (fm.c and fmf.c) required only relatively minor alteration from that fully described in a previous chapter. The key change was removal of the requirement for ARIES to send all configuration commands to its attached modem. The new modem system had the extremely useful capability that allowed the operator of the topside modem to remotely change any other modem's configuration parameters via acoustic link, thereby eliminating the need for the host computer to recompile new source code to set up the operating configuration prior to each mission. All other software functionality and use of command types, including the ability to query vehicle status and change mission parameters, remained the same.

C. CHARACTERIZATION OF LINK RANGE AND TRANSFER RATES

A comprehensive series of experimental studies was conducted to examine different aspects of modem system response and achievable data rates and ranges in various channel geometries. Although each of the four studies had a unique purpose and

test parameters, the unifying theme was to explore the bounds of data transfer in shallow water using an operational AUV. Quantification of the limits determined from the studies can then be extrapolated to a multivehicle scenario where one AUV is used as a "searcher" or "worker" vehicle to collect information in the VSW and shallow water environment while another is used as a "data server" vehicle that would accumulate data from several vehicles then surface to provide real-time information to warfare commanders.

1. Parallel Geometry (Constant Depth Channel)

Similarly to the case of acoustic mission control, successful acoustic data transfer is very dependent on the limits or bounds of the horizontal communication range. The first geometry investigated was the constant depth channel aligned parallel to the shoreline. The impetus for this study was to consider the operational scenario where a "searcher" vehicle is collecting data and a "server" AUV is required to come inshore to a similar depth to download the information for further transfer.

The ARIES, with the acoustic modem and all integrated sensors active, was placed on a multi-leg mission using the vertices of a 30 meter square as waypoints. The topside transducer and deck box, operating from a support boat, varied the horizontal distance from the ARIES operating area. Acoustic ranges were taken and query messages were sent at a baud rate of 150 bits/second at roughly 100 meter increments out to the maximum range that communications could be received and acknowledged. Transmit baud rates for the ARIES were remotely reconfigured using the deck box and varied from 150-5120 bits/second at each range. The parallel channel water depth was held constant at approximately 15 meters. Both tranducer heads operated in the mid-water column at a depth of approximately 8 meters. Using the maximum tranducer power setting (185 dB), a minimum of 10 communications attempts were made for each of the transmit baud rates at each horizontal distance to assess the failure rates and maximum effective range possible. Daily assessments of the background noise in the signal frequency spectrum (9-14 KHz) were recorded, with values ranging from 90-109 dB. Wave heights over the course of testing varied from 1-2 feet. A diagram of the channel geometry and baud rates tested is shown below.

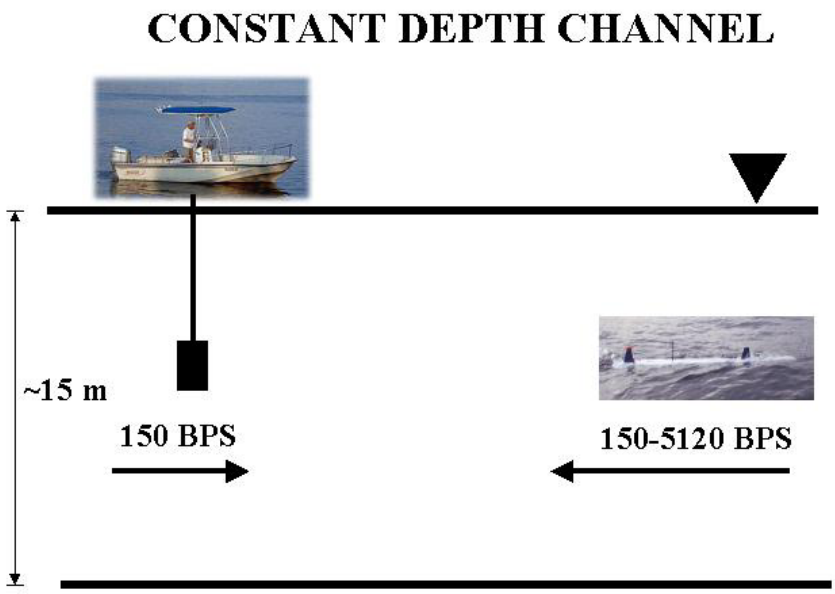


Figure 52. Parallel (Constant Depth) Channel Geometry

a. Results

The figure below summarizes the communications success rate of query attempts during seawater testing on the six achievable baud rates where the baud rate defined the combination of coding, guard period, redundancy and modulation method used. Each configuration was tested out to the maximum horizontal range in a constant depth (parallel) channel. Message attempts and receipts were logged, tabulated, and processed to produce a bar chart of communications success as a function of range and baud rate.

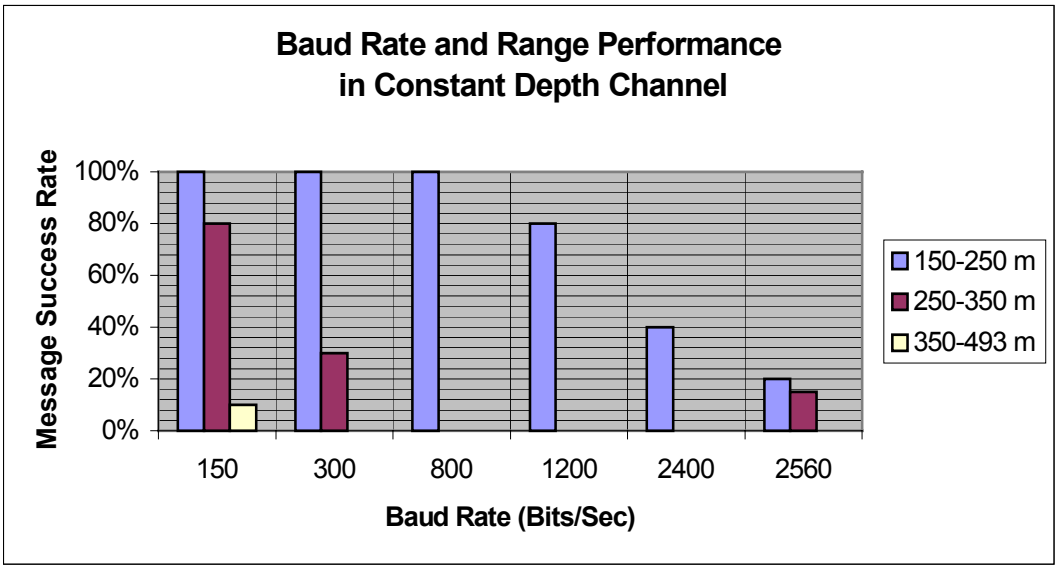


Figure 53. Message Success Rate (Constant Depth Channel)

b. Conclusions

The data in this series of tests quantified the maximum range or bounds of relatively high speed data transfer performance in a constant depth, shallow water channel for a dynamic AUV with all processes and sensors active. As in all characterization studies, the ARIES mission plan was selected to evaluate the worst case condition by ensuring there was a broad range of aspect ratios between the AUV and the deck box modem transducers.

The most important conclusion that may be drawn from the constant depth tests of the various modem configurations is that the maximum range for effective data transfer between server and searcher AUVs is approximately 300 meters with a data rate of 800 baud in this particular shallow water geometry when realistically high environmental noise levels are present. Referring to Figure 53, it can be seen that message success rates at ranges under 250 meters for the higher baud rate of 1200 bits/second are possible with about 80% success but as the baud rate increases, the success rate rapidly decreases. Although not shown on the figure, the baud rate of 5120 bits/second (using phase shift keying modulation) was attempted but was absolutely unsuccessful, even at the shortest range. When exceeding 250 meters in range, even the lowest baud rate possible, 150 bits/second, demonstrated only about 80% reliability.

Also noted is that the maximum range possible, even at the lowest data transfer rate, is less than 500 meters. The table below summarizes the data from the parallel (constant depth) channel data transfer characterization tests.

Parallel (Constant Depth) Channel	150 BPS, MFSK	300 BPS, MFSK	800 BPS, MFSK	1200 BPS, MFSK	2400 BPS, MFSK	2560 BPS, PSK
Range = 150-250 Meters (Received/Sent)	Assume 10/10	Assume 10/10	10/10	8/10	4/10	2/10
Range = 250-350 Meters (Received/Sent)	8/10	3/10	-	-	-	3/20
Range >350 Meters (493m Max) (Received/Sent)	1/10	-	-	-	-	-
Total Received at All Ranges	19/30	13/20				
Configuration Total % Success	63%	65%	100%	80%	40%	20%

Table 7. Parallel (Constant Depth) Channel Data Transfer Rate Summary

Much higher data transfer rates and longer transmission ranges had been anticipated. Using the same modem configurations in the same environment, a series of tests was conducted using two stationary transducer heads, eliminating AUV involvement. Similar range and data transfer success results were observed during static testing with only a marginal increase in maximum range obtained, reaching about 650 meters before all communication was lost.

2. Diverging (Increasing Depth) Channel Geometry

The second geometry studied to determine the bounds of relatively high speed data transfer rates and ranges was an acoustic channel perpendicular to the shore, also called a diverging or increasing depth channel, where the water column height was deeper or exceeded that in which the AUV is operating. As in the parallel (constant depth) channel study, the goal in this test series was to characterize the limiting range and the associated data transfer rates for the asymmetric modem system. However, the

operational scenario in this study had the "server AUV" (Boston Whaler and deck box) located off-shore of the in-shore "searcher vehicle" (ARIES).

Once again, the ARIES, with the acoustic modem and all integrated sensors active, was placed on a multi-leg mission using the vertices of a 30 meter square as waypoints. The topside transducer and deck box, operating from a support boat, varied the horizontal distance from the ARIES operating area. Acoustic ranges were taken and datalogger download commands were sent from the deck box at a constant baud rate of 150 bits/second at roughly 100 meter increments out to the maximum range that commands could be received and appropriate actions could be taken. Transmit baud rates for the ARIES were remotely reconfigured using the deck box and varied from 150-5120 bits/second at each range. The ARIES water depth was held constant at approximately 15 meters while the support craft water depths varied from about 15 meters at close range to nearly 30 meters at the maximum horizontal range. Both tranducer heads operated at a depth of approximately 8 meters. Using the maximum tranducer power setting (185 dB), communications attempts were made for each of the transmit baud rates at each horizontal distance to assess the failure rates and maximum effective range possible. A 38 byte message was stored in the ARIES modem datalogger memory. The command "AT$BT" was sent from the deck box to download the contents of the ARIES datalogger. Success of the download was determined by the number and quality of the data received from ARIES at each transmission baud rate. If 5 consecutive good replies were received at a particular baud rate and range, testing proceeded to the next baud and range combination. If replies were received in error, a minimum of 10 downloads were attempted. Daily assessments of the background noise in the signal frequency spectrum (9-14 KHz) were recorded, with values ranging from 92-107 dB. Wave heights over the course of testing varied from 1-3 feet. A diagram of the channel geometry and baud rates tested is shown below.

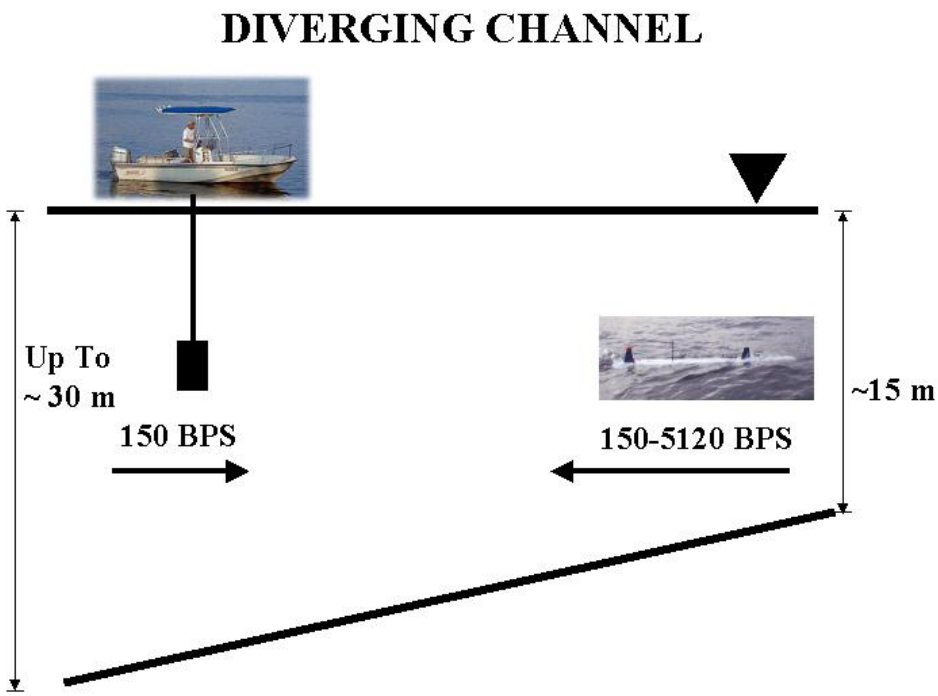


Figure 54. Diverging (Increasing Depth) Channel Geometry

a. Results

Summarized in the figure below is the communications success rate of datalogger download attempts during seawater testing on eight of the baud rates where the baud rate defined the combination of coding, guard period, redundancy and modulation method used. Each configuration was tested out to the maximum horizontal range in the diverging (increasing depth) channel. File download attempts and receipts were logged, tabulated, and processed to produce a bar chart of communications success as a function of range and baud rate.

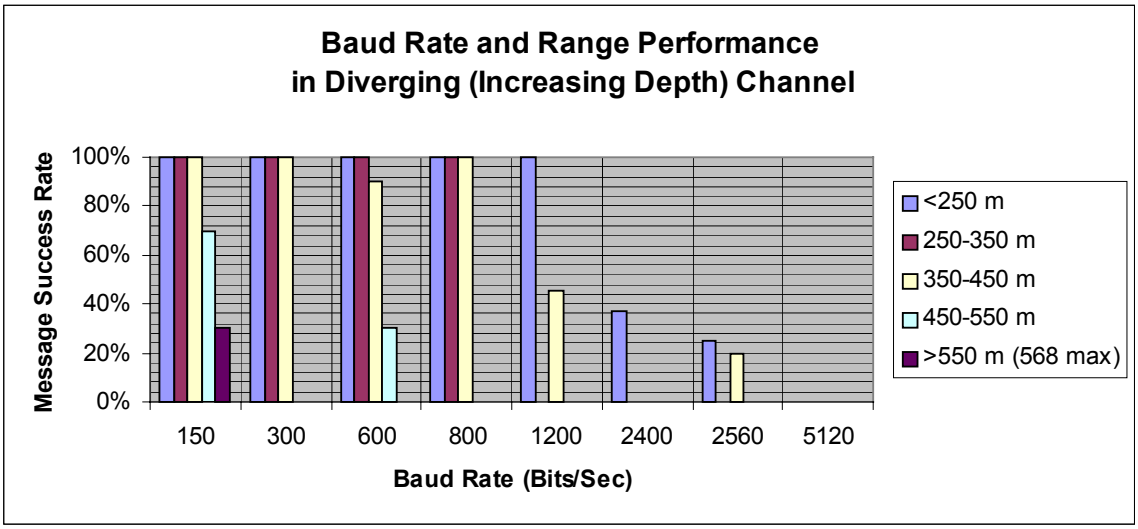


Figure 55. Message Success Rate (Increasing Depth Channel)

b. Conclusions

This series of tests bounded the maximum ranges and baud rates of relatively high speed data transfer performance in a diverging (increasing depth), shallow water channel. All processes and sensors were active and the ARIES mission plan was again selected to evaluate the worst case condition by ensuring there was a broad range of aspect ratios between the AUV and the deck box modem transducers.

Several important conclusions may be drawn from the diverging channel tests of the various modem configurations. Performance in this geometry exceeded that in the constant depth study. Most significant is that the maximum range for effective data transfer between server and searcher AUVs increased to about 450 meters with a data rate of 800 baud despite the relatively high environmental noise levels. Additionally, from Figure 55, it can be seen that message success rates at ranges under 250 meters were successful for the higher baud rate of 1200 bits/second. Similar to the constant depth channel, the success rate rapidly decreases as the baud rate increases. The baud rate of 5120 bits/second (using phase shift keying modulation) was completely unsuccessful at even the shortest ranges. When exceeding 550 meters in range, only the lowest baud rate (150 bits/second) was possible and demonstrated only about 30% success, achieving a

maximum range of 568 meters before communication was lost. The table below summarizes the data from the diverging (increasing depth) channel data transfer characterization tests.

Diverging (Increasing Depth) Channel	150 BPS, MFSK	300 BPS, MFSK	600 BPS, MFSK	800 BPS, MFSK	1200 BPS, MFSK	2400 BPS, MFSK	2560 BPS, PSK	5120 BPS, PSK
Range < 250 Meters (Received/Sent)	Assume 100%	Assume 100%	5/5	5/5	5/5	3/8	2/8	0/6
Range = 250-350 Meters (Received/Sent)	Assume 100%	Assume 100%	5/5	5/5	-	-	0/5	-
Range = 350-450 Meters (Received/Sent)	Assume 100%	10/10	9/10	5/5	5/11	0/5	2/10	0/7
Range = 450-550 Meters (Received/Sent)	14/20	-	3/10	-	-	-	-	-
Range >550 Meters (568m Max) (Received/Sent)	3/10	-	-	-	-	-	-	-

Table 8. Diverging (Increasing Depth) Channel Data Transfer Rate Summary

Similarly to the constant depth channel study, much higher data transfer rates and longer transmission ranges had been anticipated. Static tests were completed without ARIES participation using two stationary transducer heads and identical experimental parameters to confirm the baud rate and range performance observed during dynamic testing. Again, range and data transfer success results observed during static testing correlated well with the dynamic experiments. Range and baud rate performance was similar in all cases with the exception of the lowest baud rates, where an increase in maximum range was obtained, reaching about 791 meters with a 150 bits/second rate before all communication ceased.

3. Converging (Decreasing Depth) Channel Geometry

The third channel geometry studied, also an acoustic channel perpendicular to the shore, was the converging or decreasing depth channel. Unlike the previous studies, the goal in this test series was to characterize the limiting range and highest baud rate that an in-shore "searcher vehicle" could receive and correctly interpret commands from the off-shore "server AUV."

Using the standard multi-leg mission around a 30 meter square, the topside transducer and deck box horizontal distance from the ARIES operating area was systematically varied. Acoustic ranges were taken and query messages were sent from the deck box at baud rates of 150-5120 bits/second at roughly 100 meter increments out to the maximum range that commands could be received, interpreted and appropriately answered. The transmit baud rate for the ARIES was held constant at 150 bits/second at each range to ensure accurate replies if the query was received. The ARIES water depth was held constant at approximately 15 meters while the support craft water depths varied from about 15 meters at close range to nearly 30 meters at the maximum horizontal range. Both tranducer heads operated at nominal 8 meter depths. Using the maximum tranducer power setting (185 dB), query attempts from the deck box were made for each of the transmit baud rates at each horizontal distance to assess the failure rates and maximum effective range possible. Success rates were determined by the number and quality of the replies received from ARIES at each transmission baud rate. If 5 consecutive good replies were received at a particular baud rate and range, testing proceeded to the next baud and range combination. If replies were received in error, a minimum of 10 queries were attempted. Background noise in the 9-14 KHz spectrum were recorded with values ranging from 90-107 dB and wave heights varied from 1-3 feet. A diagram of the channel geometry and baud rates tested is shown below.

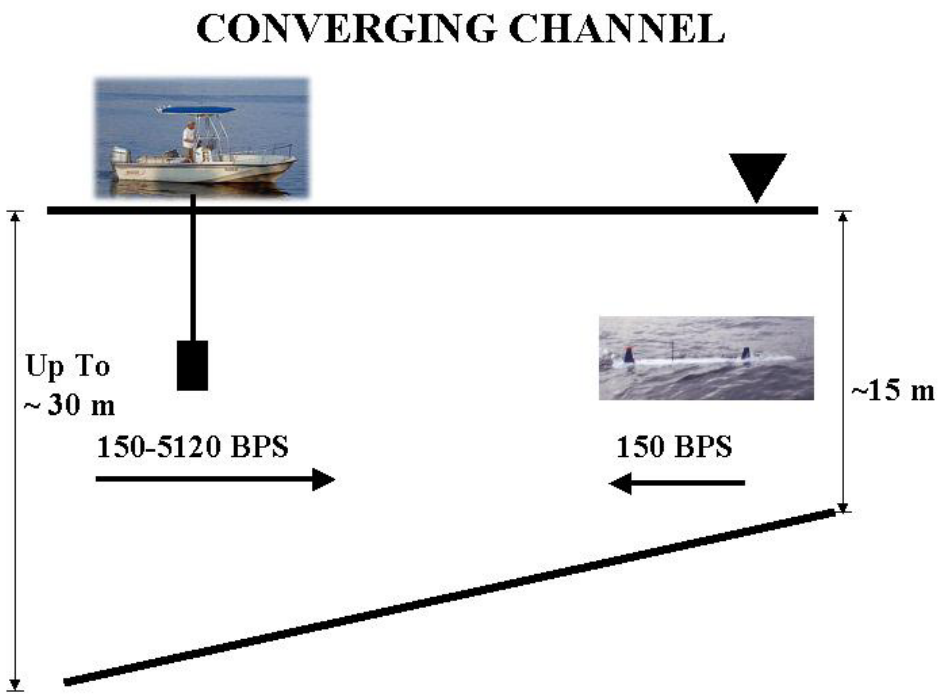


Figure 56. Converging (Decreasing Depth) Channel Geometry

a. Results

The figure below summarizes the maximum ranges and baud rates that the "server vehicle" could successfully achieve when sending commands and queries in-shore to the "searcher vehicle" in shallower water. The communications success rate of query message attempts was evaluated for all baud ranges achievable up to a maximum rate of 5120 bits/second. Each configuration was tested out to the maximum horizontal range in the converging (decreasing depth) channel. Query responses were logged, tabulated, and processed to produce a bar chart of communications success as a function of range and baud rate.

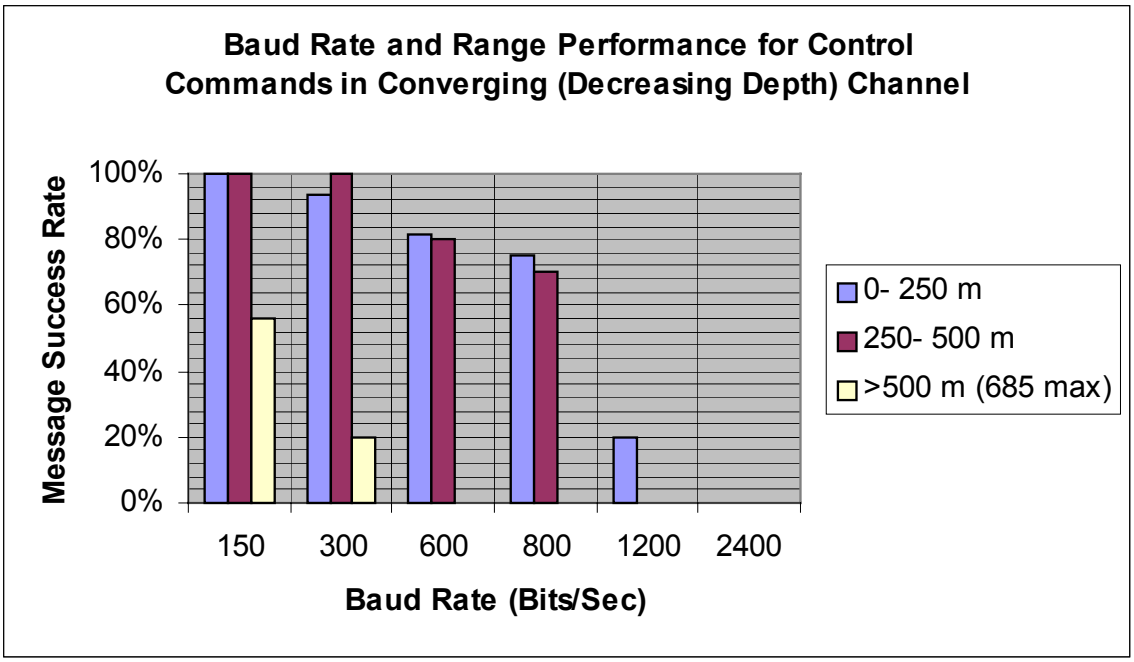


Figure 57. Message Success Rate (Decreasing Depth Channel)

b. Conclusions

The key conclusion from the converging channel test series was that a server vehicle could reliably send commands and queries successfully from about 500 meters offshore assuming the lowest, most robust baud rates were used. The maximum range and rate achieved, 685 meters at 150 bits/second, demonstrated greater than a 50% reliability. Another observation is that communication was possible only with the five lowest of the eight baud rates attempted. The highest frequency shift key modulation bit rate (2400 bits/second) and the two phase shift keying modulation rates (2560 and 5120 bits/second) were completely unsuccessful; the searcher AUV was simply unable to correctly receive and interpret commands at the highest baud rates. Referring to Figure 57, it can be seen that at long ranges, the server AUV must transmit at maximum baud rate of 150 bits/second to ensure command and control messages will be properly received and acted upon by the searcher vehicle. However, low baud rates for command and control messages are a rational and reasonable assumption since message lengths are generally very short and the speed of communication is secondary to ensuring the correct reception by the searcher vehicle. The table below summarizes the data from the converging (decreasing depth) channel characterization tests.

Converging (Decreasing Depth) Channel	150 BPS, MFSK	300 BPS, MFSK	600 BPS, MFSK	800 BPS, MFSK	1200 BPS, MFSK	2400 BPS, MFSK
Range = 0-250 Meters (Received/Sent)	10/10	14/15	18/22	15/20	2/10	-
Range = 250-500 Meters (Received/Sent)	Assume 100%	5/5	8/10	7/10	-	-
Range > 500 Meters (685 m Max) (Received/Sent)	9/16	2/10	-	-	-	-

Table 9. Converging (Decreasing Depth) Channel Data Transfer Rate Summary

4. Transducer Altitude Effects at Short Range

The final channel geometry was studied to investigate the effects, if any, on data transfer rates as the vehicle position in the vertical plane varied. Unlike the other studies of performance in the horizontal plane where maximum range characterization was desired, distance was eliminated as a variable of primary concern in these tests by limiting the range to about 150 meters or less.

The support boat and deck box modem were anchored in the immediate vicinity of the ARIES operating area while the vehicle completed a dynamic multi-leg mission around a submerged 30 meter square. Depending on the stage of the tide, total water depth of the operating area was approximately 15 meters. To negate the tidal variations during the study, altitude above bottom was used as the reference rather than depth below the surface. The deck box tranducer head depth was held constant at 8 meters while the ARIES altitude (and therefore transducer depth) was varied. Three baud rates at each of three different altitudes were selected for investigation. Using the maximum tranducer power setting (185 dB), query messages were sent from the deck box at a constant 150 bits/second baud rate to the submerged AUV. The transmit baud rate for the ARIES replies was varied from 800-2560 bits/second at 2, 8, and 12 meter altitudes. Periodically, acoustic ranges were taken to ensure distance between modems could continue to be neglected. Success rates were determined by the number and quality of the replies received from ARIES at each transmission baud rate and altitude. If 5 consecutive good replies were received at a particular baud rate and altitude, testing proceeded to the next baud rate at that altitude. If replies were received in error, a

minimum of 10 queries were attempted. Once all tests at a particular altitude were complete, ARIES was commanded to a new altitude set point and baud rate testing resumed. Background noise in the 9-14 KHz spectrum ranged from 90-105 dB. A diagram of the channel geometry and baud rates tested is shown below.

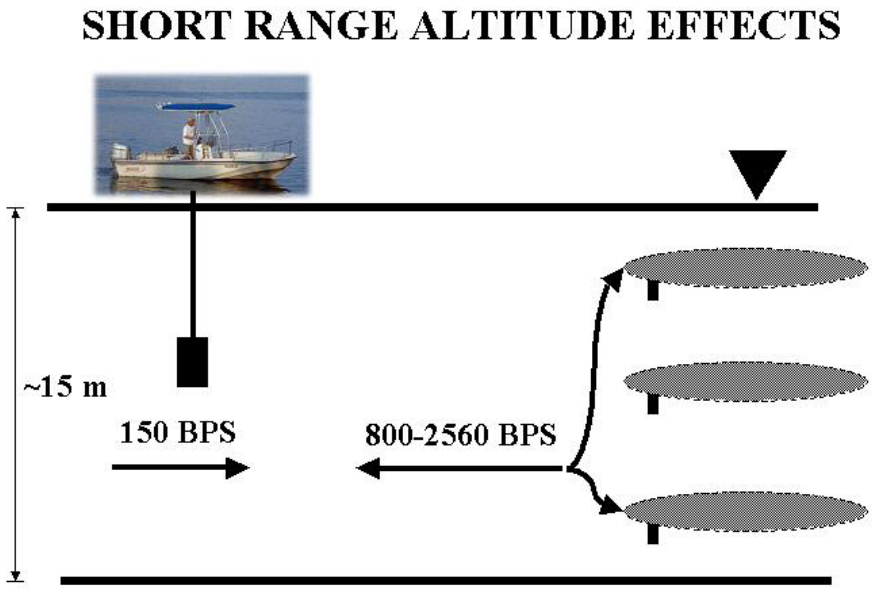


Figure 58. Short Range Altitude Effects Test Diagram

a. Results

The figures below summarize the more than 150 queries that were attempted during open water investigation of the three baud rates selected: 800, 1200, and 2560 bits/second. Each configuration was tested at three different altitudes: 2, 8, and 12 meters above bottom. Message attempts and receipts were logged, tabulated, and processed to produce bar charts of communications success as a function of altitude above bottom, baud rate, and a combination of the two.

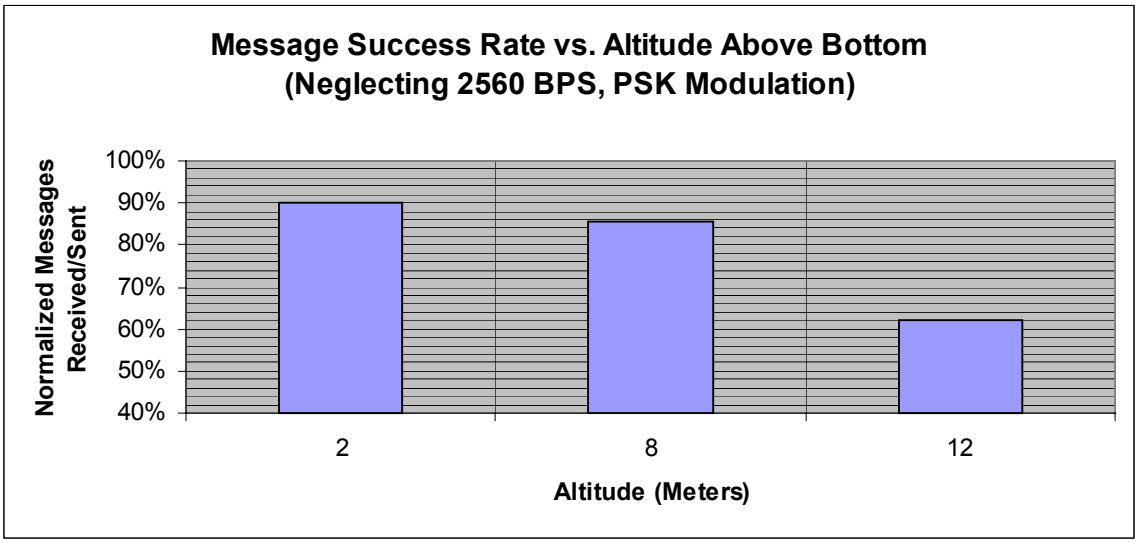


Figure 59. High Data Rate: Altitude Effects

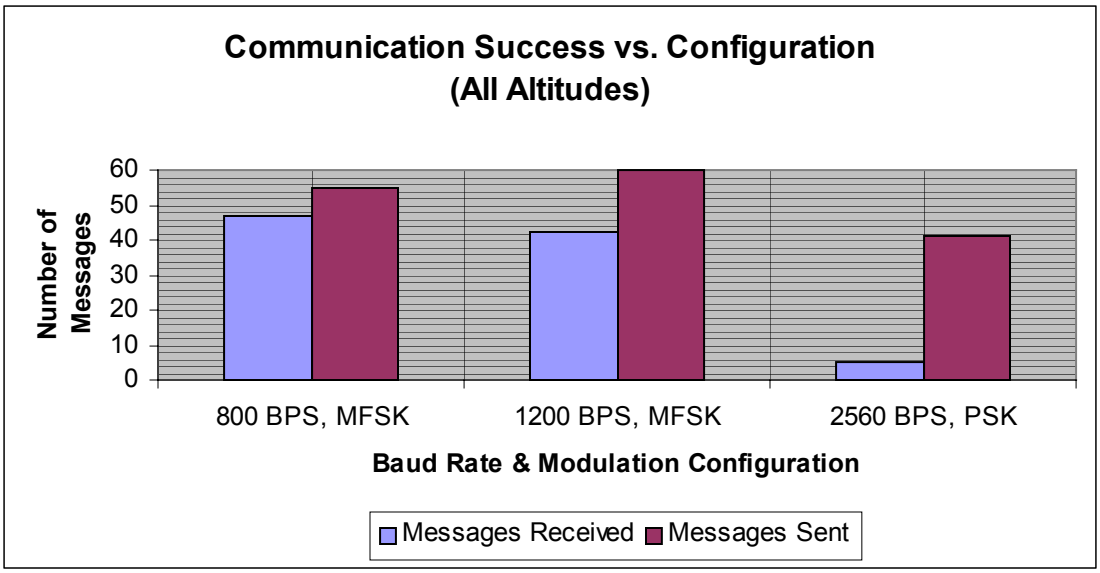


Figure 60. High Data Rate: Baud Rate Success

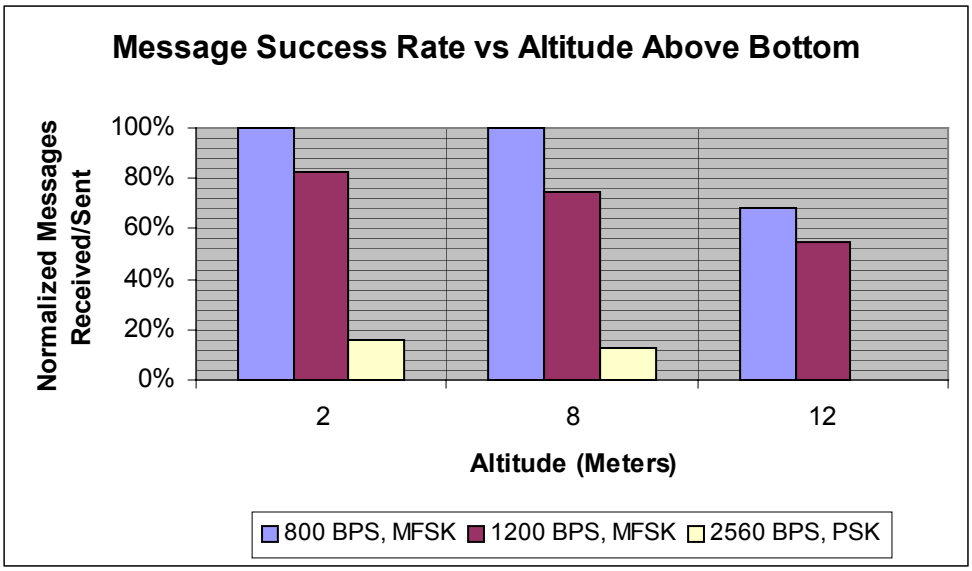


Figure 61. High Data Rate: Altitude Effects and Baud Rate Success

b. Conclusions

The data in this series of tests quantified the relatively high speed data transfer performance at different altitudes above bottom for various modem configurations. The most interesting observation that can be made from this study, evident in Figures 59 and 61, is that the messages were most successfully received when the vehicle was operating 2 meters above the bottom, a fact that correlates well with earlier studies done with the FAU modem system. In particular, Figure 61 clearly shows the decline in high speed communications performance as both the baud rate and altitude above bottom increase. While these results are not completely conclusive due to small variations in range during testing and the limited sampling size, they do indicate a strong trend toward improved communications when an AUV is operating near the ocean floor. Multipath reflections may explain this trend. Even though the horizontal range is relatively short, there are multiple acoustic signal reflections from both the surface and the bottom generating communication interference and increased signal power loss. Additionally, the number of surface reflections experienced by the vehicle transducer is possibly reduced by AUV body, which functions as a shield or baffle. The chart below summarizes the data from characterization tests on altitude effects and high speed data transfer.

Effects of Altitude on Communication Success at Short Range	800 BPS, MFSK	1200 BPS, MFSK	2560 BPS, PSK	Total	% Success Neglecting 2560 BPS, PSK	% Success Overall
Alt=2 Meters (Received/Sent)	15/15	16.5/20	2.5/16	34/51	90%	67%
Alt=8 Meters (Received/Sent)	15/15	15/20	2.5/20	32.5/55	86%	59%
Alt=12 Meters (Received/Sent)	17/25	11/20	0/5	28/50	62%	56%
Total	47/55	42.5/60	5/41			
% Success	85%	71%	12%			

Table 10. Altitude Effects and High Speed Data Transfer Rate Summary

D. CONCLUSIONS

The eight modem baud rates evaluated ranged from 150-5120 bits/second with various levels of data redundancy, multipath guard periods, and convolutional coding. The two highest baud rates used phase shift keying modulation while all other baud rates used multi-level frequency shift keying. While limited success was achieved with some of the higher baud rates, the maximum distance and rate observed for 100% reliable high speed data transfer in this environment was about 440 meters at 800 bits/second. At decreased ranges below 250 meters, the 1200 bits/second baud rate was about 90% reliable, dependent on channel geometry. The high baud rates using phase shift keying (2560 and 5120 bits/second) were unable to communicate effectively. Even at ranges of under 100 meters, communications success with the 2560 bits/second rate was less than 25% reliable and no coherent replies were received when using the 5120 bits/second baud rate. The maximum range where communication was possible, even using the most robust data transfer rate (150 bits/second), was 685 meters.

Summarizing the series of studies in this entire section, relatively high speed asymmetric data transfer in the shallow water environment with the Benthos modem system installed in the ARIES was demonstrated but was severely limited in performance in the adverse yet realistic shallow water operational environment. High ambient noise levels and the time and spatial variation of the acoustic channel degraded the performance that had been anticipated and was previously demonstrated in more benign

waters. In operational scenarios, transducer orientation and channel geometry are often arbitrary, unknown, and non-uniform. Therefore, it must be conservatively concluded that the maximum operating range for effective high speed data transfer in shallow water from a dynamic AUV with all integrated systems active (using this modem system) is nearly 300 meters in the horizontal plane with nearly 100% reliability, at bit rates up to a maximum of 800 bits/second. Although the actual distances achievable in these adverse, noisy (but realistic) waters was relatively small, in terms of non-dimensional distances as a function of water depth the modem performed quite well considering the multipath and signal reflections experienced. For example, the AUV was in a total water depth of about 15 meters but was able to effectively communicate out to 300 meters, a total of 20 water depths. Finally, there is a clear correlation between vehicle position in the vertical water column and its ability to effectively communicate. Message success tended to increase as the vehicle approached the sea floor.

It is conjectured that multiple surface and bottom bounces with very shallow water channels increase the signal power loss. This phenomenon is difficult to predict and is dependent on sea state. Increasing sea state is known to reduce communication range. Possibly further acoustic studies of very shallow water channels would elucidate this effect.

THIS PAGE INTENTIONALLY LEFT BLANK

V. COOPERATIVE AUV RENDEZVOUS AND TRACKING SIMULATION

A. DISCUSSION

Results of the in-depth experimental studies in the previous chapters clearly demonstrated that only relatively short distances (on the order of hundreds of meters) are reliably achieved by acoustic modems installed as a part of an active AUV system when operating in a realistically adverse VSW or shallow water environment, particularly as the baud rate increases. As previously stated, Warfare Commanders desire (and Net Centric warfare implies) the gathering and dissemination of near real-time information from sensor platforms. However, Warfare Commanders are generally physically separated from the AUV operating areas by relatively large distances (on the order of kilometers or tens of kilometers), far beyond the feasible range of acoustic communication. Unable to receive the acoustic information directly, an alternative method of data transfer must be considered. One possible solution would be to place stationary buoys in the vicinity of the operating areas to collect and transfer the information. However, use of this method would still suffer from the relatively short ranges of communication possible in adverse environments and could subsequently involve a rather large number of overt stationary nodes. Another, and possibly better, solution lies in the use of a mobile data collecting node or "server" vehicle similar to the ARIES. The "server" AUV would acoustically collect the data from *in situ* "searcher" vehicles in the minefield area and then surface to forward the assimilated data by RF or satellite links, allowing the "searcher" AUV to continue its mission.

The underlying goal of the server-searcher concept is to provide near real-time information to the Fleet. One crucial obstacle that must be overcome prior to concept implementation is the dual problem of rendezvous and cooperative multivehicle control during data transfer. Ensuring a relatively short range between AUVs increases the reliability of correct reception and allows for greater transmission baud rates, thereby reducing the total time required for data transfer. For example, using the results from previous chapters, a server vehicle at a 300 meter range from the searcher could reliably

receive data at a maximum rate of 800 bits/second (100 characters/second or bytes/second). Downloading a 25 kbit image (3125 bytes) would take about 32 seconds. However, if the range is reduced to under 150 meters, the maximum rate for completely reliable communication increases to 1200 bits/second (150 characters/second or bytes/second), which would reduce the download time to about 21 seconds. Continuing to further reduce the range between vehicles (to distances on the order of meters) would allow even higher reliable baud rates with the theoretical possibility of transmitting moving images and achieving baud rates up to 500 kbits/second using ultrasonic frequencies [68]. In general, higher operating frequencies and shorter transmission ranges equate to greater available bandwidth and higher data transfer rates. An additional advantage to high frequency transmission (especially in the ultrasonic range) is its clandestine nature. There is a low probability of detection of signals outside of a limited range due to the rapid attenuation of high frequency waveforms in seawater.

Initial work toward using a "server" AUV (referred to as a "data truck") to collect data has been conducted by a consortium of academic and research partners with efforts to date well-documented in [25] and [69]-[71]. The primary result of initial testing was validation of the concept since the "server" vehicle, completing mission plans of nominal circles at unspecified ranges, was able to establish and maintain communications with the stationary node at rates up to 600 bits/second. However, the thrust of previous research has been the retrieval of data from a fixed or stationary node, a capability the NPS Center for AUV Research intends to demonstrate during the August 2003 Autonomous Ocean Sampling Network (AOSN) exercise. The transfer of data between moving nodes is a more complex operation requiring the two nodes to fly in formation for at least a finite, short period of time. In order to examine the feasibility of this type operation, a computer simulation was made for a two-vehicle approach the results of which are described in this chapter.

B. SIMULATION CODE DEVELOPMENT

The simulation was created in MatLab and builds on the previous work in [72], where it was shown that it was possible to maneuver an AUV to a specific rendezvous point at a specified time. In essence, this work demonstrates the ARIES making

rendezvous with the Searcher AUV upon being acoustically summoned from its loiter area at a random time for download or transfer of data from the Searcher. Upon completion of the initial rendezvous, ARIES continues to parallel the Searcher's track for four more waypoints to allow sufficient time for the acoustic modems to pass all of the information. After data transfer is complete, ARIES breaks away from the Searcher's track and returns to its original loiter area to transmit the data collected to the Warfare Commander via RF or satellite link.

1. Mission Scenario

It is assumed that ARIES would initially traverse a 60 meter by 120 meter loiter area off-shore from the Searcher AUV at a constant speed. Meanwhile, the Searcher initiates its underwater search plan using a standard "mow-the-lawn" pattern, passing through mission waypoints every 50 meters while completing nominal 300 meter by 50 meter lanes. When the Searcher finds an object of interest, or approaches its maximum capacity for onboard data storage, it sends ARIES an acoustic "request to rendezvous". The ARIES, with *a priori* knowledge of the Searcher mission plan, computes the most suitable rendezvous waypoint using the "feasibility and waypoint selection" loop described in the next section, and proceeds to meet the Searcher. After meeting at the rendezvous site, ARIES uses position feedback data from the Searcher for speed control to minimize the distance in the horizontal plane between the two vehicles during the acoustic transmission of search data. After transferring data for four more waypoints (200 meters), ARIES breaks away and returns to its loiter box while the Searcher continues its original mission plan.

2. Highlighted Features of Code

There were many minor alterations to the previous work in [72] to more closely model realistic conditions including representation of the "request to rendezvous" by a random time generator, incorporation of an acoustic signal delay, computation of Searcher kinematics, and improvements in plotting routines, among others. However, two areas were crucial to successful simulation of the cooperative tracking and rendezvous concept: code to determine rendezvous location and feasibility and the development of a speed controller to maintain vehicle proximity after rendezvous.

Once the request to rendezvous was received, a section of ARIES MatLab code began an immediate computation in a "feasibility and waypoint selection" loop (described below) to determine the waypoint that was most suitable for initial rendezvous. Since the Searcher mission plan and search speed were known *a priori*, the position could be estimated and the time of arrival at the next waypoint could be projected. Using the projected time of Searcher arrival and ARIES known current position, feasibility of the using that waypoint as the point of initial rendezvous was determined. This was done by ensuring that the projected time was sufficient for ARIES to reach the same waypoint when operating within the limits of its speed and acceleration constraints. If simultaneous arrival of both vehicles was not possible, the next waypoint on the Searcher track was selected as a possible rendezvous point and the algorithm iterated. Once a waypoint meeting the minimum acceptable constraints of both AUVs was found, the next waypoint on the Searcher's track was selected as the rendezvous point.

After initial rendezvous, there was a need to maintain close proximity to allow high speed, reliable acoustic data transfer. Once again, the estimated position of the Searcher was known or could be assumed to be updated as part of the acoustic data transfer. Using ARIES' known position and Searcher's estimated position, an along-track position error was computed. This position error was used in a sliding mode controller to alter ARIES speed. Sliding mode control, described in great detail in [73], is essentially a robust method used here to minimize the distance error between the two vehicles. The cross-track error controller (CTE Control) used by ARIES to minimize the perpendicular distance between the vehicle and the desired track has previously been implemented and is described in depth in [36]. However, the along-track error controller is a recent development. Upon initial rendezvous, the position difference (delta_posit) between ARIES and the Searcher is computed. The along-track component of the positon error is calculated (posit_error). This result is used in a sliding mode controller to alter the speed of ARIES (new2_U) in a manner to eliminate any position error

throughout the remaining period of cooperative tracking. The speed command based on the position error is shown below and the remaining details of the development are in the MatLab code in Appendix H.

```
new2_U(iii+1) = U_Searcher(iii) +
new2_eta_posit(iii)*tanh(posit_error(iii)/new2_phi_posit(iii))

Where new2_eta_posit(iii) = 0.2
and new2_phi_posit(iii) = 2.0
```

The vehicles remained in close proximity for four waypoints (200 meters). This distance allows sufficient time for data transfer of a reasonably large file. For example, even at the relatively low baud rate of 1200 bits/second, vehicles operating at search speeds around 1.4 meters/second will travel together for about 143 seconds, enough time to transmit a 172 kbit file.

3. Simulation Output

Many mission parameters are immediately available on-screen for observation during each run of the multiple vehicle rendezvous and tracking simulation. When the request for rendezvous is sent by the Searcher vehicle, the time and ARIES position at that time are printed. Once the mission is deemed feasible, the user is informed of the time of and distance to the rendezvous point. After the initial rendezvous point is reached and its location given, exact position coordinates for ARIES are listed as each subsequent waypoint is completed. Finally, a series of plots are generated to examine simulation performance and the behavior of variables of interest such as vehicle speed, cross-track error, vehicle heading, cooperative tracking, and position error. A sample of the written simulation output for a run where Searcher requested ARIES to rendezvous after 80 seconds is in Appendix G. Some of the plots or figures associated with that run will be examined in the next chapter. The full version of the MatLab cooperative rendezvous and tracking simulation (final_data_xfer_11Dec_19Apr.m) is included in Appendix H.

C. SAMPLE RESULTS AND OBSERVATIONS

Although six figures are generated by each specific simulation run, three plots from each of two random runs have been selected for brief analysis. Random "request for rendezvous" times from the Searcher to the ARIES occurred at 17 seconds during the

first simulation run and 80 seconds in the second one. The example plots selected show either the mission tracking performance, speed response during different phases of the mission, or the reduction of along-track distance error due to the sliding mode controller. For ease of comparison, the figures below are examined in pairs with the results of the 17 second run first, then the corresponding figure from the 80 second run, followed by some brief comments and observations.

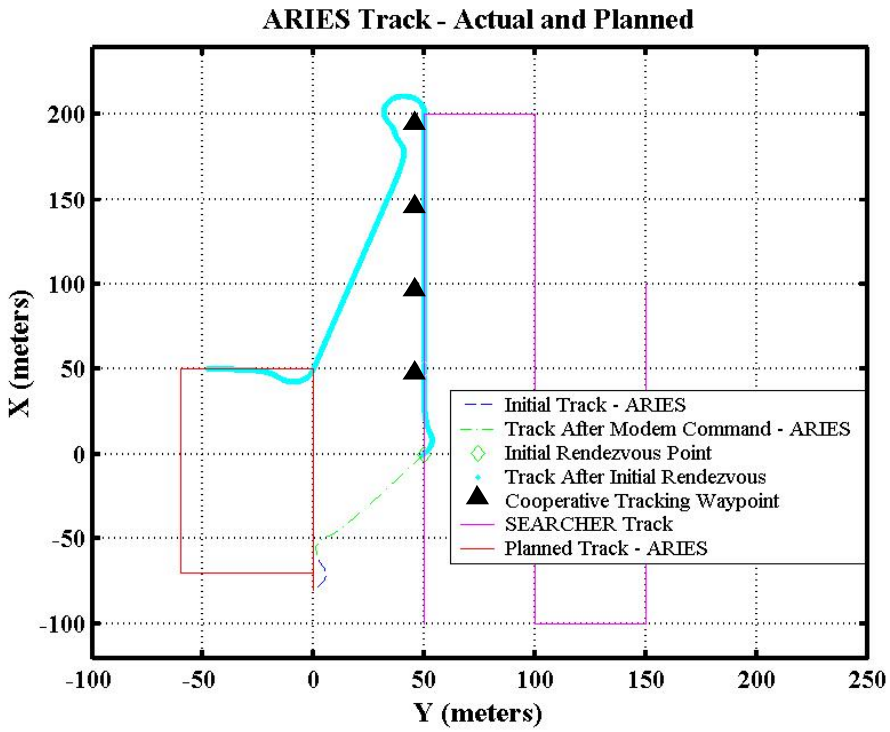


Figure 62. ARIES Track, 17 Second "Request to Rendezvous"

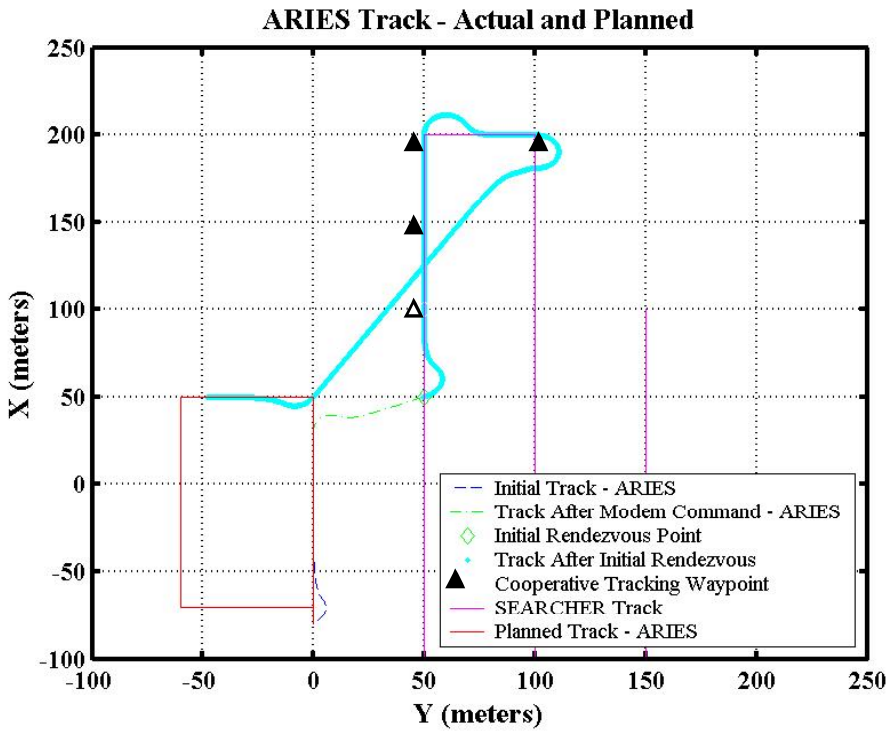


Figure 63. ARIES Track, 80 Second "Request to Rendezvous"

Figures 62 and 63 are an overall plan view of the two missions. As can be seen in each figure, ARIES begins its mission in the loiter box (Planned Track) as Searcher simultaneously begins the first part of its search pattern (SEARCHER Track), each vehicle initially at a constant speed determined by their individual mission plans. Searcher maintains a constant speed throughout the remainder of the search. Upon receiving the "request to rendezvous" call from the Searcher, ARIES departs from its planned track and proceeds to the waypoint calculated for rendezvous, altering its speed to ensure simultaneous arrival at the waypoint with Searcher. ARIES then dynamically maneuvers to parallel the Searcher track for four more waypoints (50 meters apart, annotated by the black triangles), allowing more than 200 meters for continuous data transfer. Throughout this mission phase, ARIES alters its speed to minimize the along-track position error between the two vehicles. After completing the data download, ARIES breaks away from the Searcher vehicle, resumes its initial constant speed, and returns to the loiter box.

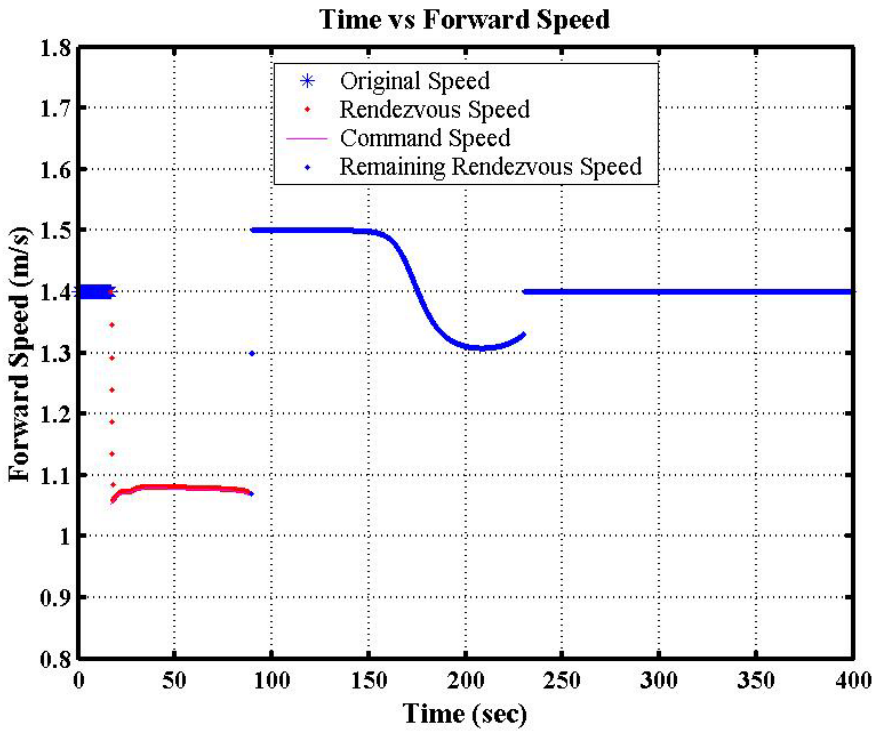


Figure 64. ARIES Speed Performance, 17 Second "Request to Rendezvous"

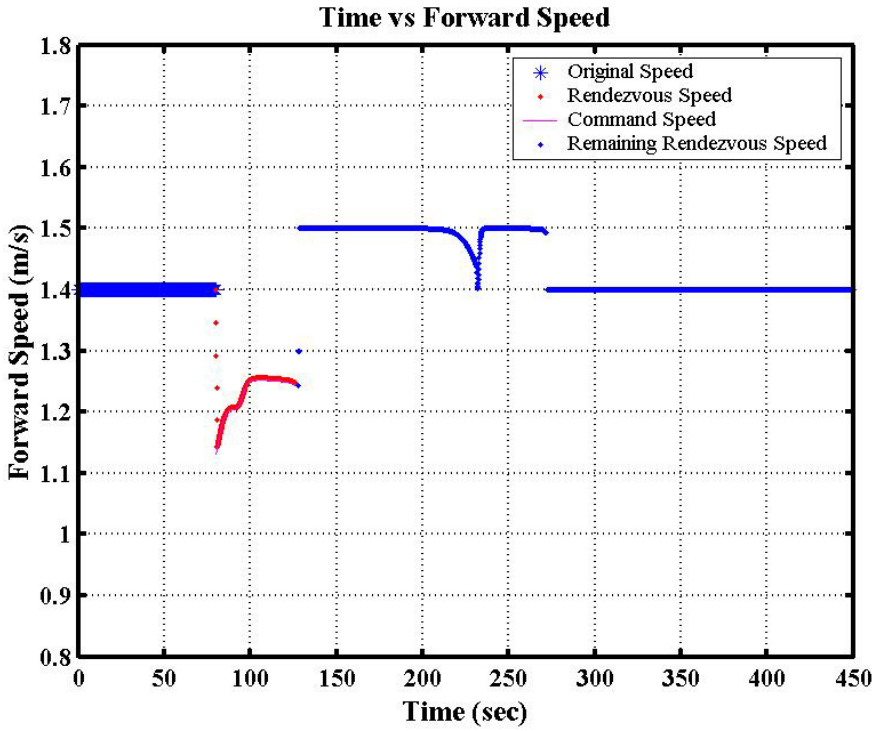


Figure 65. ARIES Speed Performance, 80 Second "Request to Rendezvous"

Recalling that the speed of the Searcher vehicle is held constant, Figures 64 and 65 capture the variation in speed of travel for ARIES throughout the entire mission. In

each case, ARIES begins the mission at a constant speed of 1.4 meters/second. However, when the request to rendezvous is received, ARIES uses the speed controller developed in [72] to rendezvous with the Searcher at a point determined by the "feasibility and waypoint selection" loop. As expected, note that the rendezvous speed required (red line in each figure) was substantially higher in Figure 65. The time and distance to reach the waypoint selected for rendezvous was 47.9 seconds and 54.1 meters whereas in Figure 64, the time and distance were 72.4 seconds and 76.3 meters. Next note the speed segment labeled "Remaining Rendezvous Speed". This is the phase of the mission where ARIES speed used the sliding mode controller moderated by position error between the two vehicles with the objective of minimizing the along-track distance error and ultimately matching vehicle speeds. The controller performed quite well in Figure 64, eventually settling out to the speed of the Searcher (1.3 meters/second). In Figure 65, the controller began to reach steady state and then increased speed again between the third and fourth cooperative tracking waypoints as the vehicle direction changed. Improvement in controller performance would be evident if the distance between waypoints was increased to a value greater than 50 meters, giving the controller enough time to reach steady state. The final portion of each speed plot shows ARIES returning to the loiter area at a constant speed of 1.4 meters/second.

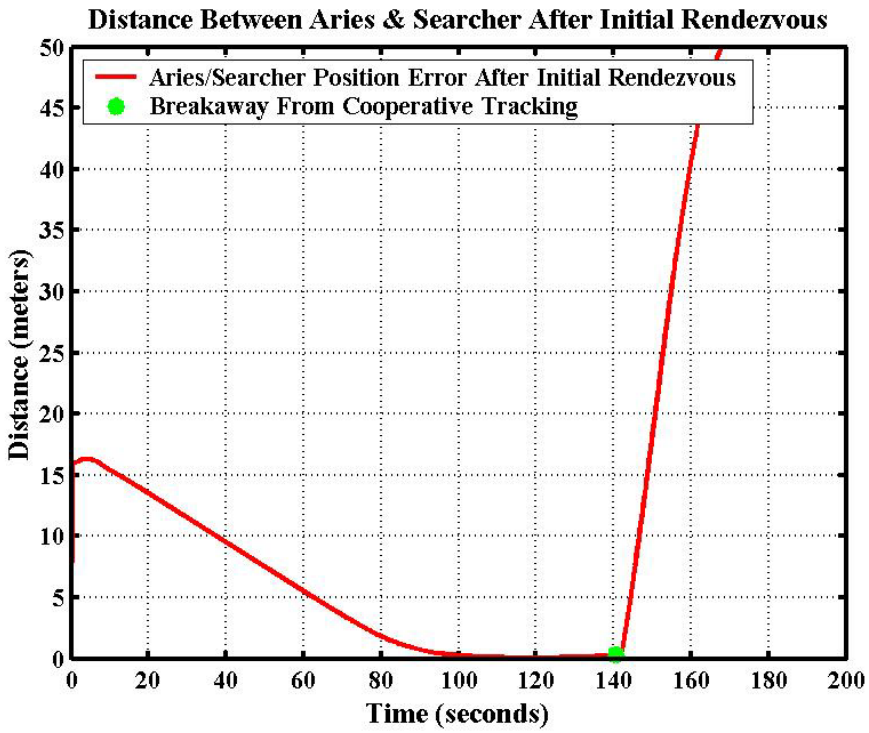


Figure 66. Cooperative Tracking Position Error, 17 Second "Request to Rendezvous"

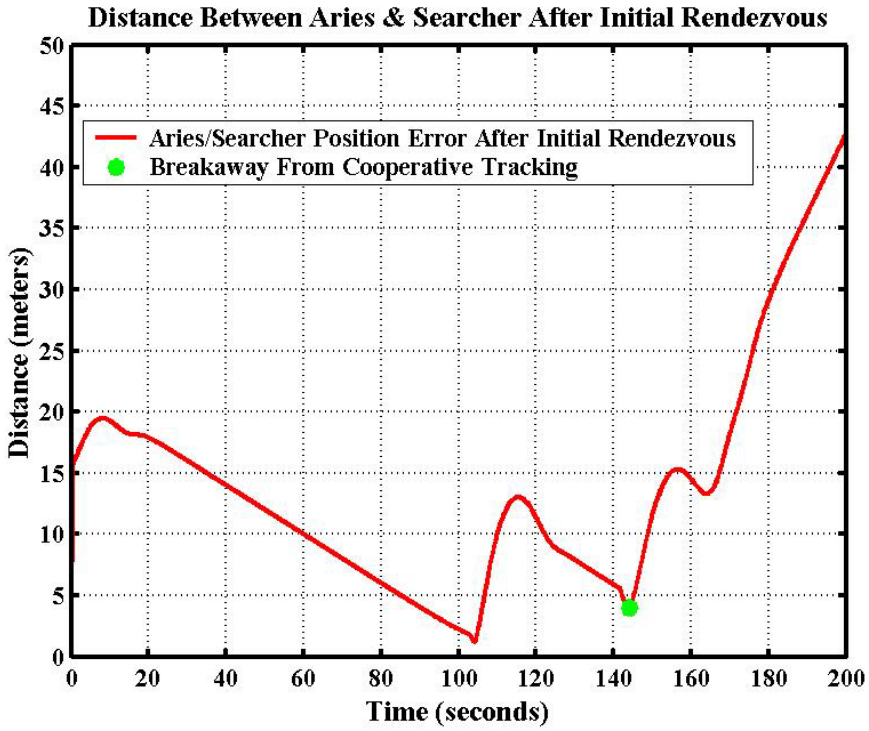


Figure 67. Cooperative Tracking Position Error, 80 Second "Request to Rendezvous"

Figures 66 and 67, plots of along-track distance between the two vehicles, are indicative of the speed controller performance during and subsequent to the cooperative tracking phase of the mission. The plots begin at the point of initial rendezvous, where, in each case, initial vehicle position error is less than 20 meters. This distance error is attributable to factors such as the acoustic delay from the "request to rendezvous," initial rendezvous point selection, and ARIES dynamics upon reaching the meeting point. Most notable, however, is the behavior of the controller for the subsequent cooperative tracking up to the point of break away (green star on figures). In both figures, the controller attempts to drive the along-track error to zero. Figure 66 is an example of the controller performance expected when the ARIES is allowed to reach steady state on a track. However, Figure 67 shows performance consistent with the speed plot in Figure 65. The along-track distance in Figure 67 is constantly decreasing toward steady state but has a marked increase between the third and fourth cooperative tracking waypoints as the vehicle direction changed and ARIES dynamically maneuvered to the new heading.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. SUMMARY AND CONCLUSIONS

A. CONTRIBUTIONS

The key result of these studies indicates one inescapable conclusion; Autonomous Underwater Vehicles operating in realistic, adverse Shallow Water and Very Shallow Water (VSW) environments are severely limited in the use of acoustic communication for either control or data transfer. Range and baud rates for successful and reliable communication are limited by multipath reflections, temporal and spatial variations in the acoustic channel, and background noise sources endemic to the specific operating area. While increased power at the source can overcome background noise, increased power also increases acoustic reflections in the multipath environment. The results of these studies suggests the possible need for a paradigm shift in the method used for AUV Shallow Water acoustic communication, particularly if high speed data transfer is the desired goal. Rather than continuing to design increasingly complex hardware to overcome signal distortion and other difficulties introduced by and inherent to the environment, perhaps the answer lies in actively decreasing the distance between the autonomous platforms through multi-vehicle cooperative control.

Specific contributions from this body of work that further the understanding of the Autonomous Underwater Vehicle field are innumerated below with brief amplifications or summaries.

1. Characterized Acoustic Modem Performance on a Dynamic, Operational AUV

Based on a review of the open literature, a comprehensive study on the bounds of AUV acoustic communications has not previously been done, particularly in a realistic, shallow water environment that closely models a mine warfare scenario. Therefore, commercially available modem systems were installed on the ARIES AUV and the acoustic transmission performance of each was thoroughly investigated. Two acoustic modems were analyzed: the first system from Florida Atlantic University (FAU) and the second from Benthos, Incorporated.

Four low speed modulation and coding configurations of the first modem system were evaluated in various channel geometries. The focus of the first seto of studies was to successfully demonstrate tactical control of the ARIES underwater vehicle, through both one way and two way acoustic communication. In addition to the development and demonstration of tactical control, this series of characterization studies also concluded that there is a minimum one-way transmission time, neglecting the channel delay due to sound velocity, about three seconds. Finally, empirical data indicated a clear trend toward improved communication performance when the AUV operated near the sea floor.

For the second modem system, eight configurations covering baud rates from 150-5120 bits/second (with various levels of data redundancy, multipath guard periods, and convolutional coding) were evaluated in several channel geometries. The primary focus of the second set of studies was to examine the limits of relatively high rate acoustic data transfer. Therefore, unlike the first modem configurations, this system employed phase shift keyed modulation in its two highest baud rates. A secondary goal in these investigations was demonstration of acoustic control at higher baud rates and a confirmation of the trend toward more reliable communication when the vehicle operated at lower altitudes in the water column.

2. Demonstrated Acoustic Control of AUV in Adverse Shallow Water Environment

AUV computer software was developed and compiled into the ARIES operating system to successfully alter missions and monitor vehicle status. Tactical acoustic control in the shallow water environment using baud rates from 55-220 bits/second with the FAU modem system was demonstrated. One-way control modes and operating set points were acoustically sent to, received by, and acted on by the ARIES. Two way queries sent by the topside tactical controller modem were responded to by the AUV. However, the success was somewhat limited. Based on the conservative limits of its demonstrated performance, the maximum operating range for effective acoustic control of a dynamic AUV with all integrated systems active was around 500 meters in the horizontal plane with ~85% reliability, at bit rates up to a maximum of 220 bits/second.

3. Demonstrated Limits on High Speed Asymmetric Data Transfer

The top combination of distance and rate observed for 100% reliable high speed data transfer in the most advantageous geometry was about 440 meters at 800 bits/second. In the most favorable channel geometry, the 1200 bits/second baud rate was about 90% reliable at ranges less than 250 meters. The high baud rates using phase shift keying (2560 and 5120 bits/second) were unable to communicate effectively in the shallow water environment. Even at ranges under 100 meters, communications success with the 2560 bits/second rate was less than 25% reliable and no coherent replies were received when using the 5120 bits/second baud rate. The maximum range where any communication was possible with even the most robust data transfer rate (150 bits/second) was 685 meters. Assuming the most conservative limits to account for uncertain vehicle orientation and channel geometry during acoustic communication, the maximum operating range for effective high speed data transfer in shallow water from a dynamic AUV with all integrated systems active was about 300 meters in the horizontal plane with nearly 100% reliability, at bit rates up to a maximum of 800 bits/second.

Note that although the actual ranges achieved during acoustic control and acoustic data transfer studies were relatively small, both modem systems performed quite well in the adverse, multipath, shallow water environment if considered in terms of non-dimensional distances as a function of water depth. The ARIES operated in a total water depth of about 15 meters but was able to effectively communicate, in all geometries, out to a minimum of 300 meters, a total of 20 water depths.

4. Developed Simulation Demonstrating Multi-Vehicle Cooperative Tracking Behavior for High Speed Acoustic Data Transfer

A simulation was developed to demonstrate the concept of multivehicle rendezvous and cooperative tracking for acoustic data transfer with the ultimate goal of transmitting the data collected to Warfare Commanders. The ARIES, acting as a server vehicle, made rendezvous with the Searcher AUV upon being acoustically summoned from its loiter area at a random time for download or transfer of data from the Searcher. After initial rendezvous, ARIES continued to parallel the Searcher's track for several more waypoints to allow sufficient time for the acoustic modems to pass information on the order of hundreds of kilobits, assuming a modest transmit baud rate of 1200

bits/second and vehicle operating speeds around 3 knots. Upon completion of data transfer, the server vehicle returned to its original loiter area to transmit the data collected by RF or satellite link. Two elements key to the simulation success were the development of a "feasibility and waypoint selection" loop to determine the waypoint that was most suitable for initial rendezvous and a sliding mode controller that altered ARIES cooperative tracking speed based on position error between the two vehicles.

B. FURTHER WORK

Two opportunities for continued research are clearly obvious from the studies in this dissertation.

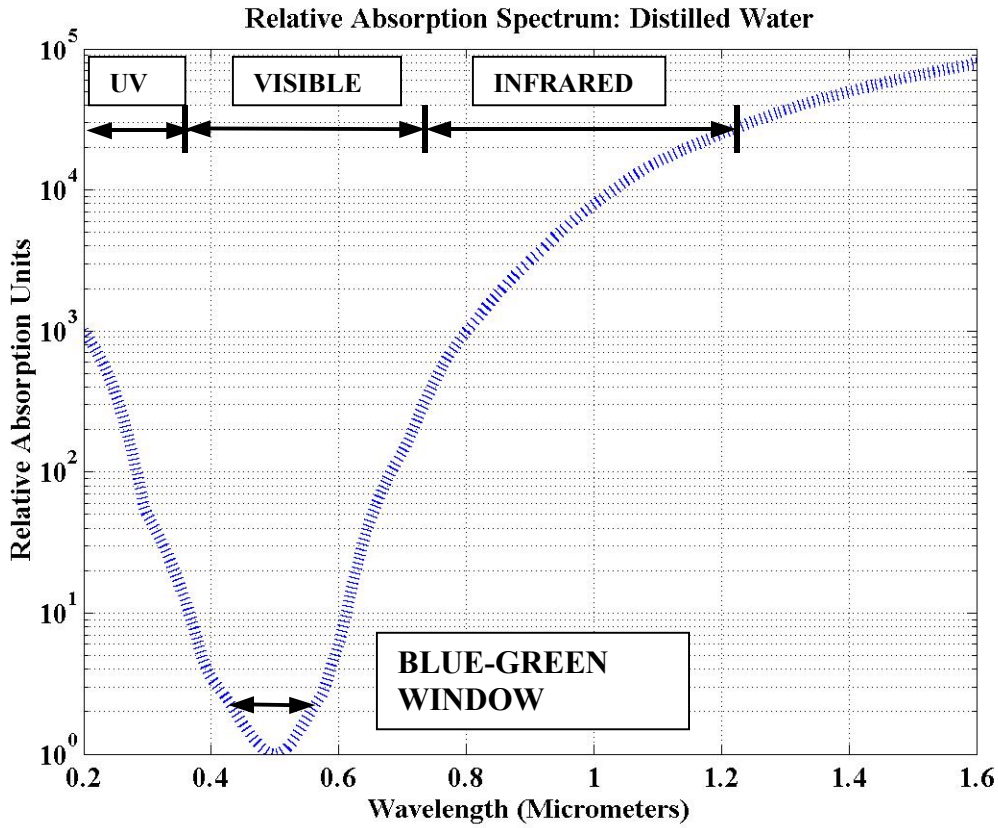
First and foremost is the need for further development of multi-vehicle rendezvous and cooperative tracking control to facilitate relatively high speed acoustic data transfer in the VSW and shallow water environment. Very shallow water communication links will always be limited by high backgraound noise and strong multipath reflections from bottom and surface bounces which make acoustic conditions dependent upon sea state and location. Therefore, reliable communication in adverse shallow water areas will be range-limited, particularly at higher baud rates. Research would necessarily involve simulations to optimize the time and energy required to rendezvous for data collection as well as multi-vehicle open water experiments to validate the results.

A second, and closely related, field for parallel pursuit is the development of a high speed modem in MHz region to take advantage of the short ranges anticipated during cooperative tracking. When operating at ranges on the order of tens of meters, ultrasonic frequency modem systems with large bandwidths and the attendant high data rates are feasible.

APPENDIX A

Acoustic transmission has been selected as the best alternative in the maritime environment, despite its well-known barriers (such as multi-path) to communication. Underwater propagation using Optical or Radio Freqency (RF) methods has several shortcomings, but the greatest among them is the high attenuation rate exhibited.

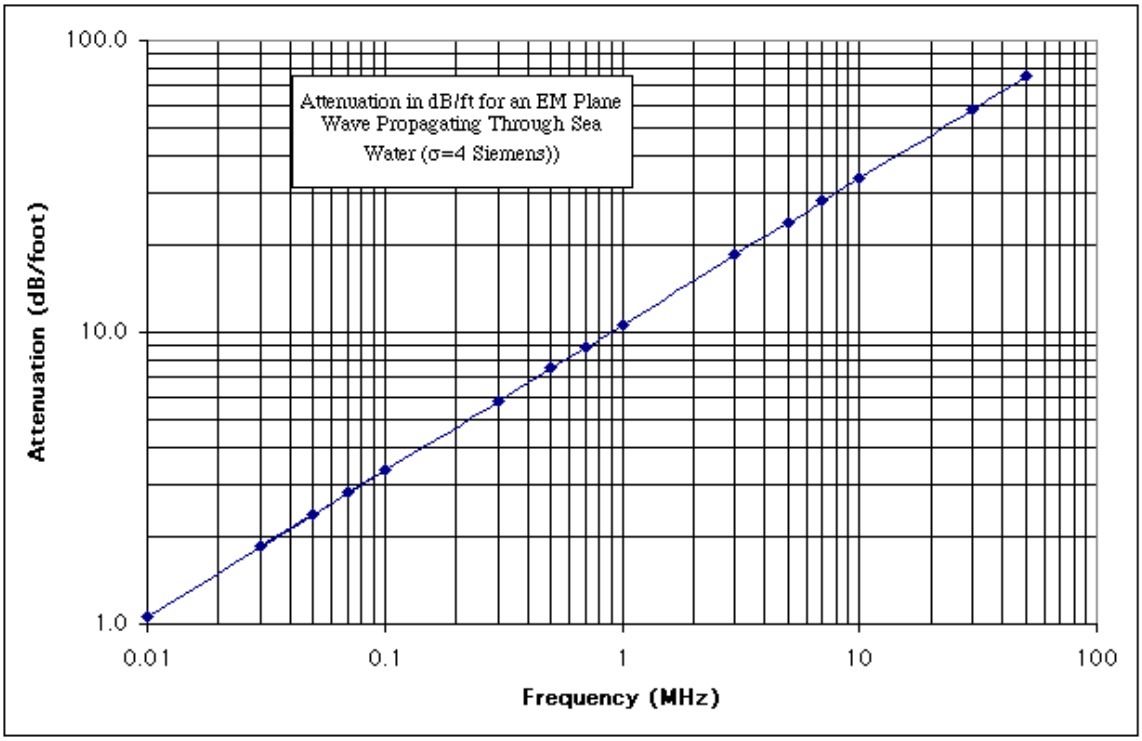
Optical (light or laser) transmission is heavily attenuated by water, primarily by absorption. The amount of attenuation is highly dependent on the wavelength. Shown below is the relative absorption in distilled water for wavelengths throughout the visible spectrum from [27].



Note that even though the blue-green laser has the lowest relative absorption of any optical source, typically, for economic reasons, much of the limited testing has been done with red diode laser. Even the best laser is still subject to variable attenuation

coefficients, restricted by water clarity, must be accurately pointed, and is limited to short distance communication on the order of meters rather than kilometers [28].

Electromagnetic or RF channel also suffer from high attenuation rates. The figure below, taken from [55], shows graphically the drastic attenuation expected during propagation of RF in sea water.



For comparison, in the approximate frequency range of one modem tested (about 30 KHz), an RF wave would be subject to about 2 dB/foot attenuation while the acoustic wave loss is less than 10 dB/kilometer.

APPENDIX B

Sample *script.d* file used to run typical ARIES mission. The exection process *Exec* parses and sequentially completes each action line.

```
MOTORS_ACTIVE
#
#                MinShlAlt ShlDepth_com ShlHeading_com ShlDuration
SET_SHOALING_PARAMETERS  1.0       1.0          315.0        30.0
USE_CTE_WAYPOINT_CONTROL Track.out
#
USE_TIME_BASED_CONTROL
#
SET_SCREW_SPEED              2.75 2.75
#
USE_SCREW_SPEED_CONTROL
#                            Depth (m)
SET_DEPTH                    0.0
#                            RudderAngle (Deg)
SET_FIXED_RUDDER_ANGLE       0.0
#
USE_FIXED_RUDDER_CONTROL
USE_FLIGHT_DEPTH_CONTROL
#                            Time (Sec)
SET_FLIGHT_DURATION          5.0
SET_SCREW_SPEED              0.0  0.0
#                            Time (Sec)
SET_FLIGHT_DURATION          15.0
SHUTDOWN
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C

Example of a typical ***Track.out*** file for a multi-legged mission. "33" is the number of waypoints or total lines in the mission. Columns 1 & 2 are the X and Y values in meters referenced to a local origin. Columns 3 & 4 are the port and starboard thruster speeds in volts. Column 5 is the control mode (0 for depth, 1 for altitude control). Columns 6 & 7 give the commanded altitude and commanded depth. Column 8 determines whether or not to do a GPS pop up on that leg (1 for yes, 0 for no). Column 9 is the watch radius in meters around the waypoint. Column 10 is the maximum time in seconds ARIES has to reach the waypoint before the mission goes into abort procedures.

```
33
550.00 300.00  2.75 2.75 0 8.0 1.5 1  25.0 10.00   40.0
580.00 300.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
580.00 330.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
550.00 330.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
550.00 300.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
580.00 300.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
580.00 330.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
550.00 330.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
550.00 300.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
580.00 300.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
580.00 330.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
550.00 330.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
550.00 300.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
580.00 300.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
580.00 330.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
550.00 330.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
550.00 300.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
580.00 300.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
580.00 330.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
550.00 330.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
550.00 300.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
580.00 300.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
580.00 330.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
550.00 330.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
550.00 300.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
580.00 300.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
580.00 330.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
550.00 330.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
550.00 300.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
580.00 300.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
580.00 330.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
550.00 330.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
550.00 300.00  2.75 2.75 0 8.0 1.5 0  25.0 10.00   45.0
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D

Below is a sample segment of the Exec.c code altered to allow the *Exec* process to take the actions necessary to modify ARIES behavior when commands where given using the FAU acoustic modem.

```
/* Read From FAU Modem */
      ReadFAUMShm(FAUM_Shmid,
                  &FAUM_Id,
                  &FAUM_NewData,
                  &NewControlMode,
                  &NewDepth_com,
                  &NewAlt_com,
                  &LoitLat,
                  &LoitLong,
                  &LoitTime1,
                  &LoitTimeOut,
                  &FAUMCom[0]);


      if(FAUM_NewData == 1)
      {
         /* We Have New Data From The Modem */
         /*printf("FAUMCom = %s\n",FAUMCom);*/

         if(!strcmp(FAUMCom,"ABORT"))
         {
            ResetFAUM_NewData(FAUM_Shmid);
            fprintf(AbortLogfp,"ABORT Sent From FAU Modem\n");
            fflush(AbortLogfp);
            Abort();
            break;
         }
         else if(!strcmp(FAUMCom,"CMODE"))
         {

                   /*  Reset Mission WayPoint Modes to NewControlMode
                    from Modem. Change dated 12/6/01-WM+AJH */

               for(k=WayPointIndex;k=NWayPoints;++k)
                   {

                     if((NewControlMode==0)|(NewControlMode==1))
                        {
                        WayControlMode[k]=NewControlMode;
                        }

                   }

           printf("Inside FAUM CMODE\n");
         }
         else if(!strcmp(FAUMCom,"GOTOL"))
```

```
         {
            printf("Inside FAUM GOTOL\n");
            /* Make Sure LoitLat and LoitLong are Non-Zero */
            if(LoitLat != 0.0 & LoitLong != 0.0)
            {

               /* Convert Lat0 & Long0 to dd.dddd */
               LatD  = (double) ((int) (Lat0/100.0));
               LongD = (double) ((int) (Long0/100.0));

               LatDeg0  = LatD  + (Lat0  - LatD*100.0)/60.0;
               LongDeg0 = LongD + (Long0 - LongD*100.0)/60.0;

               /* Convert LoitLat & LoitLong to Meters */
               /* printf("LoitLat = %f LoitLong =
%f\n",LoitLat,LoitLong); */

               /* Convert to dd.dddd */
               LatD  = (double) ((int) (LoitLat/100.0));
               LongD = (double) ((int) (LoitLong/100.0));

               LoitLatDeg  = LatD  + (LoitLat  - LatD*100.0)/60.0;
               LoitLongDeg = LongD + (LoitLong - LongD*100.0)/60.0;

               /* Convert LoitLat and LoitLong to X, Y */
               /* and OverWrite Existing WayPoint Parameters */

            }
            else
            {
               ResetFAUM_NewData(FAUM_Shmid);
               fprintf(AbortLogfp,"GOTOL Error LoitLat = %f, LoitLong =
%f\n",
                            LoitLat,LoitLong);
               fflush(AbortLogfp);
               Abort();
               break;
            }
         }
         else if(!strcmp(FAUMCom,"DEPTH"))
         {

           /*  Take Action to change Depth command - Changed 120601
WM+AJH */
                        FLIGHT_DEPTH_CONTROL =TRUE;
                        FLIGHT_ALTITUDE_CONTROL = FALSE;
                        for(k=WayPointIndex;k=NWayPoints;++k)
                           {

WayPointDepth_com[WayPointIndex]=NewDepth_com;
                            }

                        Depth_com=NewDepth_com;
                  WriteContLogFile("Depth_com changed to =
%f,%f\n",Depth_com,t);
```

```
                printf("Inside FAUM DEPTH\n");
        }
        else if(!strcmp(FAUMCom,"ALTIT"))
        {
         /*  Take Action to change Altitude command - Changed 120601
WM+AJH */
                        FLIGHT_DEPTH_CONTROL =FALSE;
                        FLIGHT_ALTITUDE_CONTROL = TRUE;
                        for(k=WayPointIndex;k=NWayPoints;++k)
                           {
                            WayPointAlt_com[WayPointIndex]=NewAlt_com;
                            }
                        Alt_com=NewAlt_com;

        printf("Inside FAUM ALTIT\n");
        }
        else
        {

        }
        ResetFAUM_NewData(FAUM_Shmid);
      }
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E

The following is an excerpt of the fm.c source code to show how strings were interpreted, parsed, built, and sent. The *fm* process was a result of compiling the "C" code from fm.c with fmf.c, a separate piece of source code containing functions.

```
/* Get Latest State Info */
            ReadExecShm(Exec_Shmid,&Exec_Id,&X,&Y,&psi,&Depth,&Alt,
                    &WayPointIndex,&WayControlMode,&WayDepth_com,
                    &WayAlt_com,&Lat0,&Long0);

            /* printf("Lat0 = %f Long0 = %f\n",Lat0,Long0); */
            printf("X = %f Y = %f Exec_Id = %d\n",X,Y,Exec_Id);
            printf("Alt = %f, Depth = %f\n",Alt,Depth);
            /* DeWrap psi to 0 - 360 */
            Heading = Rad2Deg*psi;
            while(fabs(Heading) > 360.0)
            {
               Heading = Heading - dsign(Heading)*360.0;
            }

            if(Heading < 0.0)
            {
               Heading = Heading + 360.0;
            }
            /* Deterime what Type of Message it is, Set or Query */
            /* Strip off the Header and Command */

            sscanf(FAUString,"%s %s",&Header[0],&Command[0]);
            printf("Headr = %s\n",Header);
            printf("Command = %s\n",Command);

            switch(Header[5])
            {
               case 'S':
                  /* This is a Set Command */
                  printf("This is a Set Command = %s\n",Command);
                  if(!strcmp(Command,"ABORT"))
                  {

/* Action added to abort command in fm.c 18Jul02 WJM */

                     StopScrewMotors();

                     printf("ABORT Was Sent\n");
                  }
                  else if(!strcmp(Command,"CMODE"))
                  {
                     printf("CMODE Was Sent\n");
                     sscanf(FAUString,"%s %s %d",&Header[0],
                                          &Command[0],
                                          &WayControlMode);
                     printf("WayControlMode = %d\n",WayControlMode);
```

```
            NewControlMode = WayControlMode;

         }
         else if(!strcmp(Command,"GOTOL"))
         {
            sscanf(FAUString,"%s %s %lf %c %lf %c %lf %lf",
                    &Header[0],&Command[0],
                    &LoitLat,&LatSector,
                    &LoitLong,&LongSector,&LoitTime,
                    &LoitTimeOut);
            /* Fix the Sign Here */
            if(LatSector == 'S')  LoitLat = -LoitLat;
            if(LongSector == 'W') LoitLong = -LoitLong;

         }
         else if(!strcmp(Command,"DEPTH"))
         {
            sscanf(FAUString,"%s %s %lf",&Header[0],
                 &Command[0],&Depth_com);
            printf("Depth_com = %f\n",Depth_com);
            FLIGHT_DEPTH_CONTROL      = TRUE;
            FLIGHT_ALTITUDE_CONTROL   = FALSE;


            NewDepth_com=Depth_com;
         }
         else if(!strcmp(Command,"ALTIT"))
         {
            sscanf(FAUString,"%s %s %lf",&Header[0],
                  &Command[0],&Alt_com);
            printf("Alt_com = %f\n",Alt_com);
            FLIGHT_ALTITUDE_CONTROL   = TRUE;
            FLIGHT_DEPTH_CONTROL      = FALSE;

            NewAlt_com = Alt_com;

          }
         else
         {
            printf("Unrecognized Set Command\n");
         }

      break;

      case 'Q':
         /* This is a Query Command */
         printf("This is a Query Command = %s\n",Command);
         if(!strcmp(Command,"POSIT"))
         {
            printf("Vehicle Position is Requested\n");
            /* Package the Response in A String */
         }

         /* Missing WAYPT block added by WJM 121101*/
```

```
                    else if(!strcmp(Command,"WAYPT"))
                    {
                       printf("Vehicle Waypoint is Requested\n");
                       /* Package the Response in a String */
                    }


                    else if(!strcmp(Command,"CMODE"))
                    {
                       printf("Vehicle Control Mode is Requested\n");
                       /* Package the Response in A String */
                       switch(WayControlMode)
                       {
                          case 0: /* Depth Control */
                             sprintf(FAUResString,"%d %f",
                                WayControlMode,Depth_com);
                             printf("WayControlMode = %d, Depth_com =
%f\n",
                                      WayControlMode,Depth_com);
                          break;

                          case 1: /* Altitude Control */
                             sprintf(FAUResString,"%d %f",
                                  WayControlMode,Alt_com);
                             printf("WayControlMode = %d, Alt_com =
%f\n",
                                      WayControlMode,Alt_com);
                          break;
                       }
                    }

/* Added 3 strings 22Mar02 WJM */

                else if(!strcmp(Command,"SMALL"))
                    {
                       printf("1 Character string Requested\n");
                       /* Package the Response in A String */
                    }


                    else if(!strcmp(Command,"MIDDL"))
                    {
                       printf("61 Character string Requested\n");
                       /* Package the Response in a String */
                    }


                    else if(!strcmp(Command,"LARGE"))
                    {
                       printf("80 Character string Requested\n");
                       /* Package the Response in A String */

                    }
```

```
                  else
                  {
                     printf("Unrecognized Query Command\n");
                  }

               break;

               default: /* UnKnown Command */
                  printf("UnKnown Command Header[5] = %c\n",
                              Header[5]);
                  BAD_COMMAND = TRUE;
                   /*        sprintf(FAUResString,"%21c","COMMAND NOT
RECOGNIZED");
                     ModemSendString(FAUResString);*/
               break;
            }

            if(!BAD_COMMAND)
            {
               BAD_COMMAND = FALSE;

               switch(Header[5])
               {
                  case 'S':   /* Only Write to Shm if a Set is Used */

                     FAUM_NewData = TRUE;
                     sprintf(FAUMCom,"%s",Command);
                     printf("FAUMCom = %s\n",FAUMCom);
                     WriteFAUMShm(FAUM_Shmid,FAUM_Id,FAUM_NewData,
                          NewControlMode,
                        NewDepth_com,NewAlt_com,LoitLat,LoitLong,
                        LoitTime,LoitTimeOut,FAUMCom);
                  break;

                  case 'Q':  /* Write to Modem */

            /* Added write to modem shared memory of FAUM_Id so PE
updated for queries as well as set commands.  WJM 01 Mar 02 */

                 FAUM_NewData = FALSE;
                 sprintf(FAUMCom, "%s",Command);
                 printf("FAUMCom = %s\n",FAUMCom);
                 WriteFAUMShm(FAUM_Shmid, FAUM_Id,FAUM_NewData,
NewControlMode, NewDepth_com, NewAlt_com,LoitLat, LoitLong, LoitTime,
LoitTimeOut, FAUMCom);

                     for(k=0;k<80;++k)
                     {
                        POSITString[k] = ' ';
                     }
                     for(k=0;k<80;++k)
                     {
                        WAYPTString[k] = ' ';
                     }
                     for(k=0;k<80;++k)
```

```
                     {
                        CMODEString[k] = ' ';
                     }
{
                        SMALLString[k] = ' ';
                     }
                     for(k=0;k<80;++k)
                     {
                        MIDDLString[k] = ' ';
                     }
                     for(k=0;k<80;++k)
                     {
                        LARGEString[k] = ' ';
                     }

                     /* Build Strings To Send Out to Base */
                     /* Added ModemSendString and if statements to 3
response
          conditions on 121101 by WJM.  Added extra 3 character strings
22Mar02 */

                     if(!strcmp(Command,"POSIT"))
                     {
                     sprintf(POSITString,"%6.1f %6.1f %5.1f %4.1f
%4.1f",
                        X,Y,
                          Heading,Depth,Alt);

                     ModemSendString(POSITString);
                     break;
                     }

                     else if(!strcmp(Command,"WAYPT"))
                     {
                     sprintf(WAYPTString,"%6.1f %6.1f %d",X,Y,
                          WayPointIndex);

                     ModemSendString(WAYPTString);
                     break;
                     }


                     /* Changed CMODE Query to include 5 parameters
                        (WayControlMode, WayAlt_com, WayDepth_com,Alt,
Depth)
                        17 Dec 01 WJM */


                     else if(!strcmp(Command,"CMODE"))
                     {

                            sprintf(CMODEString,"%d %4.1f %4.1f %4.1f
%4.1f",

WayControlMode,WayAlt_com,WayDepth_com,Alt,Depth);
```

```
                    ModemSendString(CMODEString);
                    break;
                    }

                if(!strcmp(Command,"LARGE"))
                    {
                    sprintf(LARGEString,"%f %f %7.2f %6.2f %6.2f %s",

X,Y,Heading,Depth,Alt,"abcdefghijklmnopqrstuvwxyzabcdefghijklm");

                    ModemSendString(LARGEString);
                    break;
                    }

                    else if(!strcmp(Command,"SMALL"))
                    {
                    sprintf(SMALLString,"%d",
                         WayPointIndex);

                    ModemSendString(SMALLString);
                    break;
                    }

                    else if(!strcmp(Command,"MIDDL"))
                    {

                           sprintf(MIDDLString,"%d %5.2f %5.2f %5.2f
%5.2f %s",

WayControlMode,WayAlt_com,WayDepth_com,Alt,Depth,
                  "abcdefghijklmnopqrstuvwxyzabcdefghi");

                    ModemSendString(MIDDLString);
                    break;
                    }



                    else
                    {
                    }


                   break;
               }
            }
            ++FAUM_Id;


         }
     }
  }
   close(fid);
```

APPENDIX F

This excerpt contains much of the C code from fm.c used to add an emergency abort feature to the *fm* process.

```
/* Added below functions to enable
 StopScrewMotors in the event of Exec lockup 19Jul02  WJM */


void InitRubyDac()
{

/*
    Configuration Byte for Input/Output Operation

    Hex    Port A    Port B    Port C (Both Halves)
    ----------------------------------------------

    9b     Input     Input     Input
    92     Input     Input     Output
    99     Input     Output    Input
    90     Input     Output    Output
    8b     Output    Input     Input
    82     Output    Input     Output
    89     Output    Output    Input
    80     Output    Output    Output
*/

   outp(RUBY_DAC_ADDR + 0x0009,0x0); /* Reset the Board */

   /* Initialize Digital I/O Register Record Bits High */
   RubyPortA_Reg = 0xff;
   RubyPortB_Reg = 0xff;
   RubyPortC_Reg = 0xff;

   outp(RUBY_DAC_ADDR+15,0x80); /* Control Register (Configured all  */
                                /*                   Ports for       */
                                /*                   OutPut Mode     */
   outp(RUBY_DAC_ADDR+12,0xff); /* Port A */
   outp(RUBY_DAC_ADDR+13,0xff); /* Port B */
   outp(RUBY_DAC_ADDR+14,0xff); /* Port C */
   /* Enable for Writing or Reading */

}
/*

   RubyDac(Ch,Volt) -- Writes Voltage to Dac Channel 'Ch'
                       (allowable channels 0-7)

        0    -->     -5 Vdc Output
     4096    -->     +5 Vdc Output

                  Pin Number
```

```
      Channel 7 --> 16*    *15 <-- Ground
      Channel 6 --> 14*    *13 <-- Ground
      Channel 5 --> 12*    *11 <-- Ground
      Channel 4 --> 10*    *9  <-- Ground
      Channel 3 -->  8*    *7  <-- Ground
      Channel 2 -->  6*    *5  <-- Ground
      Channel 1 -->  4*    *3  <-- Ground
      Channel 0 -->  2*    *1  <-- Ground
*/

int RubyDac(Ch,Volt)
   int Ch;
   double Volt;
{
   int val;

   if(Ch<0 || Ch>7)
   {
      printf("Invalid Channel in RubyDac(%d,%f)\n",Ch,Volt);
      return(-1);
   }

   val = (4095.0/10.0)*Volt + 2048; /* For +-5 V Setting */
   if(val < 0) val = 0;
   if(val > 4095) val = 4095;

   outp(RUBY_DAC_ADDR+RUBY_DAC_LSB_OFFSET,(val & 0x00ff));  /* Write
LSB */
   outp(RUBY_DAC_ADDR+Ch,(val >> 8) & 0x00ff);              /* Write
MSB */
   if(Ch <= 3) inp(RUBY_DAC_ADDR+0x0009);           /* Trigger Ch 0-3
*/

   return(0);

} /* RubyDac */

void StopScrewMotors()
{
   RubyDac(0,0.0); /* Left Screw  */
   RubyDac(1,0.0); /* Right Screw */
}
```

APPENDIX G

A sample of the simulation output written to the screen in MatLab for a run where Searcher requested ARIES to rendezvous after 80 seconds is listed below:

```
Time (sec) when Searcher modem calls Aries =  80.0
PLOT_PART = 0
Searcher X,Y coords (in meters) when modem calls Aries are  3.8, 50.0
Mission Feasible
Mission Feasible for Deceleration
Rendezvous time (seconds) = 47.9
Rendezvous Distance (meters) = 54.1
PLOT_PART = 0
Rendezvous Point Reached
Rendezvous X,Y coords (in meters) = 50.0, 50.0
PLOT_PART = 0
Aries X,Y coords (in meters) at next waypoint are 98.0, 49.8
Aries X,Y coords (in meters) at next waypoint are 148.1, 50.0
Aries X,Y coords (in meters) at next waypoint are 198.2, 50.0
Aries X,Y coords (in meters) at next waypoint are 199.9, 98.1
Aries X,Y coords (in meters) at next waypoint are 51.6,  1.0
Aries X,Y coords (in meters) at next waypoint are 50.1, -48.0
PLOT_PART = 0
Current plot held
Current plot released
Current plot held
Current plot released
Current plot held
Current plot released
Current plot held
Current plot released
Current plot held
Current plot released
Current plot held
Current plot released
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H

The complete version of "final_data_xfer_11Dec_19Apr.m," the MatLab simulation of cooperative AUV rendezvous and tracking for data transfer, is included in this appendix.

```
% Marr 11DEC02 & 19APR03

% Program final_data_xfer_11Dec_19Apr.m  Simulation demonstrates Aries making
% rendezvous with Searcher after being summoned at a random time to download
% sonar/video data from Searcher.  Upon completion of initial rendezvous,
% Aries continues to parallel Searcher for four more waypoints (time for
% modems to pass the information) then breaks away and Aries returns
% to its original initial loiter area.  Searcher waypoints 50m apart.
% Major challenges/improvements of code include
% addition of random modem call generator and development of the combined
% feasibility loop; accounting for "decreasing" waypoint values; achieving
% successful rendezvous at higher speeds; calculation of Searcher position at
% all times (kinematics only); designing a speed controller
% with the ability to use position feedback (SMC) to close the distance
% after initial rendezvous (figure 3); and plotting improvements.
% Began simulation work July 02.  Basic Aries dynamics were originally
% from Marco work (Waypoint.m).  The updated Aries speed controller
% for initial rendezvous leg based on control law and the longitudinal
% equation of motion from Keegan work(finalrendezvous.m).

whitebg('k');
clear all
TRUE  = 1;
FALSE = 0;

%Added a random time for Searcher to call for rendezvous WJM
T_call = round(rand * 80);
disp(sprintf('Time (sec) when Searcher modem calls Aries =  %4.1f',...
        T_call))

% Converts Degrees to Radians & Radians to Degrees
DegRad = pi/180;
RadDeg = 180/pi;

% State Model Parameters
W   = 600.0;                    % Weight in LB
U = 1.4*3.28;                   % Forward Speed in ft/s (1.4 m/s)
g = 32.174;                     % Gravity in ft/sec^2
Boy = 602.0;                    % Bouyancy
xg  = 0.125/12.0;               % Longitudianl CG
m = W/g;                        % Mass
rho = 1.9903;                   % Density of Seawater in slugs/ft^3
L = 10;                         % Length in ft of ARIES
Iz = (1/12)*m*(1.33^2 + 10^2);  % Approx. Using I = 1/12*m*(a^2 + b^2)

% where a is width & b is length
Iz = Iz*5.0;

% Coefficients
Yv_dot = -0.03430*(rho/2)*L^3;  % Added Mass in Sway Coefficient.
Yr_dot = -0.00178*(rho/2)*L^4;  % Added Mass in Yaw Coefficient.
Yv = -0.10700*(rho/2)*L^2;      % Coeff. of Sway Force induced by Side Slip
Yr = 0.01187*(rho/2)*L^3;       % Coeff. of Sway Force induced by Yaw
Ydrs = (0.01241*(rho/2)*L^2)/2.0; % Since Bow & Stern Lower Rudders Removed
Ydrb = (0.01241*(rho/2)*L^2)/2.0; % won't use these equations
Nv_dot = -0.00178*(rho/2)*L^4; % Added Mass Moment of Inertia in Sway Coeff
Nr_dot = -Iz;                  % Added Mass Moment of Inertia in Yaw Coeff
```

```
Nv = -0.00769*(rho/2)*L^3;     % Coeff. of Sway Moment from Side Slip
Nr = -0.00390*(rho/2)*L^4;     % Coeff. of Sway Moment from Yaw

% Below Modified on 7/12/00 The 3.5 and 3.4167 is the Moment Arm Length
% in Feet - Since Bow & Stern Lower Rudders Removed
Ndrs = -0.01241*(rho/2)*(L^2)*(3.5)/2.0;
Ndrb = 0.01241*(rho/2)*(L^2)*(3.4167)/2.0;

% Combining Stern & Bow Rudder Effectivness
Ndr = Ndrs - Ndrb;
Ydr = Ydrs - Ydrb;                          % Cancel Out

% Matrices
m1 = m - Yv_dot;
m2 = m*xg - Yr_dot;
m3 = m*xg - Nv_dot;
m4 = Iz - Nr_dot;
Y1 = Yv;
Y2 = Yr;
Y3 = U^2*Ydr;
N1 = Nv;
N2 = Nr;
N3 = U^2*Ndr;
A = [Y1*U Y2*U;N1*U N2*U];
B = [Y3 N3]';
M = [m1 m2;m3 m4];
A1 = inv(M)*A;
B1 = inv(M)*B;
AO = [A1(1,1) A1(1,2) 0;
    A1(2,1) A1(2,2) 0;
    0 1 0];
BO = [B1;0];
dt = 0.125;
t = [0:dt:1000]';
size(t);
% Added for Searcher 1 Aug 02 WJM
dtr = 0.125;
tr = [0:dtr:900]'; % Length of Searcher mission
size(tr);

% Set initial conditions
start=10;
v(1)   = 0.0;                       % Initial Side Slip Velocity
r(1)   = 0.0;                       % Initial Yaw
U(1)   = 1.4*3.28;                  % Initial Forward Speed Aries (ft/sec)
rRM(1) = r(1);
psi(1) = 50.0*DegRad;               % Initial Heading of ARIES
X(1) = -80.0;                       % Initial Position in meters
Y(1) = 0.0;
ucom=[];
U_Searcher(1) = 1.3;           % Searcher speed in m/s
X_Searcher(1) = -100;          % Initial Searcher position in meters
Y_Searcher(1) = 50;


% This data from track.out file for ARIES in Waiting Pattern
No_tracks=4;                        % Sets # of Tracks = # of Rows
Track=[ 50.0  0.0   2.75 2.75  0  1.25  1.00 0 25.00 8.00 40.00
    50.0 -60.0  2.75 2.75  0  1.25  1.00 0 25.00 8.00 200.00
    -70.0 -60.0  2.75 2.75  0  1.25  1.00 0 25.00 2.00 200.00
    -70.0  0.0  2.75 2.75  0  1.25  1.00 0 25.00 2.00 40.00];
track=Track(:,1:2);             % Defines track as Track(X,Y)
SurfaceTime = Track(:,9);       % Col 9 of Track is Surface Time for Pop-up
SurfPhase   = Track(:,8);       % Col 8 of Track designates if Pop-up

% This is the Searcher Search Pattern
Y_SEARCHER = [50 50 50 50 50 50 50 100 100 100 100 100 100 100 150 150 150 150
150];
```

```
        X_SEARCHER = [-100 -50 0 50 100 150 200 200 150 100 50 0 -50 -100 -100 -50 0 50
100];

        % Set up Searcher track waypoints 30 Jul 02 WJM
        No_Searcher_tracks=19;
        for k=1:No_Searcher_tracks,
            X_Way_Searcher_c(k) = X_SEARCHER(k);
            Y_Way_Searcher_c(k) = Y_SEARCHER(k);
        end;


         % Loop computes values for
         % each data point of Searcher corresponding to a time value
         % Completely dependent upon tracks selected so will need
         % changed if pattern or #waypts on tracks changes. Current
         % track 900 meters long.
            Searcher_time(1)=900/U_Searcher(1);
            rtt=[1:dtr:Searcher_time]';
            size(rtt);
                for rt=1: round(0.3333333*length(rtt)),
                    X_Searcher(rt+1)=X_Searcher(rt) + U_Searcher(1)*dtr;
                    Y_Searcher(rt+1)=Y_Searcher(rt);
                    Searcher_psi(rt)=0;
                end
                for rt= round(0.3333333*length(rtt)):round(0.3888889*length(rtt)),
                    X_Searcher(rt+1)=X_Searcher(rt) ;
                    Y_Searcher(rt+1)=Y_Searcher(rt)+ U_Searcher(1)*dtr;
                    Searcher_psi(rt)=pi/2;
                end
                for rt= round(0.3888889*length(rtt)):round(0.7222222*length(rtt)),
                    X_Searcher(rt+1)=X_Searcher(rt) - U_Searcher(1)*dtr;
                    Y_Searcher(rt+1)=Y_Searcher(rt);
                    Searcher_psi(rt)=pi;
                end
                for rt= round(0.7222222*length(rtt)):round(0.7777777*length(rtt)),
                    X_Searcher(rt+1)=X_Searcher(rt) ;
                    Y_Searcher(rt+1)=Y_Searcher(rt)+ U_Searcher(1)*dtr;
                    Searcher_psi(rt)=pi/2;
                end
                for rt= round(0.7777778*length(rtt)):round(length(rtt)),
                    X_Searcher(rt+1)=X_Searcher(rt) + U_Searcher(1)*dtr;
                    Y_Searcher(rt+1)=Y_Searcher(rt);
                    Searcher_psi(rt)=0;
                end

                 for rt= round(length(rtt)):length(t)-1,
                    X_Searcher(rt+1)=X_Searcher(rt) ;
                    Y_Searcher(rt+1)=Y_Searcher(rt);
                    Searcher_psi(rt)=0;
                end


        % Read in ARIES way points from track data: assumes track is loaded
        for j=1:No_tracks,
            X_Way_c(j)     = track(j,1);
            Y_Way_c(j)     = track(j,2);
        end;

        % Set start position
        PrevX_Way_c(1) = -80.0;         % meters
        PrevY_Way_c(1) =  00.0;         % meters
        r_com = 0.0;
        W_R = 10.0;                     % Sets initial Watch Radius (meters)
        a = -.3;
        b = (9/24)*a;
        x(:,1) = [v(1);r(1);psi(1)];
        % Below are in British Units for CTE Sliding Mode
        Lam1 = 2.0;
        Lam2 = 1.0;
```

```
Eta_FlightHeading = 1.0;
Phi_FlightHeading = 0.5;
% Below for tanh
Eta_CTE = 0.1;
Eta_CTE_Min = 1.0;
Phi_CTE = 0.5;
Uc = [];
Vc = [];
PLOT_PART = 0; disp(sprintf('PLOT_PART = 0'));

% Total Track Length between initial waypoint and waypoint (1)
SegLen(1) = sqrt((X_Way_c(1)-PrevX_Way_c(1))^2+(Y_Way_c(1)...
    -PrevY_Way_c(1))^2);
% Track Angle of first track - Eq (11)
psi_track(1) = atan2(Y_Way_c(1)-PrevY_Way_c(1),X_Way_c(1)-PrevX_Way_c(1));
% Computes track lengths and track angles for each track
for j=2:No_tracks,
    SegLen(j) = sqrt((X_Way_c(j)-X_Way_c(j-1))^2+(Y_Way_c(j)-...
        Y_Way_c(j-1))^2);
    psi_track(j) = atan2(Y_Way_c(j)-Y_Way_c(j-1),X_Way_c(j)-X_Way_c(j-1));
end;
j=1;
Sigma = [];
Depth_com = [];
dr=[];
drl = [];
drl(1) = 0.0;
Depth_com(1) = 5.0;
WayPointVertDist_com = [5.0 5.0 5.0 5.0 5.0];
SURFACE_TIMER_ACTIVE = FALSE;

% Starts a loop that computes values for each data point corresponding to
% a time value (in this case, every 0.125 seconds from 1 to 1000 seconds)
for i=1:length(t)-1,
    Depth_com(i) = WayPointVertDist_com(j);
    % Difference between current vehicle position & the next
    % waypoint Eq(13)
    X_Way_Error(i) = X_Way_c(j) - X(i);
    Y_Way_Error(i) = Y_Way_c(j) - Y(i);
    % DeWrap psi to within +/- 2.0*pi; Makes Heading Angle to lie between
    % 0-360 degrees
    psi_cont(i) = psi(i);
    while(abs(psi_cont(i)) > 2.0*pi)
        psi_cont(i) = psi_cont(i) - sign(psi_cont(i))*2.0*pi;
    end;
    % Cross Track Heading Error
    psi_errorCTE(i) = psi_cont(i) - psi_track(j);
    % DeWrap psi_error to within +/- pi;  Normalized to Lie between +/- 180
    % degrees
    while(abs(psi_errorCTE(i)) > pi)
        psi_errorCTE(i) = psi_errorCTE(i) - sign(psi_errorCTE(i))*2.0*pi;
    end;
    % **  Always Calculate this (Local angle side slip)
    Beta = v(i)/U(i);
    %  Beta = 0.0;
    cpsi_e = cos(psi_errorCTE(i)+Beta);
    spsi_e = sin(psi_errorCTE(i)+Beta);
    % Distance to the ith way point projected to the track line S(t)i -
    % Eq (14)
    s(i) = [X_Way_Error(i),Y_Way_Error(i)]*[(X_Way_c(j)-...
            PrevX_Way_c(j)),(Y_Way_c(j)-PrevY_Way_c(j))]';
    % s is distance to go projected to track line
    % (goes from 0-100%L) - Eq (14)
    s(i) = s(i)/SegLen(j);
    Ratio=(1.0-s(i)/SegLen(j))*100.0;     % Ranges from 0-100% of SegLen
    % Radial distance to go to next WP
    ss(i) = sqrt(X_Way_Error(i)^2 + Y_Way_Error(i)^2);
    % dp is angle between line of sight and current track line
    dp(i) = atan2( (Y_Way_c(j)-PrevY_Way_c(j)),(X_Way_c(j)-...
```

```
    PrevX_Way_c(j)) )- atan2( Y_Way_Error(i),X_Way_Error(i) );
if(dp(i) > pi),
    dp(i) = dp(i) - 2.0*pi;
end;
% Cross Track Error Definition
cte(i) = s(i)*sin(dp(i));
% If the magnitude of the CTE Heading exceeds 40 degrees, a LOS
% Controller is used.
if( abs(psi_errorCTE(i)) >= 40.0*pi/180.0 | s(i) < 0.0 ),
    LOS(i) = 1;
    psi_comLOS = atan2(Y_Way_Error(i),X_Way_Error(i));
    psi_errorLOS(i) = psi_comLOS - psi_cont(i);
    % LOS Error
    if(abs(psi_errorLOS(i)) > pi),
        psi_errorLOS(i) = psi_errorLOS(i) - 2.0*pi*psi_errorLOS(i)...
            /abs(psi_errorLOS(i));
    end;
    Sigma_FlightHeading = 0.9499*(r_com - r(i)) + 0.1701*...
        psi_errorLOS(i);
    dr(i) = -1.5435*( 2.5394*r(i)+ Eta_FlightHeading*tanh...
        (Sigma_FlightHeading/Phi_FlightHeading));
else
    % Use CTE Controller if CTE Heading is less than 40 degrees
    LOS(i) = 0;
    if(cpsi_e ~= 0.0),                % Trap Div. by Zero !
        % SMC Soln Sliding Surface
        Sigma(i) = U(i)*rRM(i)*cpsi_e + Lam1*U(i)*spsi_e + 3.28*Lam2...
            *cte(i);
        % Rudder Input
        dr(i) = (1.0/(U(i)*b*cpsi_e))*(-U(i)*a*rRM(i)*cpsi_e + U(i)*...
            rRM(i)^2*spsi_e - Lam1*U(i)*rRM(i)*cpsi_e - Lam2*U(i)*...
            spsi_e - Eta_CTE*(Sigma(i)/Phi_CTE));
    else
        dr(i) = dr(i-1);
    end;
end;                                % End of CTE Controller

% Surface Phase Logic (Independent of LOS or CTE)
if(SurfPhase(j) == TRUE)
    if(SURFACE_TIMER_ACTIVE == FALSE)
        if(Ratio > 40.0)
            % Start a Timer
            SURFACE_TIMER_ACTIVE = TRUE;
            Depth_com(i) = 0.0;
            SurfaceWait = SurfaceTime(j) + t(i);
            SurfaceWait
        end;
    end;
end;
if(SURFACE_TIMER_ACTIVE == TRUE)
    if(t(i) >= SurfaceWait)
        SURFACE_TIMER_ACTIVE = FALSE;
        Depth_com(i) = WayPointVertDist_com(j);
        SurfPhase(j) = 0;
    else
        Depth_com(i) = 0.0;
    end;
end;
if(abs(dr(i)) > 0.4)
    dr(i) = 0.4*sign(dr(i));
end;

% Jay Johnson Model
Yv  = -68.16;
Yr  = 406.3;
Ydr = 70.0;
Nv  = -10.89;
Nr  = -88.34;
Ndr = -35.47;
```

```
            MY = 456.76;
            IN = 215;
            M = diag([MY,IN,1]);
            AA = [Yv,Yr,0;Nv,Nr,0;0,1,0];
            BB = [Ydr;Ndr;0];
            A  = inv(M)*AA;
            B  = inv(M)*BB;


            x_dot(:,i+1) = [ A(1,1)*v(i) + A(1,2)*r(i) + B(1)*dr(i);
                A(2,1)*v(i) + A(2,2)*r(i) + B(2)*dr(i);
                r(i)];
            x(:,i+1) = x(:,i)+dt*x_dot(:,i);
            v(i+1)   = x(1,i+1)/12;
            r(i+1)   = x(2,i+1);
            U(i+1) = 1.4*3.28;         % Constant speed of 2.72 knots
            psi(i+1) = x(3,i+1);
            rRM(i+1) = r(i+1);

            Uc = 0.0;
            Vc = 0.0;

            % Kinematics
            X(i+1) = X(i) + (Uc + (U(i)/3.28)*cos(psi(i)) - v(i)/3.28*sin(psi(i))...
                )*dt;
            Y(i+1) = Y(i) + (Vc + (U(i)/3.28)*sin(psi(i)) + v(i)/3.28*cos(psi(i))...
                )*dt;

            %*********************************************************************

            % Proceed to rendezvous @ T_call seconds
            if i == T_call/dt,
                disp(sprintf('Searcher X,Y coords (in meters) when modem calls Aries are
%4.1f, %4.1f',...
                X_Searcher(T_call/dt), Y_Searcher(T_call/dt)))
                break;
            end

            %*********************************************************************

            % Check to See if we are Within the Watch_Radius
            if(sqrt(X_Way_Error(i)^2.0 + Y_Way_Error(i)^2.0) <= W_R | s(i) < 0.0),
                disp(sprintf('WayPoint %d Reached',j));
                if(j==No_tracks),
                    PLOT_PART = 1;
                    disp(sprintf('PLOT_PART = 1'));
                    break;
                end;
                PrevX_Way_c(j+1) = X_Way_c(j);
                PrevY_Way_c(j+1) = Y_Way_c(j);
                j=j+1;
            end;

        end; %end of i loop



        %*********************************************************************
        % Rendezvous Point Information
        % Only need X & Y of Searcher as inputs
        % plus the rendezvous time (calculated as equal to time Searcher will reach
        % next waypoint).  Combined "feasibility & rendezvous point selection
        % loop" determines rendezvous time for Searcher to make
        % next waypoint, then distance between Aries posit when call received and
        % Searcher' next waypoint. Finally checks if enough time for Aries to get there
        % given its speed and acceleration constraints.  Selects rendezvous point as
        % location immediately beyond minimum acceptable distance.

        for m=1:No_Searcher_tracks;
```

```
                if X_Way_Searcher_c(m) >= X_Searcher((T_call)/dtr + 24),
                    T_rend(1)  =  (abs(X_Way_Searcher_c(m+1)  -  X_Searcher((T_call)/dtr  +
24))+...
                        (Y_Way_Searcher_c(m+1)      -      Y_Searcher((T_call)/dtr      +
24)))/(U_Searcher(1));
                    % Rendezvous time to remaining Searcher waypoints.  24 is 3 sec
                    % delay time to rcv modem call
                    New_RendLen(m)=sqrt((X_Way_Searcher_c(m+1)-X(i))^2+...
                        (Y_Way_Searcher_c(m+1) - Y(i))^2);
                    new_time = T_rend(1);
            % Determines if the Mission is Feasible - Can Distance be covered in
            % time required traveling at maximum speed of 3.5 knots or is time
            % required to be at the rendezvous too much for vehicle travelling at
            % minimum speed of 0.5 knots? Also determines if there is
            % enough length to achieve deceleration or acceleration
                        if new_time > ((New_RendLen(m))/(1.8)) &...
                        new_time < ((New_RendLen(m))/(0.2571)),
                        disp('Mission Feasible');
                        decel_Len = abs(((0.2571)^2 - (1.4)^2)/(2*0.0249));
                        accel_Len = ((1.8)^2 - (1.4)^2)/(2*0.03);
                        time_of_decel = 114;         %Time to decel from 1.4 m/s to 0.2571
m/s
                        time_left = T_rend(1)-time_of_decel;
                        distance_remaining = (New_RendLen(m))-decel_Len;
                            if distance_remaining > 0.2571 * time_left,
                            disp('Mission Feasible for Deceleration');
                            T_rend(1)        =        (abs(X_Way_Searcher_c(m+1)        -
X_Searcher((T_call)/...
                               dtr + 24))+(Y_Way_Searcher_c(m+1) - Y_Searcher((T_call)/...
                               dtr + 24)))/(U_Searcher(1))+20/U_Searcher(1);
                            disp(sprintf('Rendezvous time (seconds) = %4.1f',...
                                T_rend(1)))
                            break;
                            end;
                        end;
                        % else if case for when Searcher current position is past
                        % waypoint being considered
                    elseif X_Way_Searcher_c(m) < X_Searcher((T_call)/dtr + 24),
                T_rend(1) = (abs(X_Way_Searcher_c(m+1) - X_Searcher((T_call)/dtr + 24))+...
                        (Y_Way_Searcher_c(m+1)      -      Y_Searcher((T_call)/dtr      +
24)))/(U_Searcher(1));
                    % Rendezvous time to remaining Searcher waypoints.  24 is 3 sec
                    % delay time to rcv modem call
                        New_RendLen(m)=sqrt((X_Way_Searcher_c(m+1)-X(i))^2+...
                       (Y_Way_Searcher_c(m+1) - Y(i))^2);
                    new_time = T_rend(1);
                        if new_time > ((New_RendLen(m))/(1.8)) &...
                        new_time < ((New_RendLen(m))/(0.2571)),
                        disp('Mission Feasible');
                        decel_Len = abs(((0.2571)^2 - (1.4)^2)/(2*0.0249));
                        accel_Len = ((1.8)^2 - (1.4)^2)/(2*0.03);
                        time_of_decel = 114;         %Time to decel from 1.4 m/s to 0.2571
m/s
                        time_left = T_rend(1)-time_of_decel;
                        distance_remaining = (New_RendLen(m))-decel_Len;
                            if distance_remaining > 0.2571 * time_left,
                            disp('Mission Feasible for Deceleration');
                            T_rend(1)        =        (abs(X_Way_Searcher_c(m+1)        -
X_Searcher((T_call)/...
                               dtr + 24))+(Y_Way_Searcher_c(m+1) - Y_Searcher((T_call)/...
                               dtr + 24)))/(U_Searcher(1))+20/U_Searcher(1);
                            disp(sprintf('Rendezvous time (seconds) = %4.1f',...
                                T_rend(1)))
                            break;
                            end;
                        end;

                end;  % end of "if X_Way_Searcher"
            end;  % end of "for m" loop
```

```
% Once feasible rendezvous decided, want to make rendezvous then continue parallel
% to Searcher for additional waypoints to download data
if m > 13,
    disp(sprintf('Cannot Complete Mission'));
    break;
else
New_No_Tracks = 7;
new_track = [X_Way_Searcher_c(m+1) Y_Way_Searcher_c(m+1)
                X_Way_Searcher_c(m+2) Y_Way_Searcher_c(m+2)
                X_Way_Searcher_c(m+3) Y_Way_Searcher_c(m+3)
                X_Way_Searcher_c(m+4) Y_Way_Searcher_c(m+4)
                X_Way_Searcher_c(m+5) Y_Way_Searcher_c(m+5)
                X_Way_c(1) Y_Way_c(1)
                X_Way_c(1) Y_Way_c(1)-50];
end
    for jj = 1:New_No_Tracks,
        New_X_Way_c(jj) = new_track(jj,1);
        New_Y_Way_c(jj) = new_track(jj,2);
    end;

    New_PrevX_Way_c(1) = X(i+1);        % Sets Abort Posit as start of new
    New_PrevY_Way_c(1) = Y(i+1);        % track for commencing rendezvous actions.
    % Total Track Length between abort point and rendezvous point
    New_SegLen(1) = sqrt((New_X_Way_c(1)-New_PrevX_Way_c(1))^2+...
        (New_Y_Way_c(1) - New_PrevY_Way_c(1))^2);
    disp(sprintf('Rendezvous Distance (meters) = %4.1f',...
                        New_SegLen(1)))
    % Track Angle of track between abort point and rendezvous point
    new_psi_track(1) = atan2(New_Y_Way_c(1)-New_PrevY_Way_c(1),...
        New_X_Way_c(1)-New_PrevX_Way_c(1));
    new_time(1) = T_rend(1); % Desired Time to Rendezvous in seconds

 jj=1;
if j == No_tracks,
    disp(sprintf('Mission Complete'));
else
    new_r_com = 0.0;
    new_v(1) = v(i+1);
    new_r(1) = r(i+1);
    new_rRM(1) = new_r(1);
    new_psi(1) = psi(i+1);
    New_X(1) = X(i+1);
    New_Y(1) = Y(i+1);
    %*********************************************************************
    new_U(1) = U(i+1)/3.28;      % Sets Initial Rendezvous Speed to 1.4 m/s
    % new_U in meters/sec.
    overall_distance_travelled(1) = 0;
    distance_travelled(1) = 0;
    time_used(1) = 0;
    time_remaining(1) = new_time(1);
    real_time(1) = 0;
    accel(1) = 0.03;                 % Sets Acceleration to 0.03 m/s^2
    decel(1) = 0.0249;              % Sets Deceleration to 0.0249 m/s^2
    ncom(1) = 12;

    %*********************************************************************
    % Below for tanh
    new_Eta_CTE = 0.1;
    new_Eta_CTE_Min = 1.0;
    new_Phi_CTE = 0.5;
    PLOT_PART = 0; disp(sprintf('PLOT_PART = 0'));
    new_x(:,1) = [new_v(1); new_r(1); new_psi(1)];

    %*********************************************************************
    % Starts loop that computes values for each data point corresponding to
    % a time value along new track
    new_Sigma = [];
    new_Depth_com = [];
```

```
new_dr=[];
new_drl = []; n = [];
new_drl(1) = 0.0;
new_Depth_com(1) = 5.0;
new_WayPointVertDist_com = [5.0 5.0 5.0 5.0 5.0];
new_SURFACE_TIMER_ACTIVE = FALSE;
tt = [t(i+1):dt:4*new_time]';
size(tt);

% Start of Loop for Rendezvous Point
for ii = 1:length(tt)-1,
    new_Depth_com(ii) = new_WayPointVertDist_com(jj);
    New_X_Way_Error(ii) = New_X_Way_c(jj) - New_X(ii);
    New_Y_Way_Error(ii) = New_Y_Way_c(jj) - New_Y(ii);
    new_psi_cont(ii) = new_psi(ii);
    while(abs(new_psi_cont(ii)) > 2.0*pi)
        new_psi_cont(ii) = new_psi_cont(ii) - sign(new_psi_cont(ii))*...
            2.0*pi;
    end;
    % Cross Track Heading Error
    new_psi_errorCTE(ii) = new_psi_cont(ii) - new_psi_track(jj);
    % DeWrap psi_error to within +/- pi;  Normalized to Lie between +/-
    % 180 degrees
    while(abs(new_psi_errorCTE(ii)) > pi)
        new_psi_errorCTE(ii) = new_psi_errorCTE(ii) - sign(...
            new_psi_errorCTE(ii))*2.0*pi;
    end;
    % **  Always Calculate this (Local angle side slip)
    new_Beta = new_v(ii)/new_U(ii);
    new_cpsi_e = cos(new_psi_errorCTE(ii)+new_Beta);
    new_spsi_e = sin(new_psi_errorCTE(ii)+new_Beta);
    % Distance to the ith way point projected to the track line S(t)i
    new_s(ii) = [New_X_Way_Error(ii),New_Y_Way_Error(ii)]*[(...
            New_X_Way_c(jj)-New_PrevX_Way_c(jj)),(New_Y_Way_c(jj)...
            -New_PrevY_Way_c(jj))]';
    %*******************************************************************
    % Calculates the Overall Distance Travelled, Distance Travelled,
    % Time Used Overall, Time Remaining, Switches velocity from 0.5
    % knots to 3.5 knots depending on Time Remaining and Distance
    % Remaining while taking into account acceleration/deceleration.
    if ii == 1,
        overall_distance_travelled(ii) = 0;
        distance_travelled(ii) = 0;
        time_used(ii) = 0;
        time_remaining(ii) = new_time(ii);
        real_time(ii) = 0;
    elseif ii == 2,
        overall_distance_travelled(ii) = New_SegLen(ii) - new_s(ii-1);
        distance_travelled(ii) = overall_distance_travelled(ii);
        time_remaining(ii) = time_remaining(ii-1)-dt;%time_used(ii);
        real_time(ii) = new_time(ii) - time_remaining(ii);
    else
        overall_distance_travelled(ii) = New_SegLen(ii) - new_s(ii-1);
        distance_travelled(ii) = overall_distance_travelled(ii) -...
            overall_distance_travelled(ii-1);
        time_used(ii) = distance_travelled(ii)/(new_U(ii)/3.28);
        time_remaining(ii) = time_remaining(ii-1) - dt;%time_used(ii);
        real_time(ii) = new_time(ii) - time_remaining(ii);
    end

    % s is distance to go projected to track line(goes from 0-100%L)
    new_s(ii) = new_s(ii)/New_SegLen(jj);
    if (time_remaining(ii)<0.125), disp('mission out of time'),...
            break,end;
    % Determines what speed to set the vehicle at
    %time_available(ii) = new_s(ii)/(new_U(ii)/3.28);
    ucom(ii)=new_s(ii)/time_remaining(ii);
    if ucom(ii) > 1.8 ;
        ucom(ii) = 1.8 ;         % Max velocity is 3.5 Knots
```

```
end
if ucom(ii) < 0.2571 ;
    ucom(ii) = 0.2571 ;      % Min velocity is 0.5 Knots
end


%*****************************************************************
% Control Law for Longitudinal Equation of Motion -Keegan
new_sigma(ii) = new_U(ii)-ucom(ii);
new_phi(ii) = 0.1;tau=0.1;ncommax=22;
ncom2(ii) = (1.39*(accel(ii)*0-800*tanh(new_sigma(ii)/...
    new_phi(ii))+67.74*(new_U(ii)*abs(new_U(ii)))));
ncom(ii)=sqrt(abs(ncom2(ii)))*sign(ncom2(ii));
% Limits propeller speed to 22 rps
if (abs(ncom(ii))>ncommax),
    ncom(ii)=ncommax*sign(ncom(ii));
end;
% always +ve in these simulations
% solve longitudinal dynamics
new_U(ii+1) = (0.004641*dt*(-10.5*(new_U(ii)*abs(new_U(ii)))+...
    0.155*(ncom(ii)*abs(ncom(ii))) - 0.05*ncom(ii)*new_U(ii)))...
    +new_U(ii);
New_SegLen(ii+1) = New_SegLen(ii);
new_time(ii+1) = new_time(ii);
accel(ii+1) = accel(ii);

%*****************************************************************
% Ranges from 0-100% of SegLen
Ratio=(1.0-new_s(ii)/New_SegLen(jj))*100.0;
% Radial distance to go to next WP
new_ss(ii) = sqrt(New_X_Way_Error(ii)^2 + New_Y_Way_Error(ii)^2);
% dp is angle between line of sight and current track line
new_dp(ii) = atan2( (New_Y_Way_c(jj)-New_PrevY_Way_c(jj)),(...
    New_X_Way_c(jj)-New_PrevX_Way_c(jj)) )- atan2...
    (New_Y_Way_Error(ii),New_X_Way_Error(ii) );
if(new_dp(ii) > pi),
    new_dp(ii) = new_dp(ii) - 2.0*pi;
end;
% Cross Track Error Definition - Keegan
new_cte(ii) = new_s(ii)*sin(new_dp(ii));
% If the magnitude of the CTE Heading exceeds 40 degrees, a
% LOS Controller is used.
if( abs(new_psi_errorCTE(ii)) >= 40.0*pi/180.0 | new_s(ii) < 0.0 ),
    new_LOS(ii) = 1;
    new_psi_comLOS = atan2(New_Y_Way_Error(ii),...
        New_X_Way_Error(ii));
    new_psi_errorLOS(ii) = new_psi_comLOS - new_psi_cont(ii);
    if(abs(new_psi_errorLOS(ii)) > pi),
        new_psi_errorLOS(ii) = new_psi_errorLOS(ii) - 2.0*pi*...
            new_psi_errorLOS(ii)/abs(new_psi_errorLOS(ii));
    end;
    new_Sigma_FlightHeading = 0.9499*(new_r_com - new_r(ii)) +...
        0.1701*new_psi_errorLOS(ii);
    new_dr(ii) = -1.5435*( 2.5394*new_r(ii)+ Eta_FlightHeading*...
        tanh(new_Sigma_FlightHeading/Phi_FlightHeading));
else
    % Use CTE Controller if CTE Heading is less than 40 degrees
    new_LOS(ii) = 0;
    if(new_cpsi_e ~= 0.0),                % Trap Div. by Zero !
        % SMC Soln Sliding Surface
        new_Sigma(ii) = new_U(ii)*3.28*new_rRM(ii)*new_cpsi_e...
            + Lam1*new_U(ii)*3.28*new_spsi_e + 3.28*Lam2*...
            new_cte(ii);
        new_dr(ii) = (1.0/(new_U(ii)*3.28*b*new_cpsi_e))*...
            (-new_U(ii)*3.28*a*new_rRM(ii)*new_cpsi_e +...
            new_U(ii)*3.28*new_rRM(ii)^2*new_spsi_e - ...
            Lam1*new_U(ii)*3.28*new_rRM(ii)*new_cpsi_e - ...
            Lam2*new_U(ii)*3.28*new_spsi_e -new_Eta_CTE*...
            (new_Sigma(ii)/new_Phi_CTE));
```

```
            else
                new_dr(ii) = new_dr(ii-1);
            end;
        end;                                % End of CTE Controller
        if(abs(new_dr(ii)) > 0.4)
            new_dr(ii) = 0.4*sign(new_dr(ii));
        end;

        new_x_dot(:,ii+1) = [ A(1,1)*new_v(ii) + A(1,2)*new_r(ii) + B(1)*...
                new_dr(ii); A(2,1)*new_v(ii) + A(2,2)*new_r(ii) + B(2)*...
                new_dr(ii); new_r(ii)];
        new_x(:,ii+1) = new_x(:,ii)+dt*new_x_dot(:,ii);
        new_v(ii+1)   = new_x(1,ii+1)/12;
        new_r(ii+1)   = new_x(2,ii+1);
        new_psi(ii+1) = new_x(3,ii+1);
        new_rRM(ii+1) = new_r(ii+1);
        Uc = 0.0;
        Vc = 0.0;

        % Kinematics note new_U is in meters /sec, new_v is in ft/sec
        % hold over from long time ago
        New_X(ii+1) = New_X(ii) + (Uc + (new_U(ii))*cos(new_psi(ii)) -...
            new_v(ii)/3.28*sin(new_psi(ii)) )*dt;
        New_Y(ii+1) = New_Y(ii) + (Vc + (new_U(ii))*sin(new_psi(ii))...
            + new_v(ii)/3.28*cos(new_psi(ii)) )*dt;
        % Check to See if we are Within the Watch_Radius (set to 1
        % meter here)
        if(sqrt(New_X_Way_Error(ii)^2.0 + New_Y_Way_Error(ii)^2.0)...
                <= 1 | new_s(ii) < 0.0),
            % Next line exists initial rendezvous
            % point if within Watch Radius.
            disp(sprintf('Rendezvous Point Reached'));
            disp(sprintf('Rendezvous X,Y coords (in meters) = %4.1f, %4.1f',...
        X_Way_Searcher_c(m+1), Y_Way_Searcher_c(m+1)))
            break;
        end
    end  %end of ii loop
end  % if j=No_Tracks


% Third loop conducts remaining waypoints after rendezvous with
% Searcher.  Involves reinitializing some values and running Aries
% dynamic model assuming matching speed to Searcher for additional
% waypoints by position feedback from Searcher as going.  WJM
% First ensures that if mission is out of time, will display
% that the mission cannot be completed.
 if (time_remaining(ii)<0.125), disp('Cannot complete mission'),...
                break,end;

if j == No_tracks,
    disp(sprintf('No Call for Rendezvous Received'));
else
% Set initial conditions
new2_v(1)= new_v(ii+1);             % Initial Side Slip Velocity
new2_r(1)   = new_r(ii+1);                      % Initial Yaw
new2_U(1)=new_U(ii+1);
% Added following definitions to use for position controller
% after rendezvous
new2_phi_posit = [];
new2_eta_posit = [];
delta_posit =[];          % Initialize values for speed control
posit_error =[];          % using position error.
new2_phi_posit(1) = 2.0;  % Adjustments for phi & eta depending on
new2_eta_posit(1) = 0.2;  % level of control required.
new2_rRM(1) = new_rRM(ii+1);
new2_psi(1) = new_psi(ii+1);               % Initial Heading of ARIES
new2_X(1) = New_X(ii+1);                       % Initial Position in meters
new2_Y(1) = New_Y(ii+1);
new2_r_com(1)= new_r_com;
```

```
        % Set start position
        New_PrevX_Way_c(2) = New_X(ii+1);    % In meters, sets initial rendezvous point
        New_PrevY_Way_c(2) = New_Y(ii+1);    % as beginning of "tunnel" to pass data.
        new2_x(:,1)=[new2_v(1);new2_r(1);new2_psi(1)];
        % Computes track length and track angle for remaining waypoints after rendezvous
        for jjj=2:New_No_Tracks,
            New2_SegLen(jjj)           =           sqrt((New_X_Way_c(jjj)-New_X_Way_c(jjj-
1))^2+(New_Y_Way_c(jjj)-...
                New_Y_Way_c(jjj-1))^2);
            new2_psi_track(jjj)          =          atan2(New_Y_Way_c(jjj)-New_Y_Way_c(jjj-
1),New_X_Way_c(jjj)-New_X_Way_c(jjj-1));
            % new_time is Searcher waypoint distances divided by Searcher speed
            new2_time(jjj)        =        (X_Way_Searcher_c(m+jjj)-X_Way_Searcher_c(m+jjj-
1))/U_Searcher(1);
        end;

        % Below are in British Units for CTE Sliding Mode
        Lam1 = 2.0;
        Lam2 = 1.0;
        Eta_FlightHeading = 1.0;
        Phi_FlightHeading = 0.5;
        % Below for tanh
        Eta_CTE = 0.1;
        Eta_CTE_Min = 1.0;
        Phi_CTE = 0.5;
        Uc = [];
        Vc = [];
        PLOT_PART = 0; disp(sprintf('PLOT_PART = 0'));
        jjj=2;
        new2_Sigma = [];
        new2_Depth_com = [];
        new2_dr=[];
        new2_drl = [];
        new2_drl(1) = 0.0;
        new2_Depth_com(1) = 5.0;
        new2_WayPointVertDist_com = [5.0 5.0 5.0 5.0 5.0];
        new_SURFACE_TIMER_ACTIVE = FALSE;
        newreal_time(1) = 0;
        cum_time = T_rend(1)+(T_call);
        ttt = [cum_time:dt:1000]';
        size(ttt);

        % Starts a loop that computes values for each data point corresponding to
        % remaining waypoints travelled (in this case, every 0.125 seconds)
        for iii=1:length(ttt)-1,
            newreal_time(iii+1) = newreal_time(iii)+dt;
            % Difference between current vehicle position & the next waypoint
            New2_X_Way_Error(iii) = New_X_Way_c(jjj) - new2_X(iii);
            New2_Y_Way_Error(iii) = New_Y_Way_c(jjj) - new2_Y(iii);
            % DeWrap psi to within +/- 2.0*pi; Makes Heading Angle to lie between
            % 0-360 degrees
            new2_psi_cont(iii) = new2_psi(iii);
            while(abs(new2_psi_cont(iii)) > 2.0*pi)
                new2_psi_cont(iii) = new2_psi_cont(iii) - sign(new2_psi_cont(iii))*2.0*pi;
            end;
            % Cross Track Heading Error
            new2_psi_errorCTE(iii) = new2_psi_cont(iii) - new2_psi_track(jjj);
            % DeWrap psi_error to within +/- pi;  Normalized to Lie between +/- 180
            % degrees
            while(abs(new2_psi_errorCTE(iii)) > pi)
                new2_psi_errorCTE(iii)         =         new2_psi_errorCTE(iii)         -
sign(new2_psi_errorCTE(iii))*2.0*pi;
            end;
            % **  Always Calculate this (Local angle side slip)
            new2_Beta = new2_v(iii)/new2_U(iii)*3.28;
            new2_cpsi_e = cos(new2_psi_errorCTE(iii)+Beta);
            new2_spsi_e = sin(new2_psi_errorCTE(iii)+Beta);
            % Distance to the ith way point projected to the track line S(t)i
```

```
            new2_s(iii)                                                                    =
[New2_X_Way_Error(iii),New2_Y_Way_Error(iii)]*[(New_X_Way_c(jjj)-...
                    New_PrevX_Way_c(jjj)),(New_Y_Way_c(jjj)-New_PrevY_Way_c(jjj))]';
            % new2_s is distance to go projected to track line
            % (goes from 0-100%L)
            new2_s(iii) = new2_s(iii)/New2_SegLen(jjj);
            Ratio=(1.0-new2_s(iii)/New2_SegLen(jjj))*100.0;       % Ranges from 0-100% of
SegLen
            % Radial distance to go to next WP
            new2_ss(iii) = sqrt(New2_X_Way_Error(iii)^2 + New2_Y_Way_Error(iii)^2);
            % dp is angle between line of sight and current track line
            new2_dp(iii)              =              atan2(              (New_Y_Way_c(jjj)-
New_PrevY_Way_c(jjj)),(New_X_Way_c(jjj)-...
                New_PrevX_Way_c(jjj))                       )-                       atan2(
New2_Y_Way_Error(iii),New2_X_Way_Error(iii) );
            if(new2_dp(iii) > pi),
                new2_dp(iii) = new2_dp(iii) - 2.0*pi;
            end;
            % Cross Track Error Definition
            new2_cte(iii) = new2_s(iii)*sin(new2_dp(iii));
            % If the magnitude of the CTE Heading exceeds 40 degrees, a LOS
            % Controller is used.
            if( abs(new2_psi_errorCTE(iii)) >= 40.0*pi/180.0 | new2_s(iii) < 0.0 ),
                new2_LOS(iii) = 1;
                new2_psi_comLOS = atan2(New2_Y_Way_Error(iii),New2_X_Way_Error(iii));
                new2_psi_errorLOS(iii) = new2_psi_comLOS - new2_psi_cont(iii);
                % LOS Error
                if(abs(new2_psi_errorLOS(iii)) > pi),
                   new2_psi_errorLOS(iii) = new2_psi_errorLOS(iii) - 2.0*pi*...
                      new2_psi_errorLOS(iii)/abs(new2_psi_errorLOS(iii));
                end;
                new2_Sigma_FlightHeading = 0.9499*(new2_r_com - new2_r(iii)) + 0.1701*...
                    new2_psi_errorLOS(iii);
                new2_dr(iii) = -1.5435*( 2.5394*new2_r(iii)+ Eta_FlightHeading*tanh...
                    (new2_Sigma_FlightHeading/Phi_FlightHeading));
            else
                % Use CTE Controller if CTE Heading is less than 40 degrees
                new2_LOS(iii) = 0;
                if(new2_cpsi_e ~= 0.0),                % Trap Div. by Zero !
                    % SMC Soln Sliding Surface
                    new2_Sigma(iii)    =    new2_U(iii)*3.28*new2_rRM(iii)*new2_cpsi_e    +
Lam1*...
                       new2_U(iii)*3.28*new2_spsi_e + 3.28*Lam2*new2_cte(iii);
                    % Rudder Input
                    new2_dr(iii)       =       (1.0/((new2_U(iii)*3.28)*b*new2_cpsi_e))*((-
new2_U(iii)*3.28)*a*...
                       new2_rRM(iii)*new2_cpsi_e                                          +
(new2_U(iii)*3.28)*new2_rRM(iii)^2*new2_spsi_e -...
                       Lam1*(new2_U(iii)*3.28)*new2_rRM(iii)*new2_cpsi_e                  -
Lam2*(new2_U(iii)*3.28)*...
                       new2_spsi_e - Eta_CTE*(new2_Sigma(iii)/Phi_CTE));
                else
                    new2_dr(iii) = new2_dr(iii-1);
                end;
            end;                                  % End of CTE Controller
            if(abs(new2_dr(iii)) > 0.4)
                new2_dr(iii) = 0.4*sign(new2_dr(iii));
            end;

            new2_x_dot(:,iii+1)   =   [   A(1,1)*new2_v(iii)   +   A(1,2)*new2_r(iii)   +
B(1)*new2_dr(iii);
                A(2,1)*new2_v(iii) + A(2,2)*new2_r(iii) + B(2)*new2_dr(iii);
                new2_r(iii)];
            new2_x(:,iii+1) = new2_x(:,iii)+dt*new2_x_dot(:,iii);
            new2_v(iii+1)   = new2_x(1,iii+1)/12;
            new2_r(iii+1)   = new2_x(2,iii+1);
            new2_psi(iii+1) = new2_x(3,iii+1);
            new2_rRM(iii+1) = new2_r(iii+1);
            U_Searcher(iii+1)  = U_Searcher(iii); % Searcher speed remains constant
```

```
        %*****************************************************************************
        % New block for speed controller based on position error.
        % Uses sliding mode control to close the distance error between Aries
        % and Searcher after making initial rendezvous thus alter
        % commanded speed for cooperative waypoint tracking.
        delta_posit(iii+1)=sqrt((X_Searcher(i+ii+iii)-new2_X(iii))^2+...
              (Y_Searcher(i+ii+iii)-new2_Y(iii))^2);
         posit_error(iii+1)=delta_posit(iii)*(cos(new2_dp(iii)));
           if jjj < 6,
           new2_U(iii+1)                =                U_Searcher(iii)                +
new2_eta_posit(iii)*tanh(posit_error(iii)/...
              new2_phi_posit(iii));
           else
              new2_U(iii+1)=1.4;
           end
              new2_eta_posit(iii+1) = new2_eta_posit(iii);
              new2_phi_posit(iii+1) = new2_phi_posit(iii);
           if new2_U(iii+1) > 1.8,
              new2_U(iii+1) = 1.8;    % Max speed 3.5 knots
           end
           if new2_U(iii+1) < 0.2571,
              new2_U(iii+1) = 0.2571;  % Min speed 0.5 knots
           end
           Uc = 0.0;
           Vc = 0.0;
           % Kinematics(ARIES)
           new2_X(iii+1)  =  new2_X(iii)  +  (Uc  +  (new2_U(iii))*cos(new2_psi(iii))  -
new2_v(iii)/3.28*sin(new2_psi(iii))...
               )*dt;
           new2_Y(iii+1)  =  new2_Y(iii)  +  (Vc  +  (new2_U(iii))*sin(new2_psi(iii))  +
new2_v(iii)/3.28*cos(new2_psi(iii))...
               )*dt;

           % Check to See if we are Within the Watch_Radius
           if(sqrt(New2_X_Way_Error(iii)^2.0   +   New2_Y_Way_Error(iii)^2.0)   <=   2   |
new2_s(iii) < 0.0),
               disp(sprintf('Aries  X,Y  coords  (in  meters)  at  next  waypoint  are  %4.1f,
%4.1f',new2_X(iii), new2_Y(iii)));
               if jjj==5,  % Marks where cooperative tracking ends 7 Oct
                   breaktime(1)=iii*dt;
                   breakaway(1)=posit_error(iii);
               end
               if(jjj==New_No_Tracks),
                   PLOT_PART = 0;
                   disp(sprintf('PLOT_PART = 0'));
                   break;
               end;
               New_PrevX_Way_c(jjj+1) = New_X_Way_c(jjj);
               New_PrevY_Way_c(jjj+1) = New_Y_Way_c(jjj);
               jjj=jjj+1;
           end;
       end; %end of iii loop
       end  % if j=No_Tracks


       %*****************************************************************************
       %*****************************************************************************
       % PLOTTING

       if PLOT_PART == 1,

           %****************

           % Plot of Time vs Rudder Angle & Vehicle Heading
           figure(1); clf
           orient tall
           plot(t([1:i+1]),psi*180/pi);
```

```
    hold;
    plot(t([1:i]),dr*180/pi,'r:');grid;
    title('Time vs Rudder Angle and Vehicle Heading');
    xlabel('Time (sec)'); ylabel('Rudder Angle/Vehicle Heading (degrees)');
    legend('Vehicle Heading', 'Rudder Angle');
    print -tiff -depsc figure1_wp5
    hold;zoom on;

    %***************

    % Plot of Time vs CTE, Distance to Go to Projected Track, Radial
    % Distance to Next Waypoint
    figure(2); clf
    orient tall
    plot(t([1:i]),cte);
    hold;
    plot(t([1:i]),s,'r:');
    plot(t([1:i]),ss,'g--');grid;
    title('Time vs Cross Track Error')
    xlabel('Time (sec)');ylabel('Distance (meters)');
    legend('Cross Track Error', 'Distance to Go Projected to Track', ...
        'Radial Distance to Go to Next Way Point');
    print -tiff -depsc figure2_wp5
    hold;zoom on;

    %***************

    %Plot of Time vs Forward Speed
    figure(3); clf
    orient tall
    plot(t([1:i+1]),U); grid;
    title('Time vs Forward Speed')
    xlabel('Time (sec)'); ylabel('Forward Speed (m/s)');

    %***************

elseif PLOT_PART == 0,

   % Plot of Time vs Rudder Angle & Vehicle Heading for Rendezvous Mission
    figure(4); clf
    orient tall
    plot(t([1:i+1]),psi*180/pi, 'g');
    hold;
    plot(real_time([1:ii]) + (T_call),new_psi([1:ii])*180/pi, 'g:');
    plot((cum_time+newreal_time([1:iii])),new2_psi([1:iii])*180/pi,'g.');
    plot(t([1:i]),dr*180/pi,'r');
    plot(real_time([1:ii-1])+ (T_call),new_dr([1:ii-1])*180/pi, 'r:');
    plot((cum_time+newreal_time([1:iii-1])),new2_dr([1:iii-1])*180/pi,'r-')
    title('Time vs Rudder Angle and Vehicle Heading');
    xlabel('Time (sec)'); ylabel('Rudder Angle/Vehicle Heading (degrees)');
    legend('Vehicle Heading Before Rendezvous Called',...
        'Vehicle Heading After Rendezvous Call',...
        'Vehicle Heading Concurrent with Searcher',...
        'Rudder Angle Before Rendezvous Called',...
        'Rudder Angle After Rendezvous Call',...
        'Rudder Angle Concurrent with Searcher');
    print -tiff -depsc figure4_wp5
    hold;grid;

    %***************

    % Plot of Time vs CTE, Distance to Go to Projected Track,
    % Radial Distance to Rendezvous & Subsequent Waypoints
    % for Rendezvous Mission.
    figure(5); clf
    orient tall
    plot(t([1:i]), cte, 'g');
    hold;
    plot(real_time([1:ii-1])+(T_call), new_cte([1:ii-1]), 'g:');
```

```
plot(cum_time+newreal_time([1:iii-1]),new2_cte([1:iii-1]),'g-');
plot(t([1:i]), s,'r');
plot(real_time([1:ii-1])+(T_call), new_s([1:ii-1]), 'r:');
plot(cum_time+newreal_time([1:iii-1]),new2_s([1:iii-1]),'r-');
plot(t([1:i-1]), ss([1:i-1]),'b');grid;
plot(real_time([1:ii-1])+ (T_call), new_ss([1:ii-1]), 'b:');
plot(cum_time+newreal_time([1:iii-1]),new2_ss([1:iii-1]),'b*');
title('Time vs Cross Track Error')
xlabel('Time (sec)'); ylabel('Distance (meters)');
legend('Cross Track Error Before Abort',...
   'Cross Track Error After Abort',...
   'Cross Track Error Additonal Waypoints',...
    'Distance to Go Projected to Track Before Abort', ...
    'Distance to Go Projected to Track After Abort',...
    'Distance to Go Projected to Track for Addional Waypoints',...
    'Radial Distance to Go to Next Way Point Before Abort',...
    'Radial Distance to Go to Next Way Point After Abort',...
    'Radial Distance to Go to Waypoints After Initial Rendezvous');
print -tiff -depsc figure5_wp5
hold;zoom on;

%***************

% Plot of Time vs Forward Speed
figure(6); clf
orient tall
i=T_call/dt;
plot(t([1:(i+1)]),U/3.28, 'b*');
hold;
plot(real_time([1:ii-1])+(T_call),new_U([1:ii-1]),'r.',...
    real_time([1:ii-1])+(T_call),ucom([1:ii-1]),'m');
plot(cum_time+newreal_time([1:iii-1]),new2_U([1:iii-1]),'b.'), grid;
title('Time vs Forward Speed')
xlabel('Time (sec)'); ylabel('Forward Speed (m/s)');
legend('Original Speed', 'Rendezvous Speed', 'Command Speed',...
    'Remaining Rendezvous Speed');
hold;
print -tiff -depsc figure6_wp5

%***************

% 3-D Plot
figure(7); clf
orient tall
i=T_call/dt;
plot3(Y, X, t([1:i+1]), 'b'); grid;
title('Time Space Plot')
xlabel('Y (meters)');
ylabel('X (meters)');
zlabel('Time (seconds)');
hold;
plot3(New_Y([1:ii]), New_X([1:ii]), real_time([1:ii])+(T_call), 'rx');
% Plots rendezvous point with a green diamond
plot3(New_Y_Way_c(1), New_X_Way_c(1), new_time(ii)+(T_call), 'gd');
plot3(new2_Y([1:iii]),new2_X([1:iii]),cum_time+newreal_time([1:iii]),'c*');
% Added additional blue diamonds to other waypoints shared between
% Aries and Searcher
plot3(New_Y_Way_c(2), New_X_Way_c(2), new_time(ii)+(T_call)...
   + (sqrt((X_Way_Searcher_c(m+3)-X_Way_Searcher_c(m+2))^2 +...
   (Y_Way_Searcher_c(m+3)-Y_Way_Searcher_c(m+2))^2)/U_Searcher(1)), 'bd');
plot3(New_Y_Way_c(3), New_X_Way_c(3), new_time(ii)+(T_call)+...
   + (sqrt((X_Way_Searcher_c(m+3)-X_Way_Searcher_c(m+2))^2 +...
   (Y_Way_Searcher_c(m+3)-Y_Way_Searcher_c(m+2))^2)/U_Searcher(1))+...
   (sqrt((X_Way_Searcher_c(m+2)-X_Way_Searcher_c(m+1))^2 +...
   (Y_Way_Searcher_c(m+2)-Y_Way_Searcher_c(m+1))^2)/U_Searcher(1)), 'bd');
 legend('Original Track', 'New Track to Rendezvous', 'Initial Rendezvous',...
    'New Track after Rendezvous','Cooperative Tracking')
print -tiff -depsc figure7_wp5
hold;
```

```
 end;

 %***************

if PLOT_PART == 1,

   % Plot of Actual Track and Planned Track
    figure(9); clf
    orient tall
    plot(Y,X,'b--');grid;                                % Actual Track
    title('ARIES Track - Actual and Planned');
    xlabel('Y (meters)');ylabel('X (meters)');
    hold;
    % Planned Track
    plot([Y_Way_c(1) PrevY_Way_c(1)],[X_Way_c(1) PrevX_Way_c(1)],'r');
    plot(Y_SEARCHER, X_SEARCHER, 'g');
    axis([-100 220 -120 60]);
    for ik=2:No_tracks,
        plot([Y_Way_c(ik) Y_Way_c(ik-1)],[X_Way_c(ik)...
                X_Way_c(ik-1)],'r');
    end;
    legend('Actual Track - ARIES', 'Planned Track - ARIES',...
        'SEARCHER Path',4);
    print -tiff -depsc figure9_wp5
    hold; zoom on

 elseif PLOT_PART == 0 | PLOT_PART == 2,

    %***************
    % Plot of Planned Track, Track after Mission Change, Rendezvous
    % Point and Initial Track
    figure(10); clf
    orient tall
    plot(Y,X,'b--');grid;
    title('ARIES Track - Actual and Planned');
    xlabel('Y (meters)');ylabel('X (meters)');
    hold;
    plot(New_Y, New_X,'g-.');
    plot(New_Y_Way_c(1), New_X_Way_c(1),'gd');
    plot(new2_Y,new2_X,'c.');
    plot(New_Y_Way_c(2), New_X_Way_c(2),'kd');
    plot(Y_SEARCHER, X_SEARCHER, 'm');
    plot([Y_Way_c(1) PrevY_Way_c(1)],[X_Way_c(1) PrevX_Way_c(1)],'r');
    for ik=2:No_tracks,
        plot([Y_Way_c(ik) Y_Way_c(ik-1)],[X_Way_c(ik)...
                X_Way_c(ik-1)],'r');
    end;
    legend('Initial Track - ARIES',...
        'Track After Modem Command - ARIES','Initial Rendezvous Point',...
        'Track After Initial Rendezvous','Cooperative Tracking Waypoint',...
        'SEARCHER Track',...
        'Planned Track - ARIES',4);
    axis([-100 220 -120 240]);
    print -tiff -depsc figure10_wp5
    hold; zoom on
    %***************
    % Plot to look at position error performance of controller during
    % modem download waypoints
    figure(3);clf
    orient tall
    plot(newreal_time([1:iii]),posit_error([1:iii]));grid;hold
    plot(breaktime(1),breakaway(1),'g*');
    title('Distance Between Aries & Searcher After Initial Rendezvous');
    xlabel('Time (seconds)');ylabel('Distance (meters)');
    legend('Aries/Searcher Position Error After Initial Rendezvous',...
        'Breakaway From Cooperative Tracking',4);
    hold; zoom on
end
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Naval Studies Board, National Research Council, *An Assessment of Undersea Weapons Science and Technology*, National Academy of Sciences, 2000.

[2] Department of the Navy, *The Navy Unmanned Undersea Vehicle (UUV) Master Plan*, April 2000.

[3] Autonomous Operations, Future Naval Capabilities Programs (FY02-07), http://www.onr.navy.mil/auto-ops/, Last Accessed November 2002.

[4] Kilfoyle, D.B. and Baggeroer, A.B., "The State of the Art in Underwater Acoustic Telemetry," *IEEE Journal of Oceanic Engineering*, Volume 25, Number 1 , pp. 4-27, January 2000.

[5] Walton, J.M., "Test and Evaluation Experience With An Untethered Supervisory Controlled Undersea Robot," *Oceans '96 MTS/IEEE*, Volume 2, pp. 703-710, September 1996.

[6] Walton, J.M., "Advanced Unmanned Search System," *Oceans '91 Proceedings*, pp.1392-1399, October 1991.

[7] Walton, J.M. and Uhrich, R.W., "114 Untethered UUV Dives: Lessons Learned," *Oceans '95 MTS/IEEE*, Volume 3, pp. 1433-1437, October 1995.

[8] Fletcher, B., "New Roles For UUVS In Intelligence, Surveillance, and Reconnaissance," [www.spawar.navy.mil/robots/pubs/pcmst2000b.pdf], Last Accessed November 2002.

[9] Thorleifson, J.M., and others, "The Theseus Autonomous Underwater Vehicle, A Canadian Success Story," *Oceans '97 MTS/IEEE*, Volume 2, pp.1001-1006, October 1997.

[10] Butler, B. and den Hertog, V., "Theseus, A Cable-Laying AUV," *Oceans '93Proceedings*, Volume 1, pp. 1210-1213, October 1993.

[11] Brancart, P.B. and Madden, J.P., "Autonomous Minehunting and Mapping At-Sea Testing," *Oceans '96 MTS/IEEE*, Volume 2, pp. 800-806, September 1996.

[12] Paglia, J.G. and Wyman, W.F., "DARPA'S Autonomous Minehunting and Mapping Technologies (AMMT) Program: An Overview," *Oceans '96 MTS/IEEE*, Volume 2, pp. 794-799, September 1996.

[13] Freitag, L.E., and others, "Acoustic Communications System for the AMMT Program," *Oceans '96 MTS/IEEE*, -92 Supplement, pp. 87-92, September 1996.

[14] Kristensen, J. and Vestgard, K., "Hugin- An Untethered Underwater Vehicle For Seabed Survey," *Oceans '98 Proceedings*, Volume 1, pp. 118-123, October 1998.

[15] Oliveira, P., and others, "Guidance and Control of the SIRENE Underwater Vehicle: From System Design to Tests at Sea," *Oceans '98 Proceedings*, Volume 2, pp.1043-1048, October 1998.

[16] Smith, S.M., and others, "The Morpheus Ultramodular Autonomous Underwater Vehicle," *IEEE Journal of Oceanic Engineering*, Volume 26, Number 4 , pp. 453-465, October 2001.

[17] Smith, S.M. and Park, J.C., "A Peer-To-Peer Communications Protocol for Underwater Acoustic Communication," *Oceans '97 MTS/IEEE*, Volume 1, pp. 268-272, October 1997.

[18] Chappell, S.G., and others, "Acoustic Communication Between Two Autonomous Underwater Vehicles," *Proceedings of AUV Technology 1994*, pp. 462-469, July 1994.

[19] Von Alt, C. and others, "Hunting for Mines with REMUS: A High Performance, Affordable, Free Swimming Underwater Robot," *Oceans 2001 MTS/IEEE*, Volume 1, pp. 117-122, November 2001.

[20] Freitag, L., and others, "Acoustic Communication in Very Shallow Water: Results from the 1999 AUV Fest," *Oceans 2000 MTS/IEEE*, Volume 3, pp. 2155-2160, September 2000.

[21] Freitag, L., and others, "Integrated Acoustic Communication and Navigation for Multiple UUVs," *Oceans 2001 MTS/IEEE*, Volume 4, pp. 2065-2070, November 2001.

[22] Trimble, G.M., "The Cetus UUV/EOD Robotic Work Package: A Low-Cost Shallow-Water UUV System for Underwater Search and Intervention," *Oceans '98 Proceedings*, Volume 1, pp.369-373, October 1998.

[23] Kim, J.H., and others, "Experiments in Remote Monitoring and Control of Autonomous Underwater Vehicles," *Oceans '96 MTS/IEEE*, Volume 1, pp. 411-416, September 1996.

[24] Rish, J.W., and others, "Operational Testing of the Battlespace Preparation AUV in the Shallow Water Regime," *Oceans 2001 MTS/IEEE*, Volume 1, pp. 123-129, November 2001.

[25] Green, D. and Bernstein, C., "Command, Control, and Data Transport for Underwater Robots Using Acoustic Communications," *Underwater Technology 2002 Proceedings*, pp. 343-348, April 2002.

[26] Urick, R.J., "*Principles of Underwater Sound*," 3rd Edition, pp. 1-16, Peninsula Publishing, 1996.

[27] Sari, H. and Woodward, B., "Underwater Voice Communications Using a Modulated Laser Beam," *Oceans '98 Proceedings*, Volume 2, pp.1183-1188, October 1998.

[28] Von der Weid, J.P., and others, "Underwater Cableless Data Transmission," *Oceans '93 Proceedings*, Volume 3, pp. III191-III193, October 1993.

[29] Benson, R.A. and Curtin, T.B., "Office of Naval Research Program in Underwater Acoustic Communications," www.onr.navy.mil/02/baa01_012/PIP/acomms.doc, Last Accessed November 2002.

[30] Stokey, R., and others, "Enabling Technologies for REMUS Docking: An Integral Component of an Autonomous Ocean-Sampling Network," *IEEE Journal of Oceanic Engineering*, Volume 26, Number 4 , pp. 487-497, October 2001.

[31] Feezor, M.D., and others, "Autonomous Underwater Vehicle Homing/Docking via Electromagnetic Guidance," *IEEE Journal of Oceanic Engineering*, Volume 26, Number 4 , pp. 515-521, October 2001.

[32] Kirkwood, W.J., and others, "Development of a Long Endurance Autonomous Underwater Vehicle for Ocean Science Exploration," *Oceans 2001 MTS/IEEE*, Volume 3, pp. 1504-1512, November 2001.

[33] Bahlavouni, A., and others, "Ice Penetrating Communication Buoy for Underwater Vehicles Operating in the Arctic," *Oceans 2001 MTS/IEEE*, Volume 3, pp. 1500-1503, November 2001.

[34] Webb, D.C., and others, "SLOCUM: An Underwater Glider Propelled by Environmental Energy," *IEEE Journal of Oceanic Engineering*, Volume 26, Number 4 , pp. 447-452, October 2001.

[35] Eriksen, C.C., and others, "Seaglider: A Long-Range Autonomous Underwater Vehicle for Oceanographic Research," *IEEE Journal of Oceanic Engineering*, Volume 26, Number 4 , pp. 424-436, October 2001.

[36] Marco, D.B. and Healey, A.J., "Command, Control, and Navigation Experimental Results With the NPS ARIES AUV," *IEEE Journal of Oceanic Engineering*, Volume 26, Number 4 , pp. 466-476, October 2001.

[37] Clay, C.S. and Medwin, H., *Acoustical Oceanography: Principles and Applications*, John Wiley & Sons, Inc., 1977.

[38] Medwin, H. and Clay, C.S., *Fundamentals of Acoustic Oceanography*, Academic Press, 1998.

[39] Wozencraft, J.M. and Jacobs, I.M., *Principles of Communication Engineering*, John Wiley & Sons, Inc., 1965, Reissued Waveland Press, Inc., 1990.

[40] Proakis, J.G., *Digital Communication*, 4th Edition, McGraw Hill, 2001.

[41] Shannon, C.E. and Weaver, W., *The Mathematical Theory of Communication*, University of Illinois Press, 1949.

[42] Bhatti, S., "Error Correction Coding," www.cs.ucl.ac.uk/staff/S.Bhatti/D51-notes/node33.html, Last Accessed December 2002.

[43] Lin, S. and Costello, D.J., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Inc., 1983.

[44] *Forward Error Correction Encoding and Decoding: Application Note 108*, Intel Corporation, December 1999.

[45] Fleming, C., "A Tutorial on Convolutional Coding with Viterbi Decoding," http://pw1.netcom/~chip.f/viterbi/tutorial.html, Last Accessed October 2002.

[46] Green, D., *D08 Workshop Notes*, Benthos, Inc., September 2002.

[47] Freitag, L., and others, "Analysis of Channel Effects on Direct-Sequence and Frequency-Hopped Spread Spectrum Acoustic Communication," *IEEE Journal of Oceanic Engineering*, Volume 26, Number 4 , pp. 586-593, October 2001.

[48] *Digital Modulation in Communications Systems- An Introduction: Application Note 1298*, Agilent Technologies, 2001.

[49] Sozer, E.M., and others, "Underwater Acoustic Networks," *IEEE Journal of Oceanic Engineering*, Volume 25, Number 1 , pp. 72-83, January 2000.

[50] Proakis, J.G., and others, "Shallow Water Acoustic Networks," *IEEE Communications Magazine*, Volume 39, Number 11, pp. 114-119, November 2001.

[51] Stojanovic, M. and Freitag, L., "Multiuser Undersea Acoustic Communications in the Presence of Multipath Propagation," *Oceans 2001 MTS/IEEE*, Volume 4, pp. 2165-2169, November 2001.

[52] Scussel, K.F., and others, "A New MFSK Acoustic Modem for Operation in Adverse Underwater Channels," *Oceans '97 MTS/IEEE*, Volume 1, pp. 247-254, October 1997.

[53] *Multi-User Frequency-Hopping Underwater Acoustic Communication Protocol Version 1.02*, Woods Hole Oceanographic Institution, May 2000.

[54] *FAU/EdgeTech Underwater Acoustic Modem Interface Communications Reference Guide for Firmware Version 2.3*, Florida Atlantic University, September 2000.

[55] Naval Research Laboratory, "Attenuation of Radio Waves Through Sea Water," http://server5550.itd.nrl.navy.mil/projects/SUBCOMM/atn.html, Last Accessed January 2003.

[56] Vemco Limited, "Signal Propagation," http://www.vemco.com/l_signal.htm, Last Accessed October 2002.

[57] Green, D. and Rice, J., "Channel-Tolerant FH-MFSK Acoustic Signaling," *IEEE Journal of Oceanic Engineering*, Volume 25, Number 1 , pp. 28-39, January 2000.

[58] LeBlanc, L.R., and others, "Chirp FSK Modem for High Reliability Communication in Shallow Water," *OCEANS '99 MTS/IEEE Conference and Exhibition*, Volume 1, pp. 222-227, 1999.

[59] LeBlanc, L.R., and others, "Improved Chirp FSK Modem for High Reliability Communication in Shallow Water," *OCEANS 2000 MTS/IEEE Conference and Exhibition*, Volume 1, pp. 601-603, 2000.

[60] Florida Atlantic University Ocean Engineering Department, *Rev. A Acoustic Modem Schematics*, April 1998.

[61] Beaujean, P.P. and Henderson, E., *FAU GPAM V. 2.0 – Specification Sheet*, 2000.

[62] Marco, D.B., *Procedure to Run Missions with ARIES*, September 2001.

[63] *DS-1 Diver Station Technical Reference Manual, Rev.3*, Desert Star Systems, March 2001.

[64] *SEACAT SBE 19 Conductivity, Temperature, Depth Recorder Operating Manual*, Sea-Bird Electronics, Inc.

[65] Mayo, R.W and Nathman, J., "ForceNet: Turning Information Into Power," *U.S. Naval Institute Proceedings*, Volume 129, Number 2, pp. 42-46, February 2003.

[66] Rice, J., and others, "Evolution of SeaWeb Underwater Acoustic Networking," *OCEANS 2000 MTS/IEEE Conference and Exhibition*, Volume 3, pp. 2007-2017, 2000.

[67] *Acoustic Telemetry Modems Users Manual, Rev. D*, Benthos Incorporated, November 2002.

[68] Kaya, A. and Yauchi, S., "An Acoustic Communication System for Subsea Robot," *Oceans '89 Proceedings*, Volume 3, pp. 765-770, September 1989.

[69] Green, D. and Nguyen, T., "Remote Sensing and Acoustic Telemetry," *OCEANS 2001 MTS/IEEE Conference and Exhibition*, Volume 1, pp. 264-268, November 2001.

[70] Green, D. and Nguyen, T., "Remote Sensing and Acoustic Telemetry," *Sea Technology*, Volume 43, Number 5, pp. 19-22, May 2002.

[71] Green, D., "Transport of Underwater Imagery Using Acoustic Communication," *Sea Technology*, Volume 44, Number 1, pp. 46-50, January 2003.

[72] Keegan, J.J., *Trajectory Planning for the ARIES AUV*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 2002.

[73] Healey, A.J. and Lienard, D., "Multivariable Sliding Mode Control for Autonomous Diving and Steering of Unmanned Underwater Vehicles," *IEEE Journal of Oceanic Engineering*, Volume 18, Number 3, pp. 327-339, July 1993.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, VA

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, CA

3. Professor Anthony J. Healey, Code ME/Hy
   Department of Mechanical Engineering
   Naval Postgraduate School
   Monterey, CA

4. Professor Morris Driels, Code ME/Dr
   Department of Mechanical Engineering
   Naval Postgraduate School
   Monterey, CA

5. Associate Professor Fotis Papoulias, Code ME/Pa
   Department of Mechanical Engineering
   Naval Postgraduate School
   Monterey, CA

6. Associate Professor Roberto Cristi, Code EC/Cx
   Department of Electrical Engineering
   Naval Postgraduate School
   Monterey, CA

7. Associate Professor Josh Gordis, Code ME/Go
   Department of Mechanical Engineering
   Naval Postgraduate School
   Monterey, CA

8. CDR William J. Marr, USN
   Naval Architecture & Ocean Engineering Department
   U. S. Naval Academy
   Annapolis, MD

9. Mr. Arnold R. Marr
   Wayne, NE

10. LTCOL Douglas C. Marr, USMC
6th Marine Corps Recruiting District
Parris Island, SC

11. Mr. Joe Rice
Engineering Acoustics Chair
Naval Postgraduate School
Monterey, CA