NAVAL POSTGRADUATE SCHOOL
Monterey, California



THESIS

SIMULATION OF AN ORTHOGONAL FREQUENCY DIVISION MULTIPLEXING BASED UNDERWATER COMMUNICATION SYSTEM USING A PHYSICS BASED MODEL FOR THE UNDERWATER ACOUSTIC SOUND CHANNEL

by

Gell Tiger Lee Pittman III

September 2001

Thesis Advisor: Roberto Cristi
Co-Advisor: Kevin B. Smith

Approved for public release; distribution is unlimited

Report Documentation Page

Report Date	Report Type	Dates Covered (from... to)
30 Sep 2001	N/A	-

Title and Subtitle Simulation of an Orthogonal Frequency Division Multiplexing Based Underwater Communication System Using a Physics Based Model for the Underwater Acoustic Sound Channel	**Contract Number**
	Grant Number
	Program Element Number
Author(s) Pittman III, Gell Tiger L.	**Project Number**
	Task Number
	Work Unit Number
Performing Organization Name(s) and Address(es) Research Office Naval Postgraduate School Monterey Ca. 93943-5138	**Performing Organization Report Number**
Sponsoring/Monitoring Agency Name(s) and Address(es)	**Sponsor/Monitor's Acronym(s)**
	Sponsor/Monitor's Report Number(s)
Distribution/Availability Statement Approved for public release, distribution unlimited	
Supplementary Notes	
Abstract	
Subject Terms	
Report Classification unclassified	**Classification of this page** unclassified
Classification of Abstract unclassified	**Limitation of Abstract** UU
Number of Pages 209	

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** September 2001	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Title (Mix case letters) Simulation of an Orthogonal Frequency Division Multiplexing Based Underwater Communication System Using a Physics Based Model for the Underwater Acoustic Sound Channel			**5. FUNDING NUMBERS**
6. AUTHOR(S) **Pittman III, Gell Tiger L.**			
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)***

The primary thrust of this thesis is the development of a computer-based simulation of an Orthogonal Frequency Division Multipexing (OFDM) based underwater acoustic communication system. The product will support the testing and evaluation of various digital signal processing algorithms applicable to underwater acoustic communication system using OFDM as well as the study of the effects of acoustic channel and communication system factors of the key parameters of the system such as bit error rate, received signal to noise ratio, frequency band of employment and overall system bit rate. The underwater acoustic sound channel is modeled using a physics based parabolic equation approximation. The simulation models the key components in the transmitter and receiver that contribute to the overall performance of the system. The results of the thesis provide expected values for system performance in terms of bit rate, bit error rate and received SNR for given frequency bands and are validated through comparison to theoretically derived expectations and to ocean testing of OFDM underwater communication systems.

14. SUBJECT TERMS orthogonal frequency division multiplexing, underwater acoustics, acoustic communication, underwater communication, OFDM, MMPE, parabolic equation			**15. NUMBER OF PAGES**
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** 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

SIMULATION OF AN ORTHOGONAL FREQUENCY DIVISION MULTIPLEXING BASED UNDERWATER COMMUNICATION SYSTEM USING A PHYSICS BASED MODEL FOR THE UNDERWATER ACOUSTIC SOUND CHANNEL

Gell Tiger L. Pittman III
Lieutenant, United States Navy
B.S., South Dakota School of Mines and Technology, 1994

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ENGINEERING SCIENCE
(MAJOR IN ELECTRICAL ENGINEERING)

from the

NAVAL POSTGRADUATE SCHOOL
September 2001

Author: _____
Gell Tiger L. Pittman III

Approved by: _____
Roberto Cristi, Thesis Advisor

_____
Kevin B. Smith, Co-Advisor

_____
Jeffrey B. Knorr, Chairman
Department of Electrical and Computer Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The primary thrust of this thesis is the development of a computer-based simulation of an Orthogonal Frequency Division Multipexing (OFDM) based underwater acoustic communication system. The product will support the testing and evaluation of various digital signal processing algorithms applicable to underwater acoustic communication systems using OFDM as well as the study of the effects of the acoustic channel and communication system factors on the key parameters of the system such as bit error rate, received signal to noise ratio, frequency band of employment and overall system bit rate. The underwater acoustic sound channel is modeled using a physics based parabolic equation approximation. The simulation models the key components in the transmitter and receiver that contribute to the overall performance of the system. The results of the thesis provide expected values for system performance in terms of bit rate, bit error rate and received SNR for given frequency bands and are validated through comparison to theoretically derived expectations and to ocean testing of OFDM underwater communication systems.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1
A. PURPOSE....1
B. GOALS....1
C. METHODOLOGY....2
D. BENEFITS....2

II. OFDM THEORY....5
A. OFDM BASICS....5
1. Subcarrier Orthogonality....8
2. Guard Time and Cyclic Extension....11
B. CHANNEL ANALYSIS....15

III. OFDM SIMULATION MODEL....21
A. METHODOLOGY....21
B. TRANSMITTER....23
1. OFDM Parameter Determination....23
a. Example Calculation....26
2. Signal Generation....28
3. Quantization....29
4. Decimal to Binary Conversion....30
5. QAM Buffer....30
6. Reed Solomon Encoder....30
7. Block Interleaver....32
8. QAM Modulator....34
9. IFFT/OFDM Modulator....41
10. Cyclic Extension....42
11. Parallel to Serial Converter....43
12. Double Side Band Modulator....43
C. MMPE CHANNEL MODEL....49
D. RECEIVER....55
1. AWGN Addition....55
2. DSB Demodulator....56
3. Synchronizer....59
4. Composite Description of Serial to Parallel Conversion through FFT/OFDM Demodulation....64
5. Acoustic Channel Application....65
6. Channel Equalization....65
7. QAM Demodulator....67
8. Composite Description of Block De-Interleaving through Error Analysis....70

IV. RESULTS....73
A. SYSTEM THEORETICAL PERFORMANCE....73

1. Single Carrier Analysis....73
2. OFDM (Multi-Carrier) Analysis....76
B. SIMULATION RESULTS....78
V. CONCLUSIONS....89
APPENDIX A. SIMULATION RESULTS....93
APPENDIX B. MMPE RESULTS....119
APPENDIX C. MATLAB CODE....127
A. TRANSMITTER CODE....127
1. ofdm_sim_xmitter.m....127
2. parameters.m....132
3. signal_generator.m....135
4. quantization.m....136
5. dec_2_bin.m....136
6. qam_buffer.m....136
7. reedsolomon.m....137
8. qam_modulator.m....137
9. inverse_fft.m....140
10. cyc_extension.m....141
11. parlel_2_serial.m....141
12. dsb_modulator.m....141
B. RECEIVER CODE....143
1. ofdm_sim_receiver.m....143
2. rcvr_dsb_demodulator.m....147
3. synchronizer.m....149
a. peakfinder.m....150
4. mmpe_channel.m....152
5. equalization.m....154
6. qam_demodulator.m....155
a. qaskdecomod.m....157
7. deinterleaver.m....159
8. rsdecoder.m....160
9. error_check.m....160
C. PERFORMANCE ANALYSIS CODE....160
1. ofdm_sim_control_func.m....160
2. ofdm_performance.m....161
3. perform_plotter.m....168
LIST OF REFERENCES....185
INITIAL DISTRIBUTION LIST....187

LIST OF FIGURES

Figure 2.1. OFDM Basic Block Diagram....5
Figure 2.2. Subcarrier and OFDM Symbol Arrangement. 7
Figure 2.3. Orthogonality of OFDM Subcarriers. 8
Figure 2.4. Subcarrier Orthogonality via Subcarrier Spectra [after Ref. 4]. 9
Figure 2.5. Multipath Effect with zero signal in the Guard Time [after Ref. 4]. 12
Figure 2.6. OFDM symbol and subcarriers with cyclic extension in the guard interval [after Ref. 4]. 13
Figure 2.7. OFDM Signal with cyclic prefix, 3 subcarriers, 2 ray multipath environment [after Ref. 4]....14
Figure 2.8. Basic Communication System. 15
Figure 2.9. Subchannel Decomposition of Channel Frequency Response....19
Figure 3.1. OFDM Simulation Block Diagram. 22
Figure 3.2. Measured Underwater Acoustic Channel Impulse Response [from Ref. 1]. 25
Figure 3.3. Information Signal. 28
Figure 3.4. Quantization Illustration. 29
Figure 3.5. Reed Solomon Encoder and Block Interleaver. 34
Figure 3.6. 16 QAM Constellation with In-phase and Quadrature Mapping. 35
Figure 3.7. QAM Modulator. 38
Figure 3.8. QAM Modulator with entire Complex OFDM Symbol Structure. 40
Figure 3.9. Interpolated Baseband OFDM Signal. 44
Figure 3.10. General FIR Filter (Hamming Window Design). 46
Figure 3.11. Interpolated Baseband OFDM Signal following FIR Filter. 47
Figure 3.12. FIR Filter Frequency Response. 48
Figure 3.13. Modulation Signals. 49
Figure 3.14. Sound Speed Profile....55
Figure 3.15. DSB Demodulator Signal Processing Effects. 59
Figure 3.16. Synchronization Flowchart [after Ref. 4]. 61
Figure 3.17. OFDM Signal Structure with Synchronizer Observation Intervals. 62
Figure 3.18. Correlation Signal Realization, SNR = 17.413 dB, N=2048, R = 5 kbps. 62
Figure 3.19. Acoustic Channel Filter Plots for Realization of Fig. 3.17. 63
Figure 3.20. Channel Estimates, N=2048, 16 QAM, 5 kbps, f_c = 6 kHz, $\overline{N}_{act} = 750$ 67
Figure 3.21. QAM Demodulator. 68
Figure 3.22. Received Symbol QAM Decoding Process. 70
Figure 4.1. Channel Bit per Subcarrier Capacity. 77
Figure 4.2. Bandwidth vs Bit Rate for Q-ary QAM OFDM Signals....78
Figure 4.3. Transmission Loss, Bottom Roughness 4 m, Water Depth 100 m. 80
Figure 4.4. Acoustic Energy Arrival, Bottom Roughness 4 m, Water Depth 100 m. 81
Figure 4.5. 8 QAM Composite Results. 82
Figure 4.6. 16 QAM Composite Results. 82
Figure 4.7. 32 QAM Composite Results. 83
Figure 4.8. 64 QAM Composite Results. 83

Figure 4.9. 16 QAM Signal Realization, f_c = 6 kHz, R = 5 kbps, SNR = 5.3 dB.....84
BER = 1.6 E-2.....84
Figure 4.10. 16 QAM Signal Realization, f_c = 6 kHz, R = 5 kbps, SNR = 5.3 dB.....85
Figure 4.11. 16 QAM Signal Realization, f_c = 6 kHz, R = 5 kbps, SNR = 7.4 dB.....86
BER = 2.5 E-4.....86
Figure 4.12. 16 QAM Signal Realization, f_c = 6 kHz, R = 5 kbps, SNR = 7.4 dB.....87
Figure A.1. BER vs SNR for 8 QAM, fc = 8 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m. 93
Figure A.2. BER vs SNR for 8 QAM, fc = 8 kHz, R = 5 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m. 94
Figure A.3. BER vs SNR for 8 QAM, fc = 10 kHz, R = 6 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m. 94
Figure A.4. BER vs SNR for 8 QAM, fc = 10 kHz, R = 6 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m. 95
Figure A.5. BER vs SNR for 8 QAM, fc = 8 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m. 95
Figure A.6. BER vs SNR for 8 QAM, fc = 12 kHz, R = 7.5 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.....96
Figure A.7. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m. 96
Figure A.8. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 2m, Water Depth = 100m. 97
Figure A..9 BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 4m, Water Depth = 100m. 97
Figure A.10. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m. 98
Figure A.11. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 4m, Water Depth = 100m. 98
Figure A.12. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 340m. 99
Figure A.13. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 2m, Water Depth = 100m. 99
Figure A.14. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 2m, Water Depth = 340m. 100
Figure A.15. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 0m, Water Depth = 340m. 100
Figure A.16. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 2m, Water Depth = 340m. 101
Figure A.17. BER vs SNR for 16 QAM, fc = 12 kHz, R = 10 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.....101
Figure A.18. BER vs SNR for 16 QAM, fc = 12 kHz, R = 10 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.....102
Figure A.19. BER vs SNR for 16 QAM, fc = 8 kHz, R = 6.67 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.....102

Figure A.20. BER vs SNR for 16 QAM, fc = 8 kHz, R = 6.67 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.....103
Figure A.21. BER vs SNR for 16 QAM, fc = 10 kHz, R = 8 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.....103
Figure A.22. BER vs SNR for 16 QAM, fc = 10 kHz, R = 8 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.....104
Figure A.23. BER vs SNR for 32 QAM, fc = 10 kHz, R = 10 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.....104
Figure A.24. BER vs SNR for 32 QAM, fc = 10 kHz, R = 10 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.....105
Figure A.25. BER vs SNR for 32 QAM, fc = 8 kHz, R = 8.33 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.....105
Figure A.26. BER vs SNR for 32 QAM, fc = 8 kHz, R = 8.33 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.....106
Figure A.27. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m. 106
Figure A.28. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 2m, Water Depth = 100m. 107
Figure A.29. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 4m, Water Depth = 100m. 107
Figure A.30. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m. 108
Figure A.31. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 2m, Water Depth = 100m. 108
Figure A.32. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 4m, Water Depth = 100m. 109
Figure A.33. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 340m. 109
Figure A.34. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 2m, Water Depth = 340m. 110
Figure A.35. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 0m, Water Depth = 340m. 110
Figure A.36. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 2m, Water Depth = 340m. 111
Figure A.37. BER vs SNR for 64 QAM, fc = 8 kHz, R = 10 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.....111
Figure A.38. BER vs SNR for 64 QAM, fc = 8 kHz, R = 10 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.....112
Figure A.39. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m. 112
Figure A.40. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m. 113
Figure A.41. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 4m, Water Depth = 100m. 113

Figure A.42. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m. 114
Figure A.43. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 2m, Water Depth = 100m. 114
Figure A.44. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 4m, Water Depth = 100m. 115
Figure A.45. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m. 115
Figure A.46. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 2m, Water Depth = 100m. 116
Figure A.47. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m. 116
Figure A.48. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 2m, Water Depth = 100m. 117
Figure B.1 Transmission Loss, Roughness 0 m, Range 2 km 119
Figure B.2 Acoustic Paths, Roughness 0 m, Range 2 km, f_c 7 kHz....119
Figure B.3 Transmission Loss, Roughness 2 m, Range 2 km 120
Figure B.4 Acoustic Paths, Roughness 2 m, Range 2 km, f_c 7 kHz....120
Figure B.5 Transmission Loss, Roughness 4 m, Range 2 km 121
Figure B.6 Acoustic Paths, Roughness 4 m, Range 2 km, f_c 7 kHz....121
Figure B.7 Transmission Loss, Roughness 0 m, Range 2 km 122
Figure B.8 Acoustic Paths, Roughness 0 m, Range 2 km, f_c 7 kHz....122
Figure B.9 Transmission Loss, Roughness 2 m, Range 2 km 123
Figure B.10 Acoustic Paths, Roughness 0 m, Range 2 km, f_c 7 kHz....123
Figure B.11 Transmission Loss, Roughness 0 m, Range 2 km 124
Figure B.12 Acoustic Paths, Roughness 0 m, Range 2 km, f_c 10 kHz....124
Figure B.13 Transmission Loss, Roughness 0 m, Range 2 km 125
Figure B.14 Acoustic Paths, Roughness 0 m, Range 2 km, f_c 12 kHz....125
Figure B.15 Transmission Loss, Roughness 0 m, Range 2 km 126
Figure B.16 Acoustic Paths, Roughness 0 m, Range 2 km, f_c 14 kHz....126

LIST OF TABLES

Table 3.1. Typical Delay Spread Values.....24
Table 5.1. Experimental Multicarrier Results [from Ref. 1].....89

THIS PAGE INTENTIONALLY LEFT BLANK

ACRONYMS, ABBREVIATIONS & SYMBOLS

ADC	Analog to Digital Converter
AWGN	Additive White Gaussian Noise
b	Number of Bits in an OFDM Symbol
b_{sc}	Bits per QAM Symbol
BER	Bit Error Rate
d	Distance Between Points in a QAM Constellation
d_{min}	Minimum Distance Between Received QAM Constellation Points at Channel Output
d(n)	Demodulating Signal
DAC	Digital to Analog Converter
DSBSC	Double Side Band Suppressed Carrier
ε	Two Dimensional Square QAM Symbol Energy
fc	DSB Modulator Carrier Frequency
FIR	Finite Impulse Response
$F_{s,FFT}$	Sampling Rate in the FFT Interval
$F_{s.\,\mathrm{mod}}$	Interpolated Sampling Rate
FFT	Fast Fourier Transform
Δf_{sc}	Subcarrier Spacing
γ_c	Coding Gain

γ_m	Margin
h(n)	Filter Impulse Response
I	Interpolation Constant
ICI	Inter Symbol Interference
IFFT	Inverse Fast Fourier Transform
ISI	Inter Symbol Interference
k	Reed Solomon Input Symbol Word Length
Γ	SNR Gap
LPF	Low Pass Filter
MMPE	Monterey-Miami Parabolic Equation
n	Reed Solomon Code Word Length
N_h	Hamming Window FIR Filter Order
N_0	Number of Zero Symbols in QAM Modulator
$\overline{N}$	Number of Subcarriers
$\overline{N}_{act}$	Number of Active Subcarriers
N_{block}	Number of OFDM Symbol Blocks
$N_{QAM,I}$	Number of Complex QAM Symbols for Transmission
$\overline{N}_{zer}$	Number of Zero Subcarriers
OFDM	Orthogonal Frequency Division Multiplexing
PE	Parabolic Equation
q	Number of Bits per QAM Symbol
Q	QAM Constellation Size

QAM	Quadriture Amplitude Modulation
r	Coding Rate
R	Overall Bit Rate
R_s	OFDM Symbol Rate
SNR	Signal to Noise Ratio
SSP	Sound Speed Profile
t_{ds}	Channel Delay Spread
t_g	Guard Time
T	OFDM Symbol Period
T_{FFT}	FFT Interval Duration
ω	Digital Frequency
ω_p	Pass Band Digital Frequency
ω_s	Stop Band Digital Frequency
W	OFDM Signal Bandwidth
$x_r(n)$	Received Signal with AWGN
$x_{r,d}(n)$	Demodulated $x_r(n)$
$x_{r,f}(n)$	xr,d(n) after Anti-aliasing LPF
$x_{r,fd}(n)$	$x_{r,f}(n)$ Decimated by D
$\underline{x}_N$	Matrix of Real Time Domain Samples
$\underline{x}_{N,c}$	Matrix of Real Time Domain Samples (w/ cyclic extension)
X_h	Complex QAM Data Block
$X_{p,1}$	Complex QAM Pilot Block

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGEMENTS

First and foremost I am grateful to my beautiful wife and wonderful children for the support they provided and the sacrifices they made in support of my completing this work.

Secondly I thank my thesis advisors, Dr. Roberto Cristi and Dr. Kevin Smith, for their guidance and patience.

Finally in appreciation of CDR Jim Hill, Code 35 Curriculum Officer, and Eva Anderson, Code 35 Educational Specialist, for their assistance throughout my tour at the Naval Postgraduate School.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Communication in the underwater battle space is essential to Netcentric Warfare concepts in the 21st century. Acoustic communication holds promising capabilities for connecting the undersea environment. The underwater acoustic sound channel however, presents challenging obstacles that must be overcome to allow satisfactory acoustic communication. Possibly the most significant are the frequency selective fading and multipath natures the underwater channel presents.

Orthogonal Frequency Division Multiplexing (OFDM) is extremely robust to multipath frequency selective fading channels and therefore is rapidly developing interest in application to underwater acoustic communication. The primary thrust of this work is to develop a computer-based simulation of an OFDM based communication system for the undersea environment. The result will then provide a foundation upon which the key parameters and capabilities of underwater acoustic communication using OFDM can be investigated. The immediate benefit of the work is the establishment of the achievable communication system parameters, common to all communication systems, which are bit rate, bit error rate, bandwidth and the required signal to noise ratios.

The simulation is performed in the Matlab environment where all major hardware in the transmitter and receiver are simulated. The underwater acoustic sound channel is modeled using the physics based parabolic equation approximation. The model is the Monterey-Miami Parabolic Equation (MMPE) which employs the Split Step Fourier method. Use of a physics based model vice the typical stochastic model for the acoustic channel ensures a better realization of the phenomenon that is experienced in the undersea environment.

The simulation allows for any type of data to be transmitted from analog voice signals to binary data. Significant flexibility is coded into the simulation to allow for a wide range of parameter analysis and sensitivity testing. Reed Solomon coding and block interleaving are applied to achieve several decibels of forward error correction coding gain. Quadrature amplitude modulation (QAM) is applied with constellation sizes of 8 to 64 points. Double sideband suppressed carrier modulation is used to modulate the baseband signal to the desired transmission band.

Bit rates achieved range from 4 to 10 kilobits per second (kbps) utilizing from 2 to 6 kilohertz (kHz) of bandwidth depending on the size of the QAM constellation applied. The transmitted signals occupy frequency bands between 4 and 14 kHz.

The results are validated through a theoretical development of achievable performance. In addition validation is made through comparison to in water testing of OFDM.

The results substantiate the feasibility of OFDM as an attractive candidate for underwater acoustic communication. The product of the work provides validated system performance expectations as well as a foundation upon which future work in this area can be performed.

I. INTRODUCTION

A. PURPOSE

Netcentric Warfare has moved to the forefront of the Navy's priorities for the 21st century. Essential to the concept of Netcentric Warfare is the ability to communicate quickly, reliably, efficiently and in many cases covertly in all regions of the battle space. More specifically, key characteristics of the communication capability required are high data rates, low error rates, environmental robustness and minimally overt acoustic signatures. The undersea environment is arguably the most challenging region of the battle space in which to communicate effectively. Current and future goals for underwater wireless communication involve data flow between submarines, surface units, unmanned undersea vehicles (UUVs) and ocean surface and ocean bottom communication buoys in any combination via the underwater sound channel. Research and development in the fundamental communication methods and protocols that support Netcentric Warfare concepts will enhance the Navy's ability to fight successfully in all areas of the battle space.

Orthogonal Frequency Division Multiplexing (OFDM) possesses characteristics that have the potential to overcome the adverse effects of frequency selective multipath fading present in the underwater sound channel. Establishing groundwork in this area to address the feasibility of OFDM for underwater communication is essential to developing an underwater communication system that is beneficial to the Navy.

B. GOALS

The primary goal is to develop a computer simulation that will allow for research into the key parameters, capabilities, and limitations of OFDM in underwater acoustic communication. The simulation provides a 'test bed' for future study of other modulation methods, forward error correction coding, quantization, signal detection, decoding etc. In

developing and validating the simulation the feasibility of OFDM systems in underwater communication is verified. There is one known experiment with OFDM in the underwater acoustic sound channel that is used as a standard against which the results of the simulation are validated. The work was done by Coatelan and Glavieux. [Ref. 1]

While there has been considerable work using OFDM for communication all but the work of Coatelan and Glavieux [Ref. 1] and that of Kim and Lu [Ref. 2] has been outside the area of underwater acoustic communication. Kim and Lu [Ref. 2] present a new broadband underwater channel model that has the capability to model the Doppler time scaling effect as well as multi-path effects. The propagation path delay is modeled with two components that generate the simulated Doppler shift determined by the relative motion between the source and receiver. The simulation work by Kim and Lu [Ref. 2] represents the only previous work in simulating an underwater acoustic communication system employing OFDM as the backbone. A physics based model is used to provide a realistic model of underwater acoustic sound channel.

C. METHODOLOGY

The simulation is run on a PC using Matlab code. The underwater sound channel is modeled using a physics based, parabolic equation (PE) model, namely the Monterey Miami Parabolic Equation model.

The simulation is used to conduct experimentation to examine key parameters such as achievable bit rate and bit error rate, required SNR, transmission range, robustness to multipath propagation and frequency selective fading, and bandwidth requirements. The strengths and weaknesses of OFDM applied to underwater acoustic communications will be identified to assess its feasibility and merit for future work.

D. BENEFITS

Simulation is rapidly becoming a popular method for experimentation with technology-based systems. One of the biggest advantages of simulation is the ability to perform countless numbers of experiments under controlled conditions at a minimal cost.

Some of the biggest obstacles to simulation are the decision of what level of simulation is required to accurately capture the real world and validation of the simulation. With all simulations there must be an initial model. This model suffers from some simplifications that cause departure from reality. Later improvements will help the model to converge on the characteristics of the real world system that is being modeled. With that in mind many design decisions have been made in developing the simulation. The emphasis is on modeling the aspects of the system which have the greatest impact on the key parameters of interest.

Finally, the benefit of this thesis is the ability to experiment with OFDM based communication systems such that the factors influencing the key parameters in communication can be understood. Understanding these factors will provide the knowledge necessary to build actual systems for underwater acoustic communication.

THIS PAGE INTENTIONALLY LEFT BLANK

II. OFDM THEORY

A. OFDM BASICS

Modulating an orthogonal set of subcarriers with Q-ary complex QAM symbols generates an OFDM signal. The basic OFDM system is shown in Fig. 2.1 below. The input bit stream is buffered and converted to parallel complex QAM symbols in the QAM Modulator.

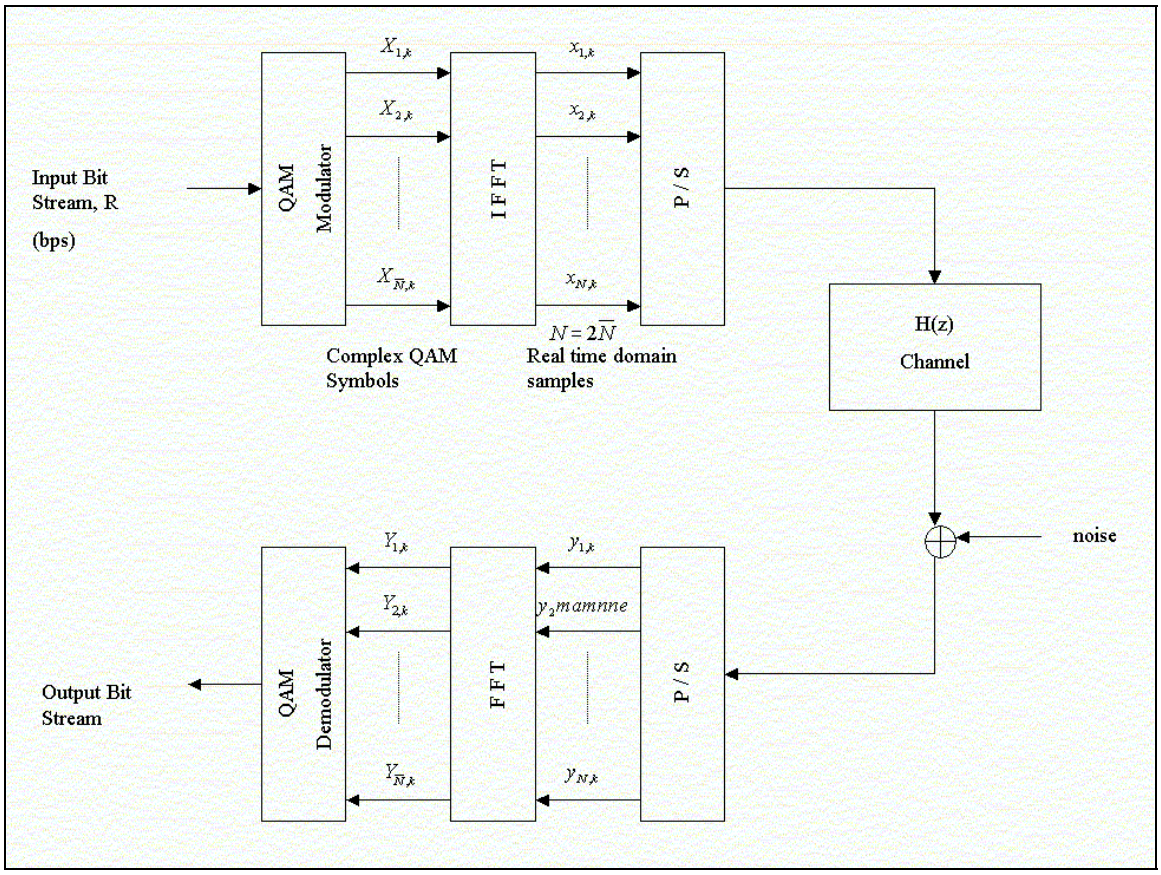


Figure 2.1. OFDM Basic Block Diagram.

The i^{th} subchannel, denoted as $X_{i,k}$, where i is the subchannel index and k is the OFDM symbol index, can be independently modulated as in adaptive modulation

schemes or all subcarriers may be modulated in the same manner. The parallel QAM symbols are then applied to the IFFT, which generates N parallel real samples. The real samples are then converted to serial format. In this simplified block diagram the receiver reverses the operations performed by the transmitter.

The QAM modulator buffers the input bits into blocks of size b, where $b = RT$, T is the symbol duration in seconds and R is the overall bit rate in bits per second. The symbol rate then is $R_s = 1/T$ symbols per second. The number of bits per subchannel determines the size of the QAM constellation, Q. For example with $q = 4$ bits per subchannel, $Q = 2^4$, which is 16 QAM. In OFDM the frequency spectrum of the signal is created in the QAM Modulator and then the time domain signal results from passing the output through the IFFT operation. The complex QAM symbols derived from the binary code words are sequentially placed in a subcarrier band and OFDM symbol. Figure 2.2 illustrates the organization of four OFDM blocks with eight subcarriers.

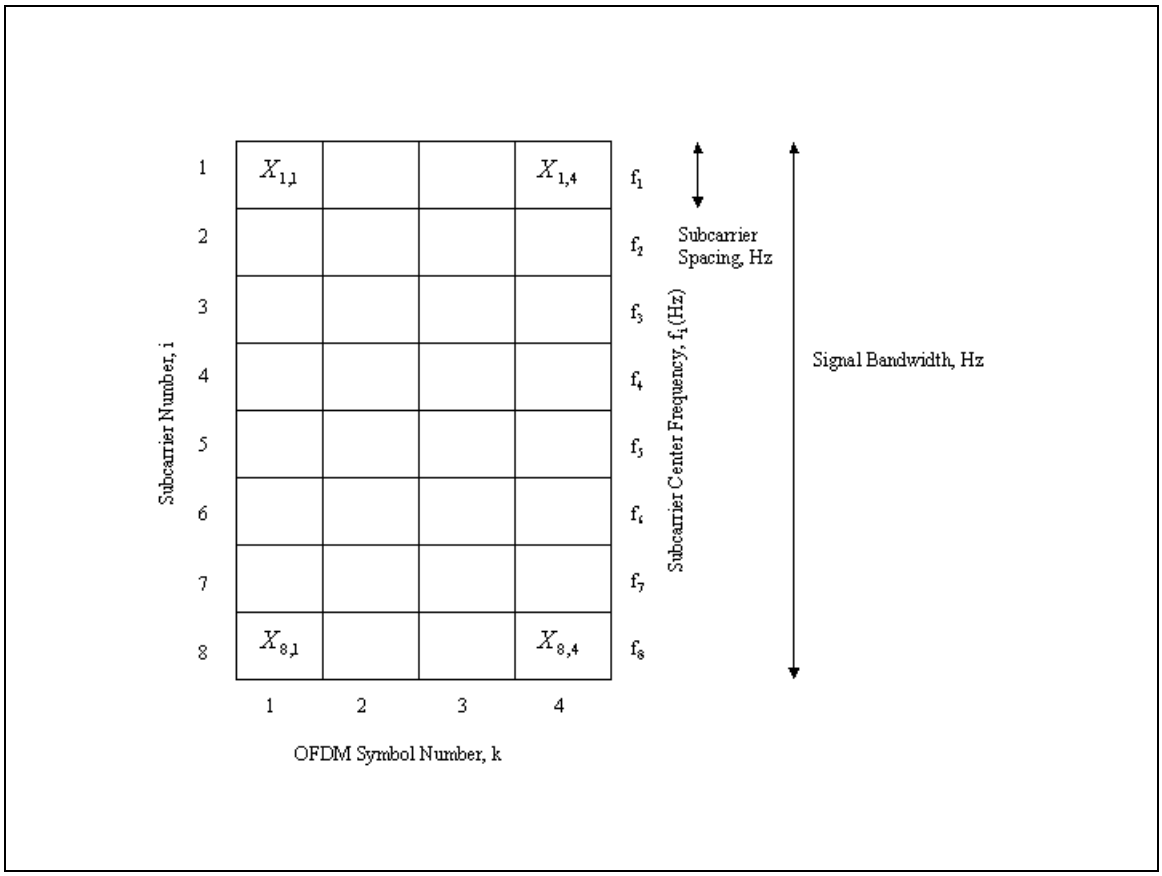


Figure 2.2. Subcarrier and OFDM Symbol Arrangement.

The IFFT provides an orthogonal transformation and maintains the energy contained in the input symbol $X_{i,k}$ [Ref. 3] Written as an equation, this is

$$\frac{1}{N}\sum_{i=1}^{N}\left|X_{i,k}\right|^{2}=\sum_{n=1}^{N}x_{n,k}^{2}. \tag{2.1}$$

Due to the orthogonality of the IFFT the subcarriers each have an integer number of cycles within the FFT interval. The number of cycles in the FFT interval for a given subcarrier differs by one from that of adjacent subcarriers. Note that the terms subchannel and subcarrier are used interchangeably in OFDM terminology. Figure 2.3 illustrates the orthogonality of 3 subcarriers in an OFDM symbol. In this example each of the

subcarriers has the same magnitude and phase but this is not in general the case. As can be seen, each subcarrier has an integer number of cycles within the FFT interval.

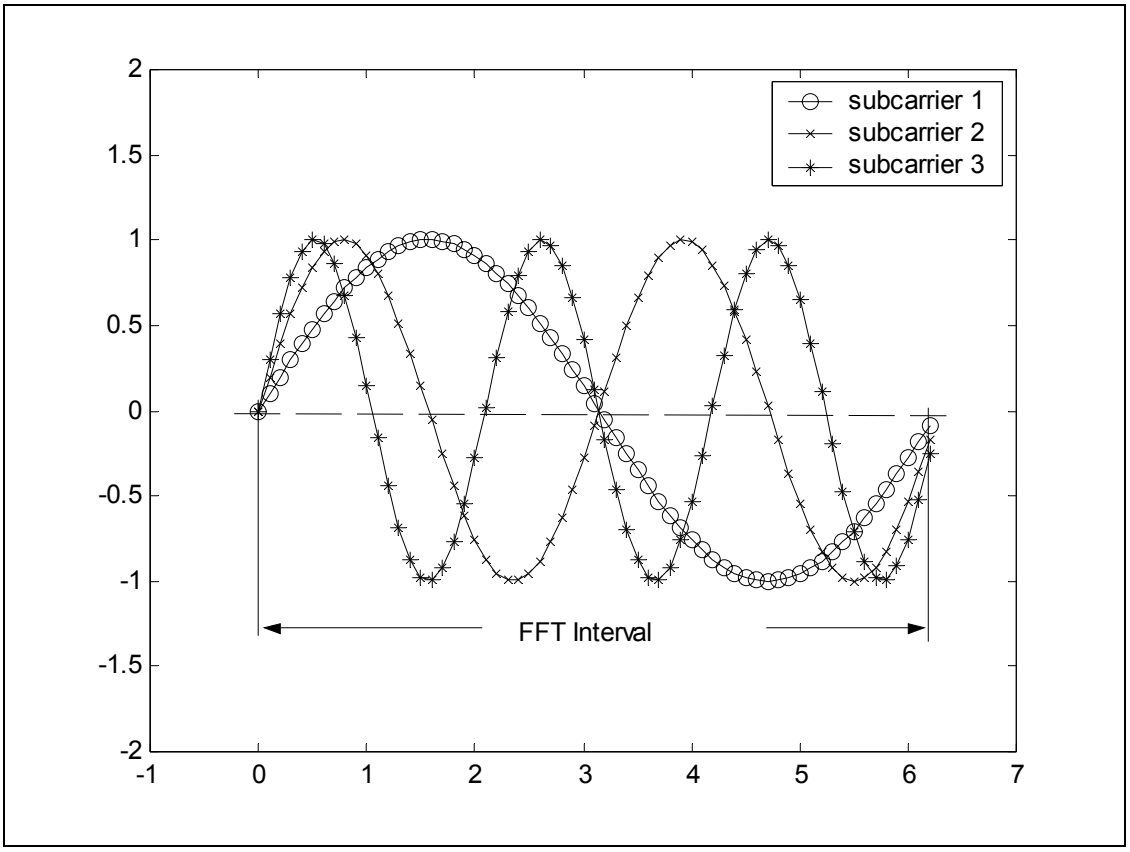


Figure 2.3. Orthogonality of OFDM Subcarriers.

1. Subcarrier Orthogonality

The real bandpass OFDM symbol can be expressed as follows:

$$x(t) = \mathrm{Re}\{\sum_{i=-\frac{\overline{N}}{2}}^{\frac{\overline{N}}{2}-1} d_{i+\overline{N}/2} \exp(j2\pi(f_c - \frac{i+1/2}{T})(t-t_0))\},\ t_0 \le t \le t_0 + T$$

$$x(t) = 0, t < t_{0,}\ t > t_0 + T \qquad (2.2)$$

where T is the symbol period, f_c is the carrier frequency, $\overline{N}$ is the number of subcarriers, d_i are the complex QAM symbols and t_0 is the symbol starting time [Ref. 4]. From Eq.

2.2 it can be seen that the OFDM symbol consists of non-zero subcarriers over the symbol period T. As a result the frequency spectrum of the OFDM symbol consists of the convolution of dirac delta functions at frequencies corresponding to the individual subcarrier frequencies with the frequency spectrum of a unit amplitude square wave with duration T. The frequency spectrum of a unit amplitude square wave is sinc(πfT), which is zero at all frequencies that are a multiple of $1/T$. As a result the spectrum of an OFDM symbol with overlapping sinc functions is as shown in Figure 2.4. Note that at the maximum of each sinc, i.e. frequency spectrum of each subcarrier, all other sinc functions are zero. This is the point where the OFDM receiver calculates the spectral values for the individual subcarriers. This means that the OFDM receiver is able to demodulate each subcarrier free of interference from other carriers. Inter Carrier Interference (ICI) would exist and degrade the received signal if this were not the case. Note that this analysis is done in the frequency domain of the OFDM signal and therefore it is ICI rather than Inter Symbol Interference (ISI) that is avoided. [Ref. 4]

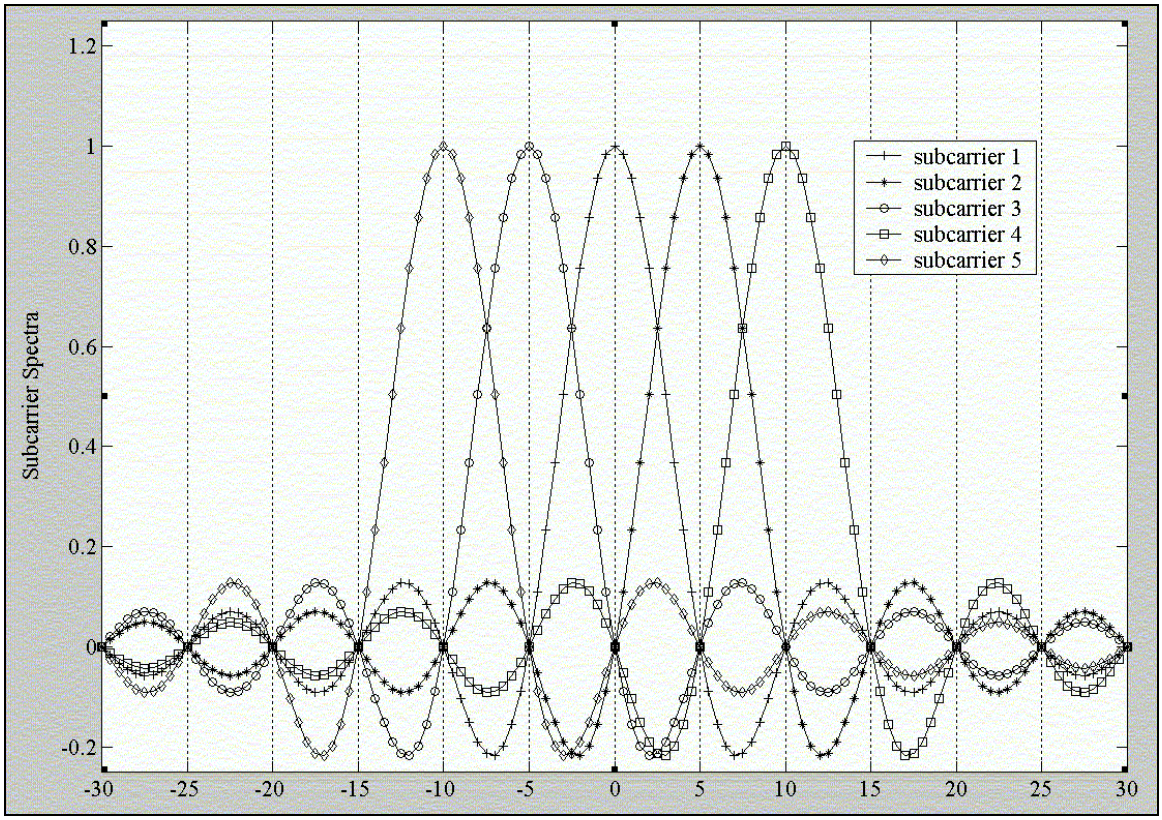


Figure 2.4. Subcarrier Orthogonality via Subcarrier Spectra [after Ref. 4].

The real bandpass signal of Eq. 2.2 can also be expressed in complex baseband form as

$$\boxed{\begin{aligned} x(t) &= \sum_{i=-\frac{\overline{N}}{2}}^{\frac{\overline{N}}{2}-1} d_{i+\overline{N}/2} \exp\left(j2\pi \frac{i}{T}(t-t_0) \right), \quad for\ \ t_0 \le t \le t_0 + T \\ x(t) &= 0, \quad for\ \ t < t_0, t > t_0 + T \end{aligned}} \qquad (2.3)$$

The complex baseband signal is the inverse Fourier transform of $\overline{N}$ QAM symbols. Specifically each OFDM symbol is the inverse Fourier transform of the columns or blocks of Fig. 2.2. In practice the Inverse Fast Fourier Transform (IFFT) is applied to the QAM symbols to generate the discrete time OFDM signal written in equation form as

$$\boxed{x(n) = \sum_{i=0}^{\overline{N}-1} d_i \exp\left(j2\pi \frac{in}{N} \right)} \qquad (2.4)$$

where N is the size of the IFFT and is equal to 2* $\overline{N}$.[Ref. 2]

Since the output of the IFFT must produce a real signal, the OFDM blocks of Fig. 2.2 must be modified prior to application of the IFFT. From Fourier transform theory it is known that a complex frequency domain signal that is conjugate symmetric will produce a real time domain signal upon application of the inverse Fourier transform. Therefore since the $\overline{N}$ QAM symbols in each OFDM block represent the information to be transmitted, and are defined in the frequency domain, the blocks are conjugated and repeated to produce the conjugate symmetric counterpart of the original OFDM block. The result is an OFDM block of length N(=2* $\overline{N}$) complex QAM symbols. The first $\overline{N}$ QAM symbols represent the real frequency portion of the frequency spectrum and the remaining $\overline{N}$ QAM symbols represent the imaginary frequencies in the spectrum. Then upon application of the IFFT to each block of length N complex QAM symbols, a block of length N real time domain samples is generated which are converted to serial in preparation for transmission.

2. Guard Time and Cyclic Extension

One of the major advantages of OFDM is its robustness to multipath propagation and the associated delay spread that occurs. The delay spread is the time difference of arrival of transmitted signals that follow different paths from the source to the receiver and therefore have different propagation times.

Dividing the input data stream into $\overline{N}$ subcarriers reduces the symbol duration by $1/\overline{N}$. To combat the effects of ISI a guard time is incorporated into each OFDM symbol. The duration of the guard time is chosen to be at least as long as the maximum expected delay spread. The maximum expected delay spread is the time difference of arrival between the first and last transmitted signals to arrive at the receiver. This ensures that multipath receptions of one symbol do not interfere with the receptions of the following symbol.

One possibility for the guard time is to transmit no signal. However this would result in ICI, which is crosstalk between carriers that results from a loss of orthogonality. Recall that orthogonality means that all subcarriers have an integer number of cycles in the FFT period with adjacent subcarriers having a number of cycles that differ by one. Integration of a subcarrier which has a non-integer number of cycles in the FFT interval will result in a loss of orthogonality and therefore signal degradation due to ICI. This effect is illustrated in Figure 2.5 below.

Figure 2.5 shows 2 subcarriers with the second subcarrier delayed. Note that the guard time consists of a zero signal. Also note that subcarrier one has exactly two cycles within the FFT interval while subcarrier two, due to its delay, does not contain an integer number of cycles within the FFT interval which will cause ICI as discussed above and indicated in Fig. 2.5. The shaded portion of subcarrier 2 indicates the region of the signal that causes ICI. One can see that if the subcarriers were extended cyclically in the guard interval that there would be an integer number of cycles in the FFT interval regardless of the multipath delay of subcarrier 2. The cyclic extension is implemented by simply

extending the length of each OFDM block following the IFFT operation by an amount greater than or equal to the maximum expected delay spread.

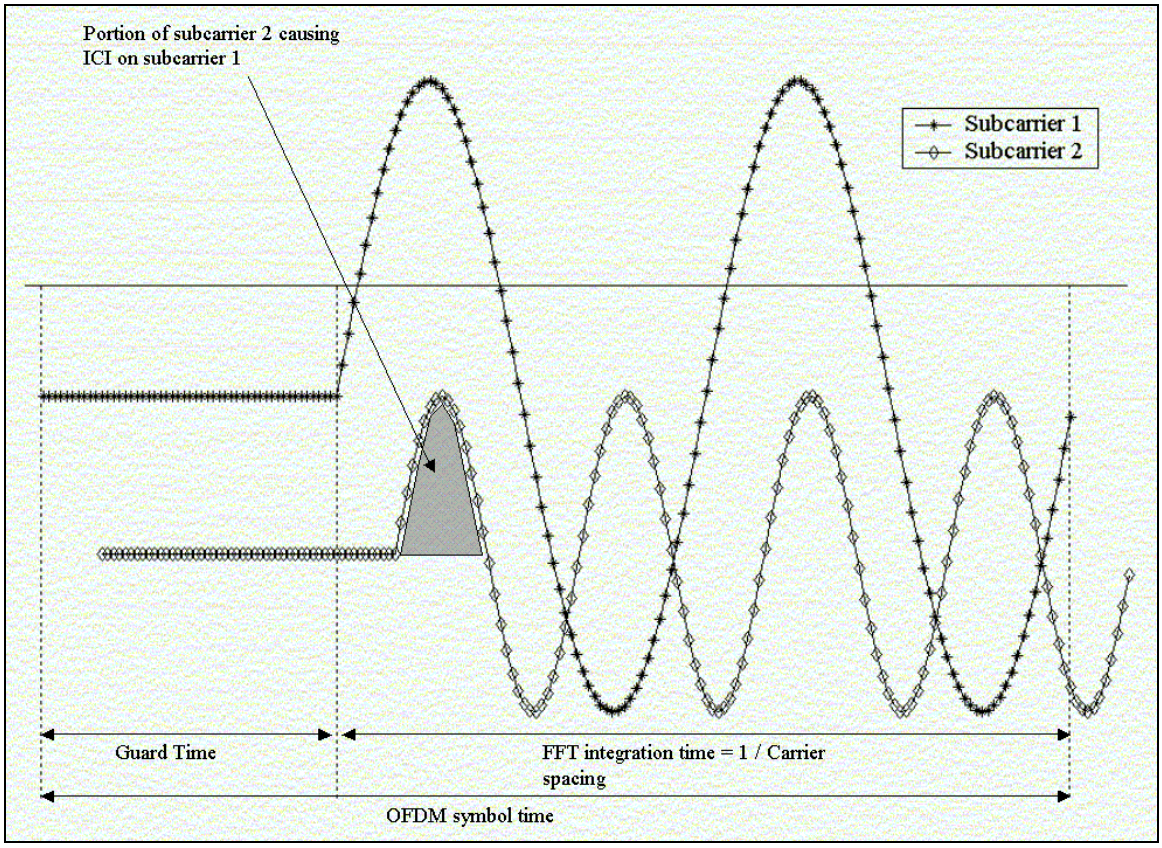


Figure 2.5. Multipath Effect with zero signal in the Guard Time [after Ref. 4].

Figure 2.6 illustrates the cyclic extension of the subcarriers in the guard interval. The benefit of using a cyclic extension in the guard interval is shown in Figure 2.7, which shows a 2 ray multipath environment with the amplitude of the second ray half the amplitude of the first ray. The second ray arrives after the first ray but its delay spread is less than the duration of the guard interval thus there is no ISI.

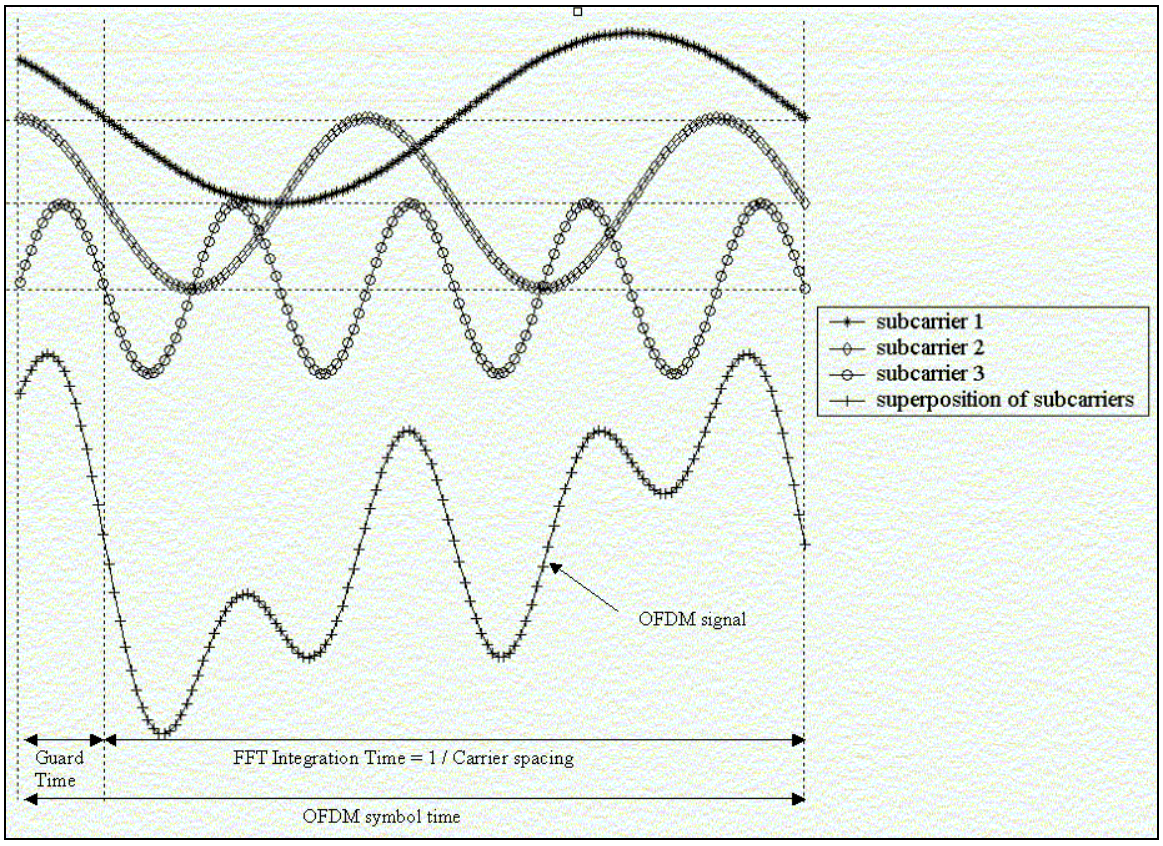


Figure 2.6. OFDM symbol and subcarriers with cyclic extension in the guard interval [after Ref. 4].

In addition, it can be seen that there are an integer number of cycles for rays within the FFT integration time which maintains the orthogonality of the subcarriers and prevents ICI. The subcarriers in Fig. 2.7 are BPSK modulated. Therefore only two possible phases are possible and they differ by 180 degrees. The phase shifts that result are indicated in the figure. Note that the OFDM receiver would actually see the summation of all the received paths but the subcarriers are shown separately for illustration purposes.

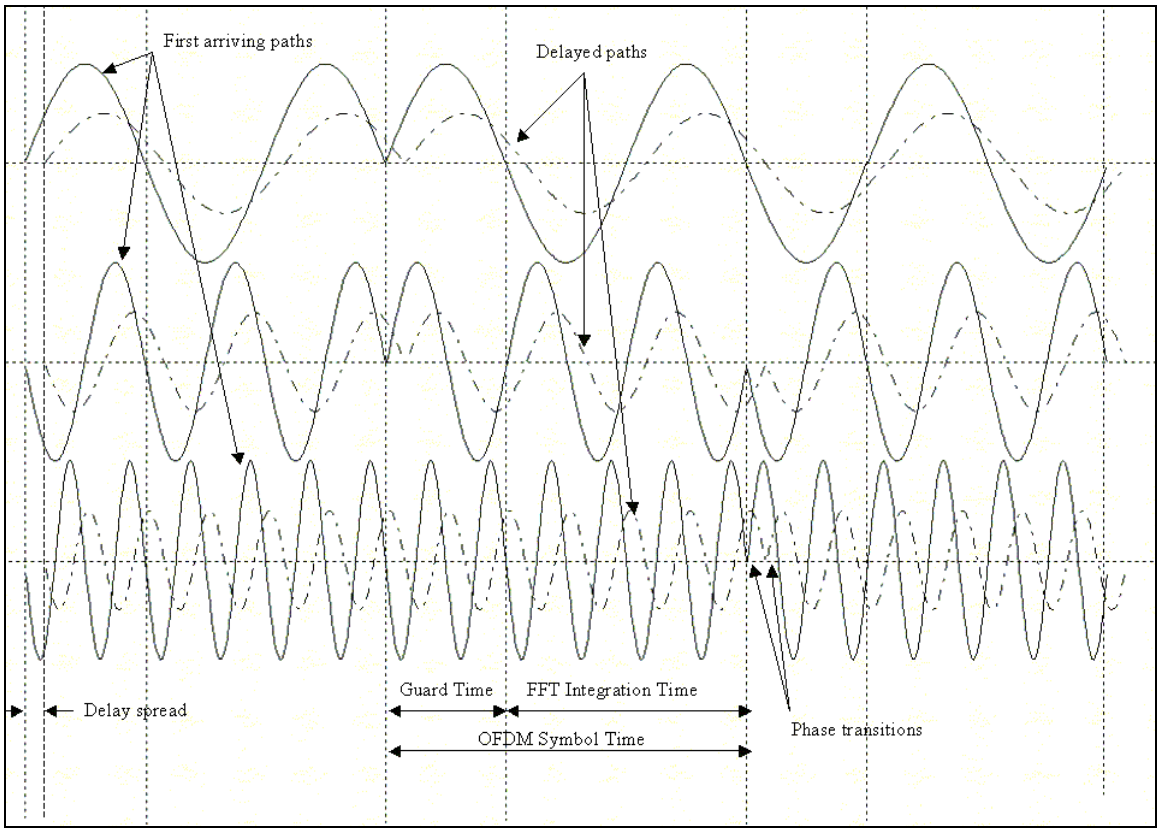


Figure 2.7. OFDM Signal with cyclic prefix, 3 subcarriers, 2 ray multipath environment [after Ref. 4].

As discussed above, the maximum delay spread must not exceed the guard interval incorporated in the OFDM signal or ISI and a loss of orthogonality will result causing ICI. Van Nee and Prasad [Ref. 4] show that with the delay spread greater than the guard interval by less than about three percent, the signal interference is acceptable but at around 10% delay spread excess the interference is unacceptable for communication purposes. However OFDM is still significantly more robust to delay spreads that exceed the value for which the system was designed relative to that for single carrier systems as error propagation causes abrupt performance degradation in single carrier systems [Ref. 4].

B. CHANNEL ANALYSIS

In general communication systems, the input is considered to be the transmitted signal and the output is the received signal. The received signal is the transmitted signal modified by the channel and corrupted by additive noise. The channel is usually band limited and assumed to be linear. It may be either time varying or time invariant. This concept is illustrated in Figure 2.8 below, where x(n) is the input signal, y(n) is the output signal, h(n) is the channel impulse response and w(n) is additive gaussian noise.

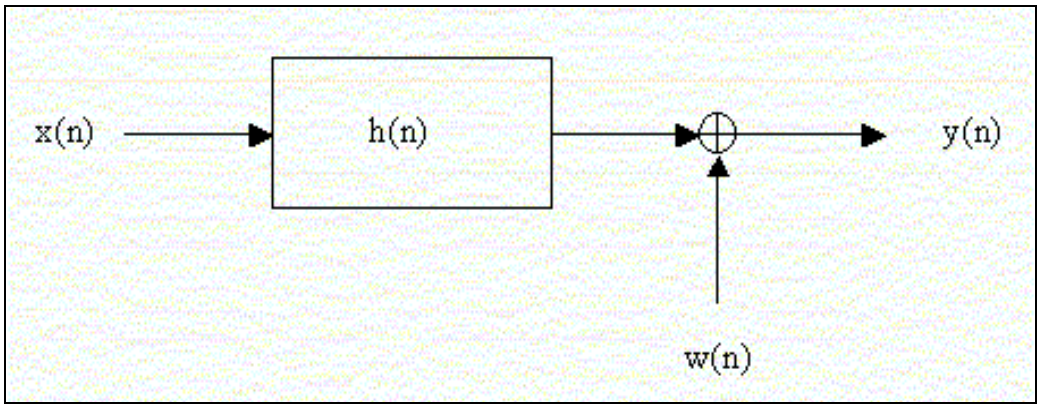


Figure 2.8. Basic Communication System.

In OFDM, each of the $\overline{N}$ subchannels is transmitted with its own carrier. Consider an OFDM symbol defined as

$$X_k = \begin{bmatrix} X_1(k) \\ X_2(k) \\ \vdots \\ X_{\overline{N}}(k) \end{bmatrix}. \qquad (2.5)$$

where k refers to the OFDM block or symbol. As discussed above, X_k must be modified to be conjugate symmetric prior to applying the IFFT. Therefore X_k becomes

$$X_k = \begin{bmatrix} X_1(k) \\ X_2(k) \\ \vdots \\ X_N(k) \end{bmatrix}. \tag{2.6}$$

where $N = 2^* \overline{N}$. The IFFT then modulates the $\overline{N}$ subcarriers resulting in the real time domain signal x_k ,

$$x_k = IFFT[X_k] = \begin{bmatrix} x_1(k) \\ x_2(k) \\ \vdots \\ x_N(k) \end{bmatrix}. \tag{2.7}$$

The channel is assumed to have a finite impulse response of length *L*. Therefore a cyclic extension of length *L* is applied to the OFDM symbol x_k. The OFDM signal becomes

$$x_k = [\underbrace{x_{N-L}(k) \cdots x_{N-1}(k)}_{\text{cyclic extension}}, \underbrace{x_1(k) \cdots x_N(k)}_{\text{IFFT}\{X_k\}}] \quad \text{Transmitted OFDM Signal} \tag{2.8}$$

The output from the channel is the convolution of the channel impulse response with the input signal. Written as an equation this is

$$y_k(n) = x_k(n) \otimes h(n) \tag{2.9}$$

where $\otimes$ represents circular convolution and $y_k(n)$ is the k^{th} received OFDM block. It is well known that for continuous time signals the linear convolution in the time domain is equivalent to multiplication in the frequency domain. In order for this to hold in discrete time one of two conditions must be met. Either the block length N must be infinite or at least one of the convolved sequences must be periodic with period N. Observe that the input OFDM signal is periodic with period N due to the application of the cyclic extension. Therefore the received frequency domain signal is the product of the input OFDM frequency domain signal and the frequency response of the channel. In equation form the *kth* received OFDM symbol, with the subscript k omitted for simplicity, is

$$Y_i = H_i X_i \quad for\ i = 1,2 \ldots \overline{N} \tag{2.10}$$

where the subscript i refers to the subcarrier. Note that the use of the cyclic prefix reduces the achievable data rate by a factor of $N/N+L$. However for most applications $N>>L$ so the loss is negligible relative to the identified benefits of using the cyclic prefix. Most OFDM systems use the cyclic prefix [Ref. 5 and 6].

The received signal is

$$y_k = \underbrace{\left[y_{N-L}(k) \cdots y_{N-1}(k), \underbrace{y_1(k) \cdots y_N(k)}_{[x_1(k), \cdots x_N(k)] \otimes [h(1), \cdots h(L)] + w}\right]}_{\text{Received OFDM Signal}} \tag{2.11}$$

As indicated in Eq. 2.11 the last N values of y_k are the convolution of the OFDM input signal with the channel impulse response. Therefore the received QAM symbols in the k[th] block are written as Eq. 2.12 where the last $\overline{N}$ QAM symbols are the complex conjugate of the first $\overline{N}$ QAM symbols and are discarded in the receiver.

$$Y_k = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_N \end{bmatrix} = FFT \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} H_1 X_1 \\ H_2 X_2 \\ \vdots \\ H_N X_N \end{bmatrix} + \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_3 \end{bmatrix} \tag{2.12}$$

Therefore, the received QAM symbol for the i^{th} subchannel and the k^{th} block can be expressed as

$$Y_{i,k} = H_i X_{i,k} + W_i \quad for\ i = 1,2 \ldots N \tag{2.13}$$

where H is defined as

$$H = \begin{bmatrix} H_1 \\ H_2 \\ \vdots \\ H_N \end{bmatrix} = FFT \begin{bmatrix} h(1) \\ h(2) \\ \vdots \\ h(L) \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{2.14}$$

and $h(n)$ is zero padded to length N in taking the IFFT. As a result the effect on the channel is that it is also divided into narrow subchannels corresponding to the subcarriers. For large $\overline{N}$, the subchannels are very narrow and there is negligible change in $|H_i|$ and

$\angle H_i$ versus frequency. In other words the frequency selective fading that may be present is eliminated. Subchannel independence can be shown provided the additive noise is Gaussian(whether the noise is colored or not) [Ref. 3]. This result is illustrated in Fig. 2.9 below.

The rectangles represent the subchannels that are used by the subcarrier bands. Each subchannel is assumed flat for large *N*. In Fig. 2.9, only 11 subcarriers exist however in practice the number of subcarriers is typically between 500 and 2000. With this in mind one can see that the frequency response within each of the subchannels would be flat as the width of the subchannel would be drastically reduced.

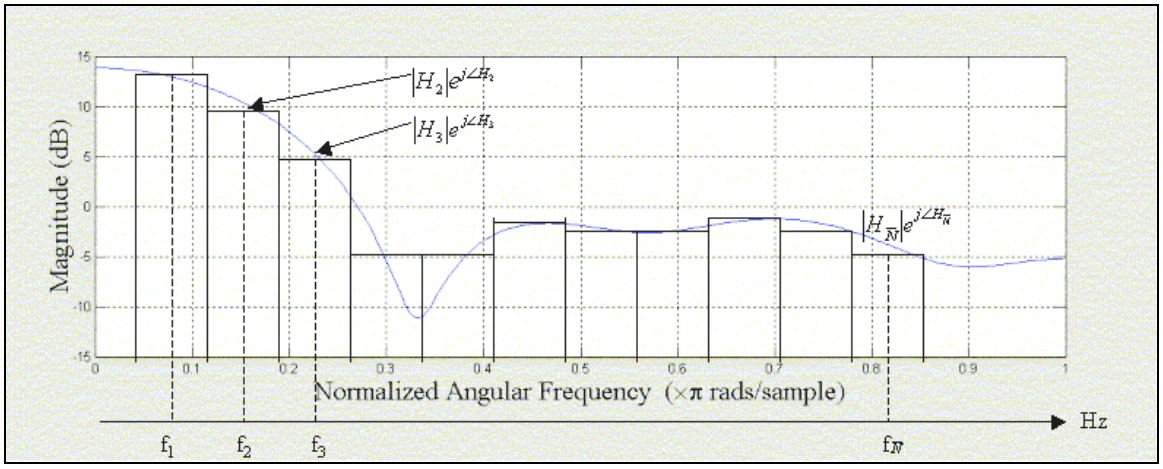


Figure 2.9. Subchannel Decomposition of Channel Frequency Response.

THIS PAGE INTENTIONALLY LEFT BLANK

III. OFDM SIMULATION MODEL

The development of a computer-based simulation of an underwater OFDM communication system is the primary thrust of this thesis. The model simulates most of the significant hardware that would exist in an actual system. This chapter describes the details of the model and the theory behind the methods applied. The intent is not to exhaustively model every component in an actual system but to model those components and features of a real system that contribute the dominating characteristics of the system's performance. It is also understood to be unfeasible and foolish to attempt to develop a model from scratch that will have the ability to experiment with every possible design and operating feature of the system. With that in mind, the model was developed to be comprehensive enough to explore some of the basic specifications common to any communication system immediately but then to also provide a test bed upon which further modeling and experimentation could take place.

A. METHODOLOGY

The development of each block in the model followed essentially the same path. First the topic of interest was explored in the literature to the extent required. The extent of literature research was a function of the number of choices available to accomplish the task, the experience of this author in the subject and the relative importance of the block to the entire model. For example, the modulation and demodulation blocks were only mildly researched while the topic of forward error correction was researched much more extensively. Figure 3.1 provides a block diagram of the entire simulation.

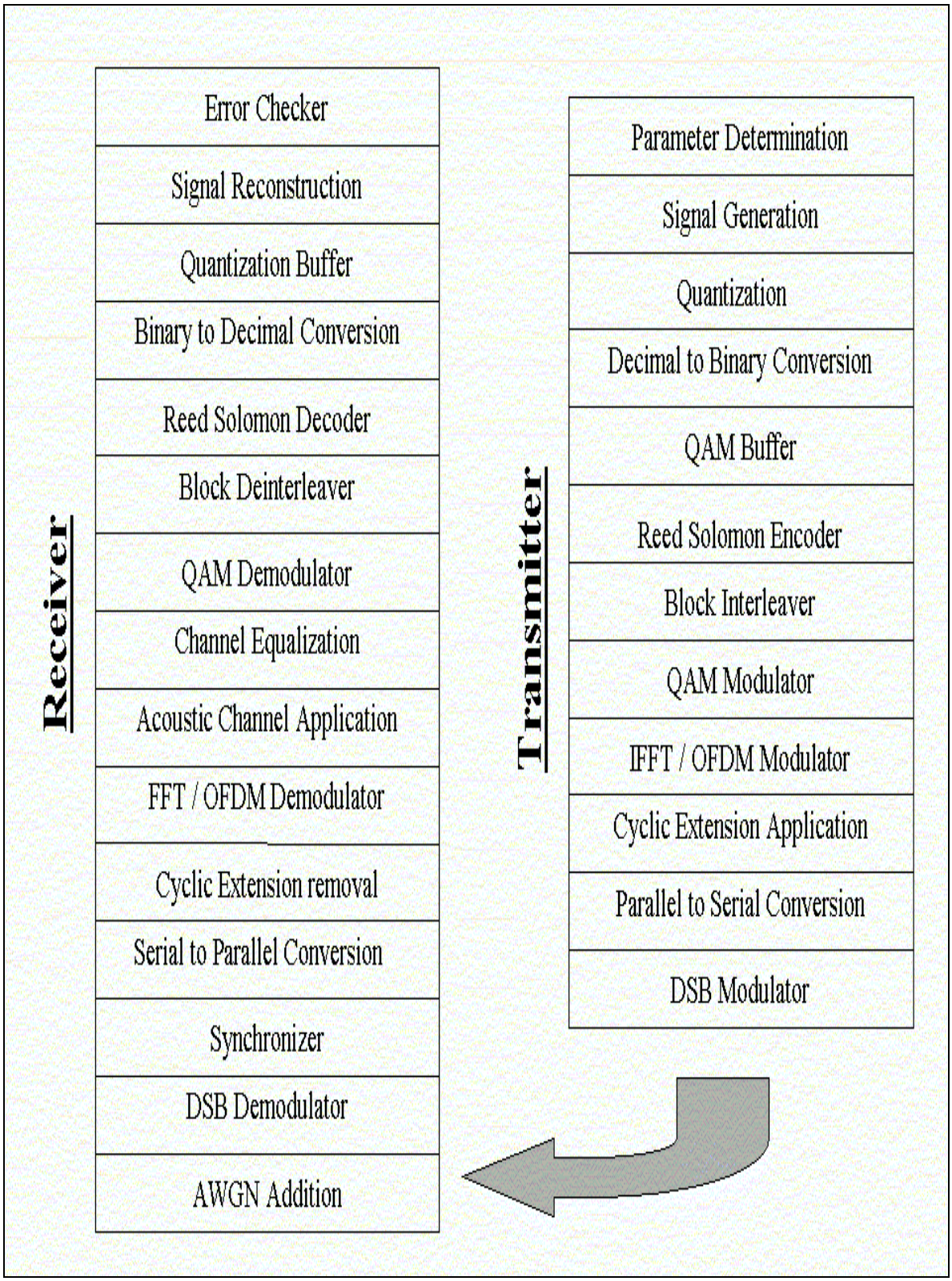


Figure 3.1. OFDM Simulation Block Diagram.

B. TRANSMITTER

The transmitter covers all aspects of signal processing and transmission from information signal development to Quadrature Amplitude Modulation (QAM), to Double Side Band Suppressed Carrier (DSBSC) modulation of the signal. Each of the transmitter blocks of Fig. 3.1 is discussed in the following numbered subsections.

1. OFDM Parameter Determination

In designing a communication system there are several parameters of interest that must be determined. The final design will be a compromise between conflicting requirements to achieve the optimal performance. Parameter determination is performed in the simulation by the Matlab function parameters.m contained in Appendix C. In general there are three main parameters to determine: bit rate, R, channel delay spread, t_{ds}, and bandwidth, W. Delay spread is a function of the channel and therefore represents the opening argument in designing the system. The OFDM system must be designed such that the tolerable delay spread of the system is greater than the delay spread of the channel to avoid signal degradation. Therefore the guard time, t_g, is typically two to four times the maximum expected delay spread. That is,

$$\boxed{2 * t_{ds} \leq t_g \leq 4 * t_{ds}}. \tag{3.1}$$

The delay spread for the underwater acoustic sound channel is a function of but not limited to bottom type, transmission range, transmitter and receiver depths and frequency band. Coatelan and Glavieux [Ref. 1] report measured delay spreads in Table 3.1 below. The table identifies the transmission range between the receiver and the transmitter, the seafloor composition and the associated t_{ds} measured.

Bottom Type	Transmission Range (m)	Delay Spread (msec)
Rocky	< 1000	< 10
Sludgy	1500 – 2500	<10
Sandy	1500 – 2500	> 20
Unknown	1430	~20

Table 3.1. Typical Delay Spread Values.

The next step is to determine the OFDM symbol duration. The OFDM symbol duration is the sum of the cyclic prefix duration and the FFT interval duration. Since the signal in the cyclic prefix does not contain any new information it is desirable to have the FFT interval duration much greater than the guard time to maximize the achievable bit rate. However there is a limit. As T increases, the subcarrier spacing, Δf_{sc}, decreases for a given bandwidth. Smaller subcarrier spacing results in greater sensitivity to phase errors and frequency offset. In practice the symbol duration is typically 5 or more times t_g, therefore

$$T \geq 5 * t_g \quad . \tag{3.2}$$

For channels with frequency selective fading, smaller Δf_{sc} values are preferred as this ensures the subchannel frequency response is flat. Conversely, for channels with large Doppler spreading characteristics larger Δf_{sc} values are better.

Figure 3.2 below presents correlator output results measured by Coatelan and Glavieux [Ref. 1]. The results in Fig. 3.2 correspond to the 1430 m acoustic channel of

Table 3.1 with a sludgy seafloor and a depth of 90 m. Note that the delay spread in Fig. 3.2 is approximately 20 msec and the first path received is not necessarily the most energetic one [Ref. 1].

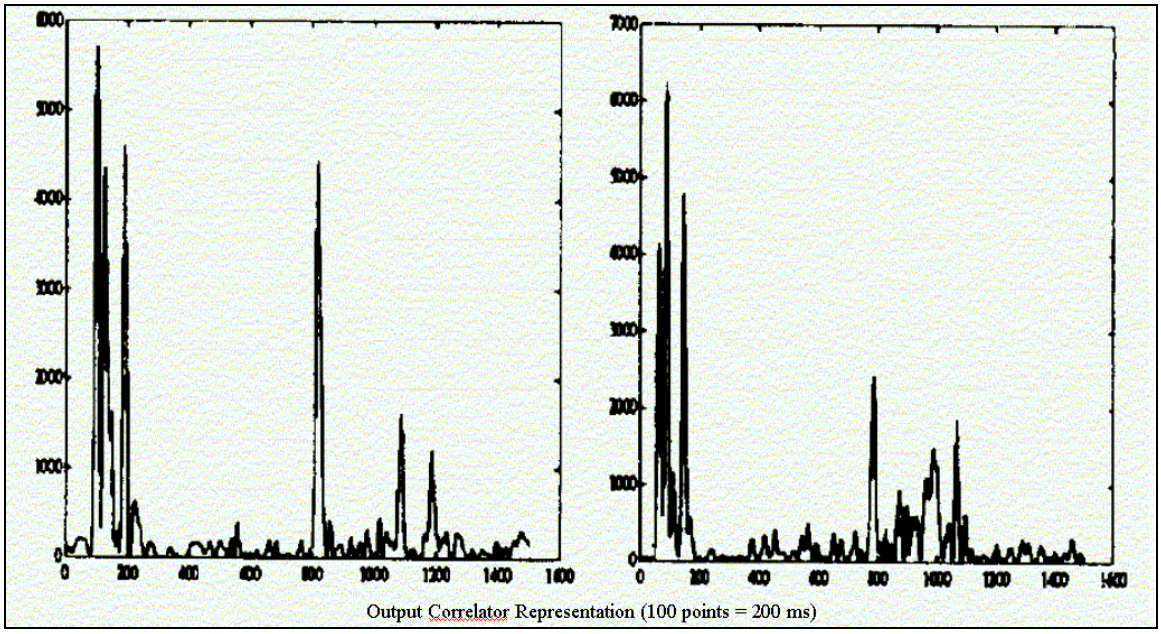


Figure 3.2. Measured Underwater Acoustic Channel Impulse Response [from Ref. 1].

With the guard time and symbol duration established, the number of subcarriers $\overline{N}$ can be determined. There are two approaches that can be taken. The first is starting with a known bandwidth constraint. In this case the number of subcarriers is simply the available bandwidth, W, divided by the subcarrier spacing, therefore

$$\boxed{\overline{N} = \frac{W}{\Delta f_{sc}}}. \tag{3.3}$$

The other approach, which is the one used in the simulation, is to start with a desired bit rate, R. Then the number of subcarriers is simply R divided by the number of bits transmitted by each subcarrier, b_{sc}. That is,

$$\boxed{\overline{N} = \frac{R}{b_{sc}}}. \tag{3.4}$$

a. Example Calculation

Suppose that a system is to be designed that would provide a 10 kbps data rate and the maximum expected delay spread for the acoustic channel of interest is 27 msec. That is,

$$R = 10 \text{ kbps}$$

$$t_{ds} = 27 \text{ msec}$$

let $t_g = 4t_{ds} = 4(27) = 108$ msec

let $T = 6t_g = 6(108) = 648$ msec

Assume that the system is capable of transmitting 6 bits per subcarrier, which corresponds to 64-QAM, then

$$\overline{N} = \frac{R}{b_{sc}} = \frac{10E3}{6} \cong 1667 \text{ subcarriers}$$

However since the QAM symbols on each of the subcarriers will be modulated using the IFFT, it is computationally more efficient to have the number of subcarriers equal to a power of two. Therefore $\overline{N}$ is set to 2048 subcarriers but only 1667 of the subcarriers have non-zero values with the remaining subcarriers providing oversampling to avoid aliasing. In essence the signal itself is unaffected and changing $\overline{N}$ only affects the internal signal processing efficiency. Next the FFT interval duration is calculated as,

$$T_{FFT} = T - t_g = 648 - 108 = 540 \text{ msec}$$

Another requirement is that there be an integer number of samples within both the FFT interval, T_{FFT}, and the entire OFDM symbol, T, in order to maintain orthogonality of the subcarriers. For this example there must be 2048 samples within the FFT interval. The FFT sampling rate, $F_{s,FFT}$, is then,

$$F_{s,FFT} = \frac{\overline{N}}{T_{FFT}} = \frac{2048}{540} \cong 3.793 \text{ kHz}$$

However the number of samples within the OFDM symbol interval, T, is,

$$n_T = F_{s,FFT} * T = 3.793 * 540 = 2457.6$$

which is not an integer. Therefore the subcarriers will not be orthogonal and signal degradation will occur. The remedy is to adjust the sampling rate slightly. The sampling rate is increased to achieve an integer number of samples within the FFT interval and the OFDM symbol interval. For this example $F_{s,FFT}$ becomes 3842.6 Hz, which yields

$$n_T = F_{s,FFT} * T = 3842.6 * 648E-3 = 2490 \text{ samples}$$

$$n_{FFT} = F_{s,FFT} * T_{FFT} = 3842.6 * 540E-3 = 2075 \text{ samples}$$

$$n_g = F_{s,FFT} * t_g = 3842.6 * 108E-3 = 415 \text{ samples}$$

where n_T, n_{FFT} and n_g are the number of samples in the OFDM symbol interval, the FFT interval and the guard interval respectively. Note that the OFDM symbol is oversampled by increasing the sampling rate since there are 2075 vice 2048 samples within the FFT interval. Synchronization, which will be discussed in the receiver section, will determine which of the 2075 samples in the FFT interval are to be used for signal reconstruction. As a result of changing the sampling rate,

$$T_{FFT} = \frac{n_{FFT}}{F_{s,FFT}} = 0.5330 \text{ sec}$$

$$t_g = \frac{n_g}{F_{s,FFT}} = 0.115 \text{ sec}$$

and finally,

$$\Delta f_{sc} = T_{FFT}^{-1} = 1.876 \text{ Hz}$$

$$W = \overline{N} * \Delta f_{sc} = 2.0264 \text{ kHz}$$

2. Signal Generation

In underwater communication several possible data forms may be transmitted including video, voice and text format messages. For the simulation, voice data in .wav format is used as the information signal. Voice data was chosen as it requires an analog signal to be converted to binary format via sampling, quantization and decimal to binary conversion. Once the data is in binary form the original form of the data is irrelevant to the OFDM system. Figure 3.3 presents the information signal used in the simulation. The upper plot is the discrete time signal. The sampling rate of the analog voice signal is 11.025 kHz and the data is quantized using 8 bits per sample. The lower plot is the frequency spectrum of the transmitted signal.

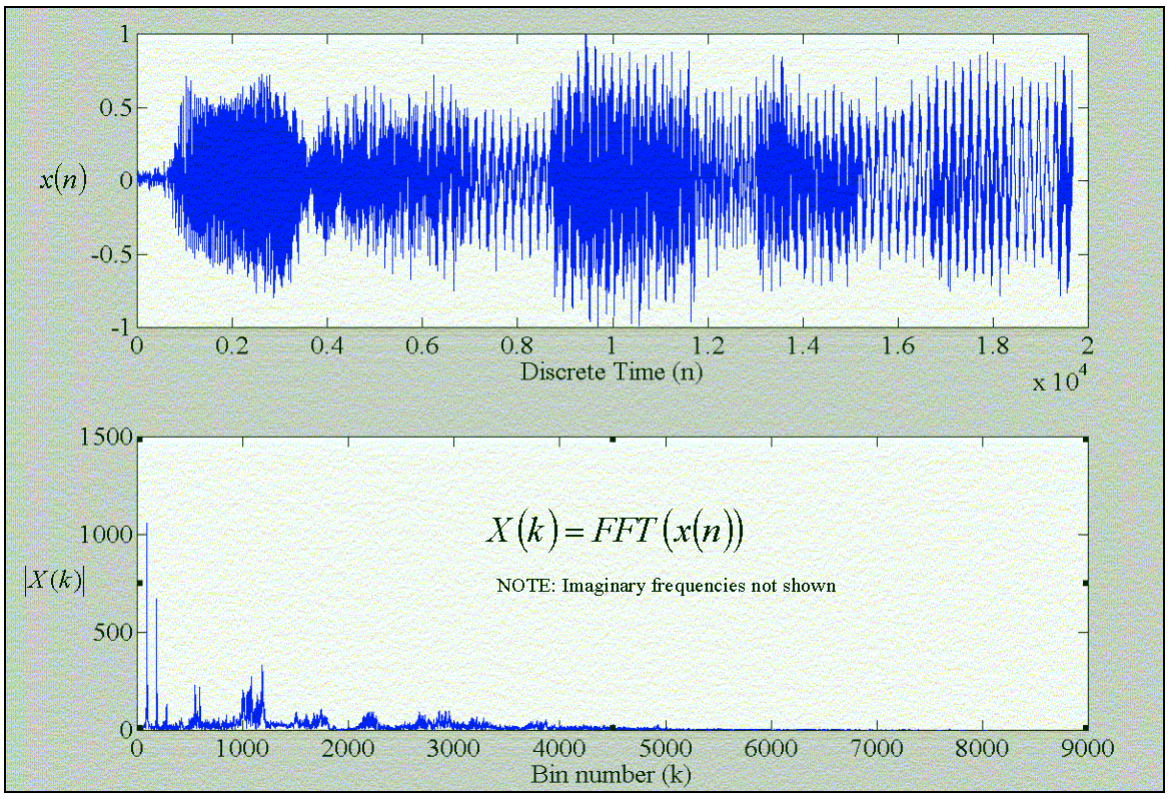


Figure 3.3. Information Signal.

3. Quantization

The sampled information signal is then quantized using uniform quantization. The level of quantization is adjustable but 8-bit quantization is primarily used in the simulation which results in 256 quantization levels. Quantization is performed by the Matlab function quantization.m contained in Appendix C.

The output of the quantizer is a stream of decimals from 0 to M-1 where M is 2^{bits} and *bits* is the number of bits per binary digits per quantization level. The decimal value corresponds to the quantization level for which the sampled value of the information signal was nearest. The quantizer also outputs a codebook that identifies the quantization level corresponding to the decimal value for signal reconstruction. Figure 3.4 clarifies the quantization process with a simple 2-bit example.

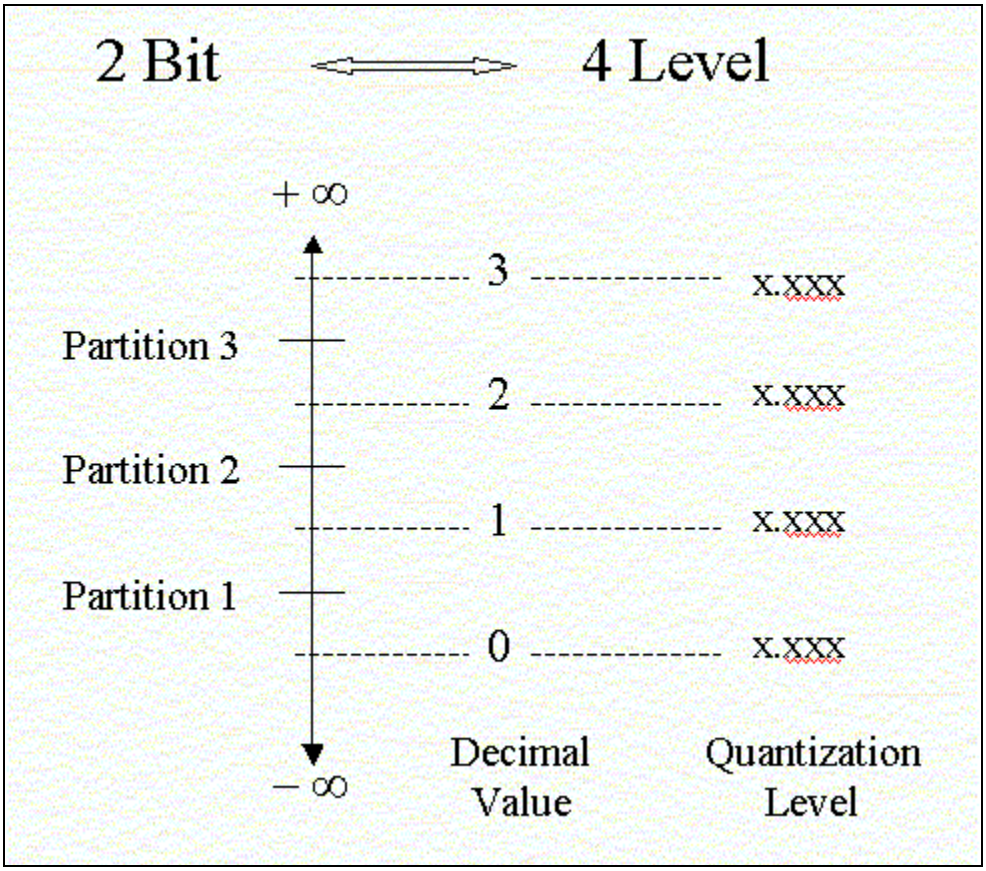


Figure 3.4. Quantization Illustration.

4. Decimal to Binary Conversion

Decimal to binary conversion is performed by the Matlab code binary_conversion.m contained in Appendix C. The process involves straightforward conversion of a base 10 integer to the corresponding base 2 representation.

5. QAM Buffer

QAM modulation involves the mapping of bits to a QAM constellation. The size of the constellation, Q, is a function of the number of bits per QAM symbol.

$$Q = 2^{q} \tag{3.5}$$

In general the number of bits per QAM symbol will not be the same as the number of bits per quantization level. Therefore it is necessary to buffer the bits prior to mapping to a QAM constellation. The Matlab code qam_buffer.m contained in Appendix C performs the buffering operation. This involves converting the bit stream from the decimal to binary conversion process into a matrix of bits. Each row in the matrix represents a QAM symbol in binary form and the number of columns per row is q. The number of rows is a function of the size of the data to be transmitted. Since the matrix must be square in order to convert all bits to a QAM symbol, up to q-1 zeroes are added to the data.

6. Reed Solomon Encoder

Due to the multipath, frequency selective fading nature of the underwater acoustic sound channel the subcarriers will arrive with different amplitudes. Some subcarriers may experience severe enough fading that they are lost entirely. As a result the bit error for those subchannels will approach ½ and will dominate the BER performance of the entire system[Ref. 4].

Forward error correction coding is used to combat the severe fading affect. The coding is applied across all subcarriers such that the errors of the weak subcarriers can be

corrected. The performance of the system is then a function of the average received power vice the received power of the weak subcarriers [Ref. 2].

In the simulation block coding is applied, specifically Reed Solomon block coding which is performed at the symbol level. Coding involves the addition of redundant symbols to increase the Hamming distance between symbols. In general a message of length *k* is added thereby creating a code word of length *n*. The coding rate *r* is defined as *k/n*. The minimum Hamming distance, d_{min}, is the minimum number of different symbols between any 2 code words.

For a given code with an associated d_{min}, *t* errors can be corrected by the code where *t* is

$$t \leq floor\left(\frac{d_{\min}-1}{2}\right). \tag{3.6}$$

where the floor function denotes rounding down to the nearest integer[Ref. 2]. The minimum Hamming distance has an upper bound defined by the number of redundant symbols as

$$d_{\min} \leq n-k+1. \tag{3.7}$$

In terms of bit level codes, only repetition and single parity check codes reach this upper bound. Reed Solomon is a class of non-binary codes that reach this upper bound.[Ref. 4]

Reed Solomon codes are defined for code words of length *n* symbols associated with *m* bits per symbol where

$$n = 2^m - 1. \tag{3.8}$$

The required message length, *k*, is related to message and the number of bits per symbol as

$$k = 2^m - d_{\min}. \tag{3.9}$$

Associated with the number of symbol errors the code can correct is the number of bits the code is capable of correcting, t_b, defined as

$$\boxed{t_b \leq m * floor\left(\frac{n-k}{2}\right)}. \tag{3.10}$$

Equation 3.10 is true provided the bit errors occur within the maximum correctable number of symbol errors. For example consider a code designed to be able to correct 3 symbol errors with 4 bits per symbol. The code will not be able to correct an arbitrary occurrence of 4 bit errors as they may occur in 4 different symbols. As a result Reed Solomon codes are useful for 'bursty' error channels such as the multipath, frequency selective fading underwater acoustic channel since the errors are concentrated in a few of the subcarriers that are deeply faded.

The Reed Solomon encoder receives QAM coded decimals in serial format from the QAM buffer. The vector of decimal symbols must be reordered to a *k* column square matrix. Therefore subsymbols may need to be added to create a square matrix, in which case they are added to the beginning of the vector.

7. Block Interleaver

The frequency selective fading causes varying amplitudes of the subcarriers where some of the carriers will be unreliable due to severe fading. As a result burst errors vice random errors will occur over the weak subcarriers. Most FEC codes are designed for random errors not burst errors[Ref. 4]. The transmitter permutes the data such that the adjacent symbols are separated by several blocks after interleaving. The receiver performs the inverse operation.

As discussed Reed Solomon codes are designed to correct a certain number of errors per block. The interleaving must distribute the subsymbols across several blocks such that the bursts caused by the severely faded subcarriers appear random in the receiver after de-interleaving. Therefore rather than having bursts of symbol errors within

each block corresponding to the severely faded subcarriers, the errors within each block will be randomly distributed which is optimal for the Reed Solomon decoding algorithm.

The interleaving process is essentially a reordering of the data and is performed by the reshape.m command in Matlab.

The input data to the interleaver is an n column matrix where the data is ordered in time sequence across the rows. Therefore the sequential order of the data in the matrix begins at row 1, column 1, then proceeds across row 1 to column n, then to row 2, column 1 etc.

The reshape function takes the data column wise such that the subsymbols are interleaved in frequency. The process is illustrated in Fig. 3.5 with n equal to 7 and k equal to 3. In the simulation the length of the input vector A is on the order 10^4, n is at most 64, and the number of active subcarriers per OFDM symbol is typically less than 10^3 . Therefore the interleaving is over several blocks as desired.

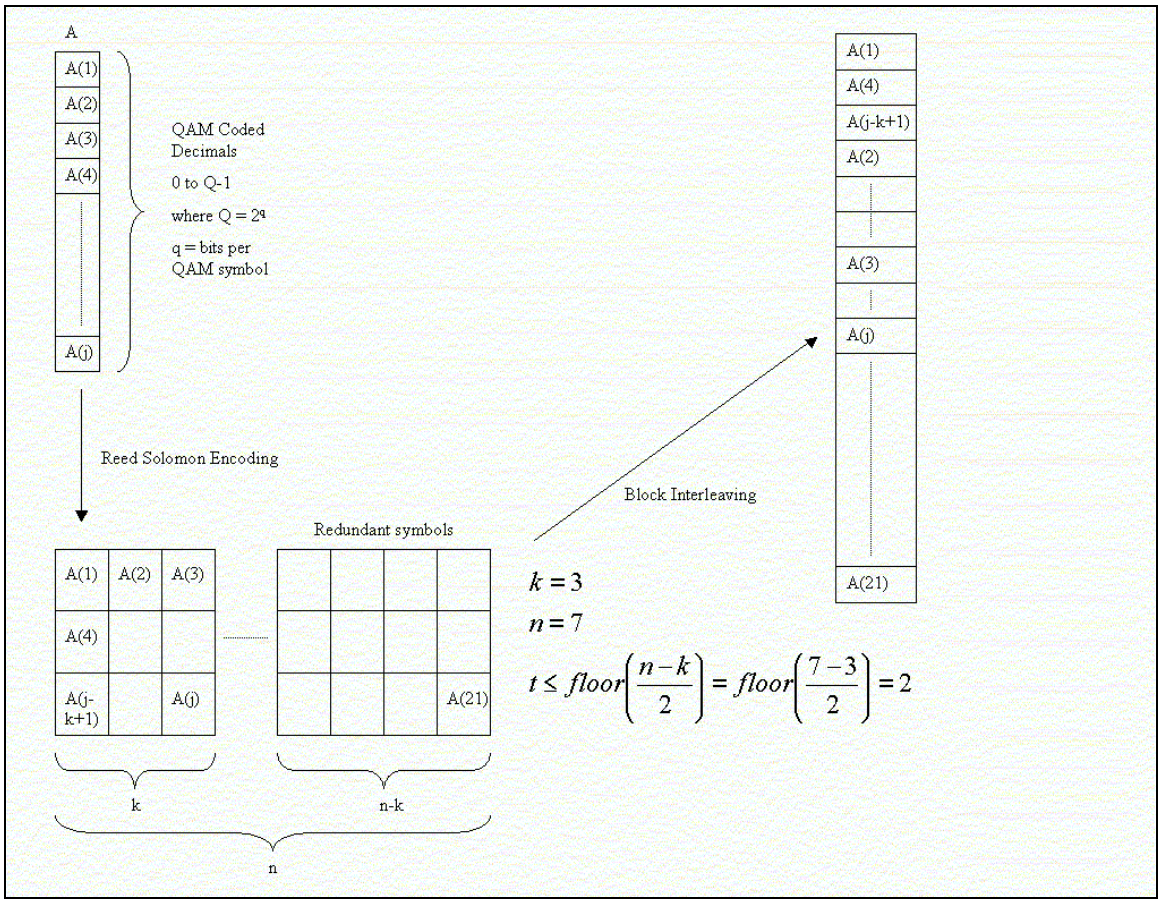


Figure 3.5. Reed Solomon Encoder and Block Interleaver.

8. QAM Modulator

QAM Modulation maps the QAM coded decimals representing points on a QAM constellation into real and imaginary components. The real component is called the in-phase component, and the imaginary component is called the Quadrature component of the signal. The mapping from decimal to real and imaginary components is illustrated in Fig. 3.6 below.

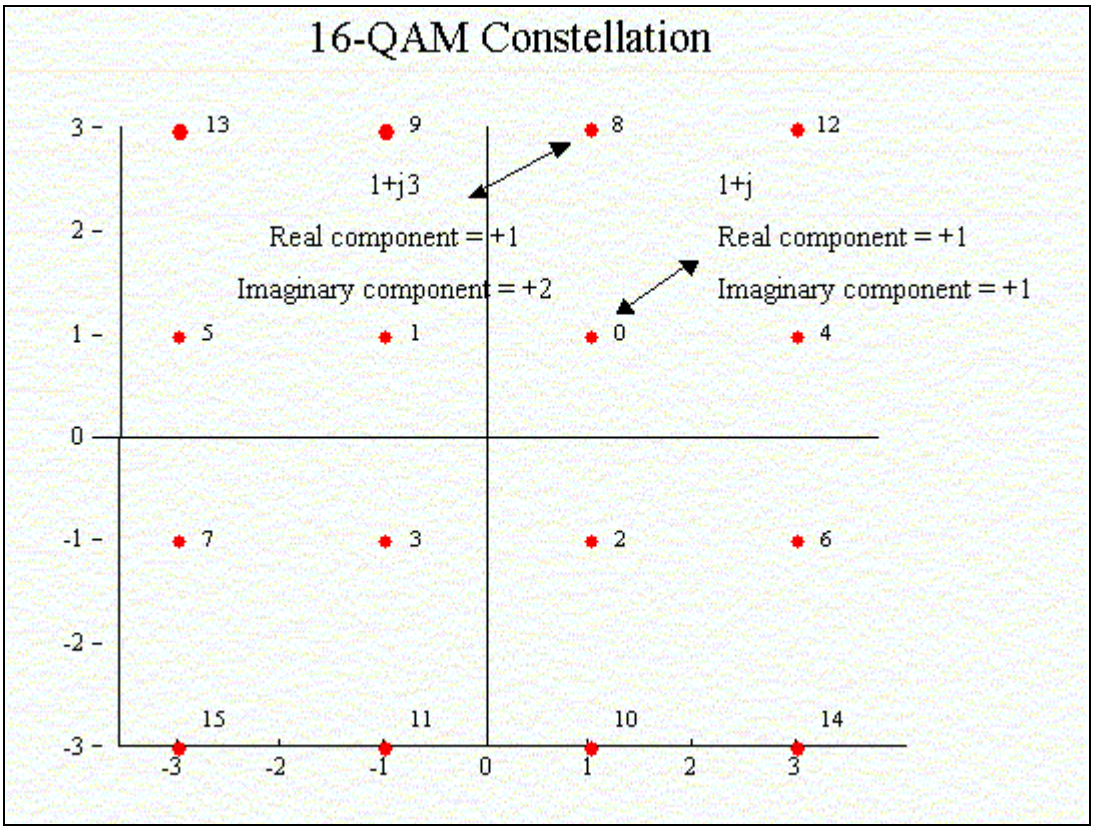


Figure 3.6. 16 QAM Constellation with In-phase and Quadrature Mapping.

The Cartesian coordinates of in-phase and Quadrature components are converted to polar coordinates prior to creating the OFDM symbol blocks. The QAM Modulator creates the frequency spectrum of the signal using the complex QAM symbols now in magnitude and phase components. The goal is to generate blocks of complex QAM symbols such that applying them to the IFFT operation results in a real signal. The IFFT is given by

$$x_n = \frac{1}{\sqrt{N}} \sum_{i=1}^{N} X_i e^{-j\frac{2\pi}{N}in} \quad (3.11)$$

where N is the length of the DFT. The discrete time signal x_n will be real provided the following condition of conjugate symmetry holds for the frequency domain signal X_i,

$$X_i = \begin{cases} X_i & i = 2 \ldots \overline{N} \\ \mathrm{Re}\left\{X_{\overline{N}+1}\right\} & i = 1 \\ \mathrm{Im}\left\{X_{\overline{N}+1}\right\} & i = \overline{N}+1 \\ X^*_{N-i+1} & i = \overline{N}+2 \ldots N \end{cases}. \tag{3.12}$$

The OFDM blocks are then formed by sequentially filling the active subcarriers in each block with the complex QAM symbols.

In the parameter determination algorithm the number of active subcarriers, $\overline{N}_{act}$, and the number of zero subcarriers, $\overline{N}_{zer}$, are determined. The zero subcarriers are required to prevent aliasing from signal power near the Nyquist sampling rate which corresponds to $\overline{N}$. The number of data blocks, N_{block} required to transmit a given amount of data is given by the number of complex QAM symbols to be transmitted divided by the number of active subcarriers rounded up to the nearest integer. As a result of rounding up to get the N_{block}, there is a surplus of active subcarriers relative to the number of QAM symbols to be transmitted. In future work this surplus could be filled with protocol information. The surplus subcarriers are filled with zero symbols in the simulation.

The original design placed the required number of zeroes at the beginning of the first block. However, this was found to be a poor choice as it results in an uneven power distribution across the frequency spectrum. The current design spreads the zero symbols across the frequency band used for transmission.

The number and location of zero symbols to be inserted are determined according to

$$N_0 = N_{block} * \overline{N}_{act} - N_{QAM,i}. \tag{3.13}$$

where N_0 is the total number of zero symbols to be inserted and $N_{QAM,i}$ is the total number of complex QAM symbols to be transmitted. In general N_0 will not be a multiple of N_{block}, therefore the number of zero symbols inserted will not be the same for each

OFDM block. The following method is used to determine the number of zeroes inserted into each OFDM block.

$$\boxed{\begin{aligned} Z_{col} &= floor\left(\frac{N_0}{N_{block}}\right) \\ Z_{col,1} &= N_0 - (N_{block} - 1) * Z_{col} \end{aligned}} \tag{3.14}$$

where $Z_{col,1}$ is the number of zero symbols inserted into the first OFDM block and Z_{col} is the number of zero symbols inserted into the remainder of the OFDM blocks. The process is illustrated in Fig. 3.7 below.

Application of Eq. 3.12 to the $\overline{N}$ QAM symbols in each of the OFDM blocks results in OFDM symbols which are conjugate symmetric. The conjugate symmetry will ensure that each of the OFDM blocks of complex QAM symbols will result in $N(=2\overline{N})$ real samples upon application of the IFFT. The result is shown in Fig. 3.8 with the same parameters as introduced in Fig. 3.7. The symbol $*$ in Fig. 3.8 refers to complex conjugation.

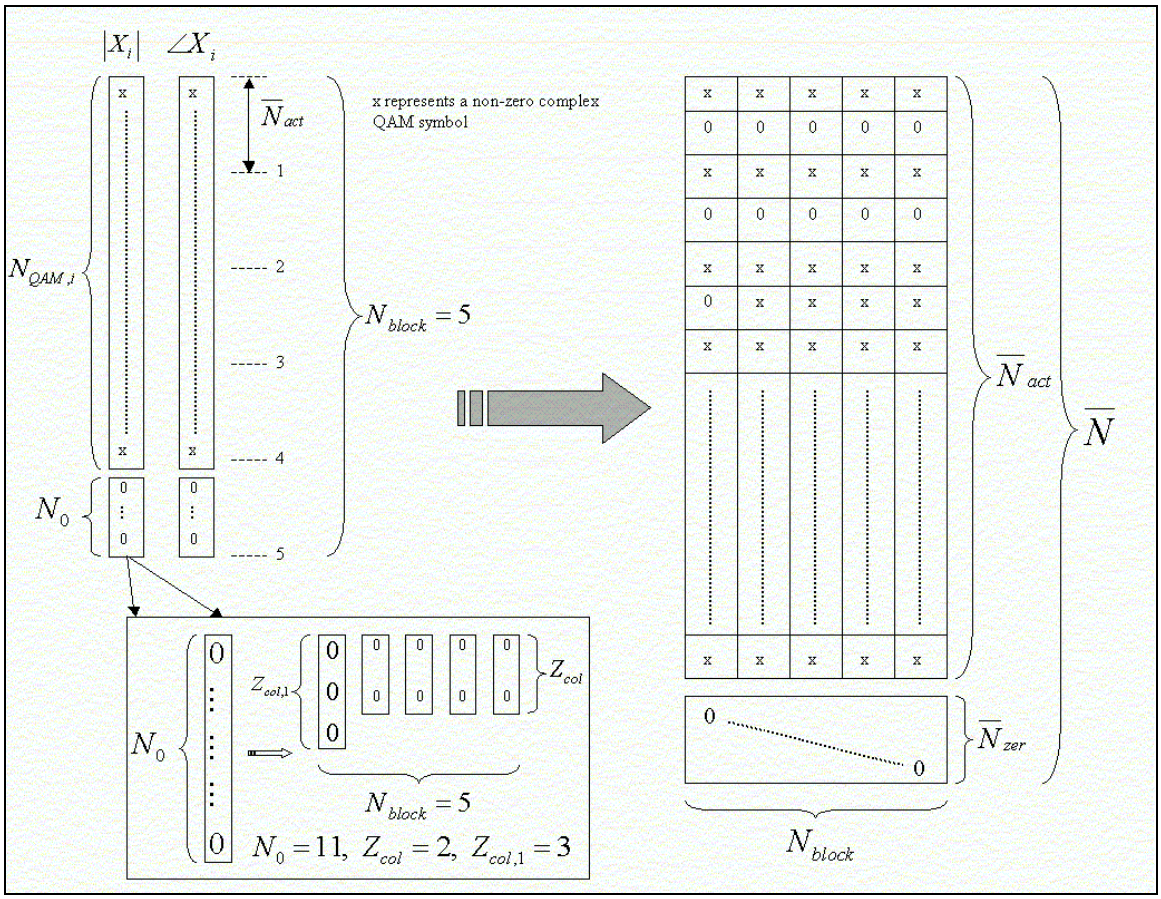


Figure 3.7. QAM Modulator.

The QAM Modulator also creates the pilot blocks which are used for channel estimation and equalization. Although the theory behind channel estimation and equalization will be discussed later, the creation of the pilot blocks is presented here.

In simple terms the pilot blocks are symbols known to the receiver that are used to estimate the channel frequency response. In the simulation two pilot blocks are implemented and make up the first two OFDM symbols that are transmitted. The pilot blocks are made up of pseudo random complex OFDM symbols which are determined as follows.

The data used to create the pilot blocks comes from the data blocks illustrated in Fig. 3.7. Each complex QAM symbol is drawn from the data blocks and placed in the first pilot block as follows,

$$X_{p,1}(c,r) = X_h(r,c) \quad for \begin{cases} r = \overline{N}_{act} + 1, \overline{N}_{act}, \ldots 1 \\ c = N_{block}, N_{block}, \ldots 1 \end{cases} \quad (3.15)$$

Equation 3.12 is applied to the first $\overline{N}$ complex QAM symbols creating a length N block of complex QAM symbols that is conjugate symmetric. Reversing the order of the first pilot block creates the second pilot block.

The final effect of the QAM Modulator is that by adding the complex conjugate counterpart to each of the $\overline{N}$ complex QAM symbol blocks the input data is effectively interpolated. Typically interpolation involves the insertion of one or more zeroes in a time sequence with the effect of increasing the sampling rate and 'compressing' the frequency spectrum of the sequence. In the QAM Modulator the interpolation constant is 2 as the number of data points is doubled. The result is that the sampling rate, $F_{s,FFT}$, is doubled. Therefore the new sampling rate in the FFT interval is,

$$F_s^{'} = 2F_{s,FFT} \quad (3.16)$$

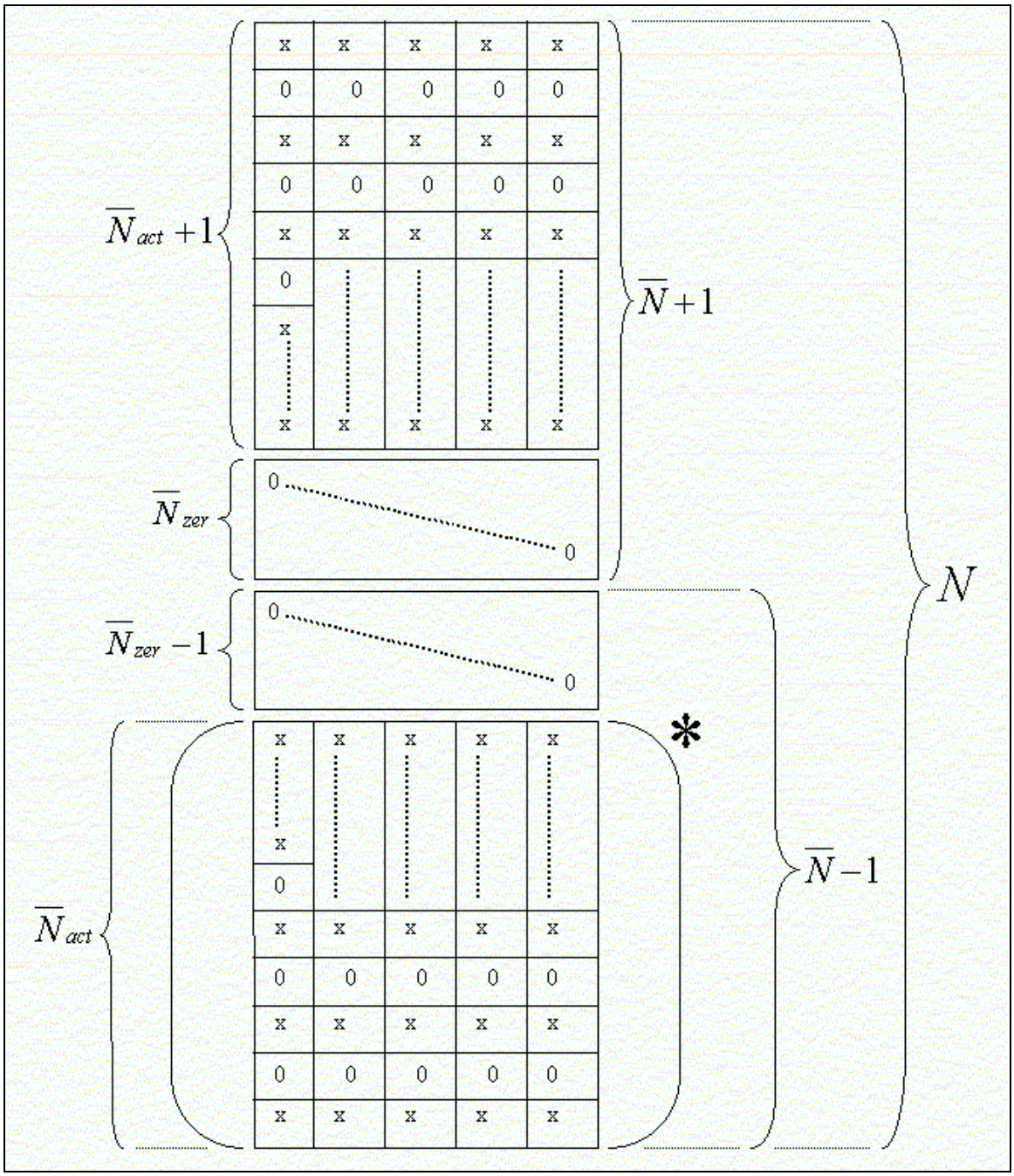


Figure 3.8. QAM Modulator with entire Complex OFDM Symbol Structure.

9. IFFT/OFDM Modulator

The IFFT is used to modulate the complex QAM symbols to the subcarrier frequencies. In other words each subcarrier is modulated by the corresponding complex QAM symbol.

The IFFT can be expressed as

$$x(n) = \frac{1}{N} \sum_{1}^{N} X(k) W_N^{-kn} \qquad for\ n = 1,2 \ldots N \tag{3.17}$$

where

$$W_N = e^{-j\frac{2\pi}{N}}. \tag{3.18}$$

The IFFT can also be expressed in matrix form as

$$\underline{x}_N = \frac{1}{N} \underline{W}_N^* \underline{X}_N. \tag{3.19}$$

where $\underline{W}_N \in C^{N \times N}$ and $\underline{X}_N \in C^{N \times M}$ therefore $\underline{x}_N \in R^{N \times M}$ for $\underline{X}_N$ conjugate symmetric as defined in Eq. 3.12.

Note that *C* refers to the complex subspace and *R* refers to the real subspace and the *M* refers to the number of columns in the matrix. In the simulation *M* is N_{block}. The output from the IFFT is a matrix of real samples with each column representing a baseband real OFDM symbol. Columns 1 and 2 are the pilot symbols and the remaining columns contain the information to be transmitted.

10. Cyclic Extension

The need for the cyclic extension was discussed in Chapter II. The implementation of the cyclic prefix in the simulation is quite simple. First the required length of the cyclic extension is determined by

$$\boxed{L = round\left(t_g * F_s^{'}\right)} \qquad (3.20)$$

where L is the number of discrete time samples that must be prefixed to each real OFDM symbol.

Recall that $\underline{x}_N$ from the IFFT is an N x N_{block} matrix of real time samples therefore the cyclic extension results in a matrix that is $N+L$ x N_{block} with the first L samples being identical to the last L samples in each column. Therefore the resulting matrix is

$$\underline{x}_{N,c}(i) = \begin{bmatrix} \underline{x}_N(N-L+1,i) \\ \underline{x}_N(N-L,i) \\ \vdots \\ \underline{x}_N(N,i) \\ \underline{x}_N(1,i) \\ \underline{x}_N(2,i) \\ \vdots \\ \underline{x}_N(N,i) \end{bmatrix} \qquad for\ i = 1,2 \ldots N_{block} + 2 \qquad (3.21)$$

The columns of the matrix of Eq. 3.21 represent the real baseband OFDM signals that are to be transmitted.

11. Parallel to Serial Converter

The parallel to serial converter simply reshapes $\underline{x}_{N,c}$ to a vector of length (N_{block}+2)*(N+L). In the simulation, round-off error causes the output from the IFFT to contain small imaginary components. Therefore the real part of the serial time samples is taken.

12. Double Side Band Modulator

After serial conversion the signal is baseband and must be modulated to the desired transmission band. The modulation is performed in discrete time to take advantage of the ability to use high order digital filters.

In order to prevent aliasing the signal must be interpolated. The interpolation factor, I, is determined as

$$I = \frac{2\left(f_c + W\right)}{F_s'} \tag{3.22}$$

where f_c is the carrier frequency.

Interpolation is performed by inserting I-1 zeroes between each sample. The interpolated signal can be expressed as

$$y(m) = \begin{cases} x\left(m/I\right) & m = 0, \pm I, \pm 2I \\ 0 & \text{otherwise} \end{cases} \tag{3.23}$$

where $x(n)$ is the baseband signal and $y(m)$ is the baseband signal following interpolation by the factor I. The increased sampling rate is then

$$F_{s,\text{mod}} = IF_s' \tag{3.24}$$

Figure 3.9 illustrates the effect of interpolation on the signal. The signal of Fig. 3.9 is for a 5 kbps transmission rate using 16-QAM. There are 1024 subcarriers with 750 active subcarriers which results in a bandwidth, *W*, of 1.501 kHz. The baseband signal is interpolated by a factor of 4 as shown in the lower graph of Fig. 3.9.

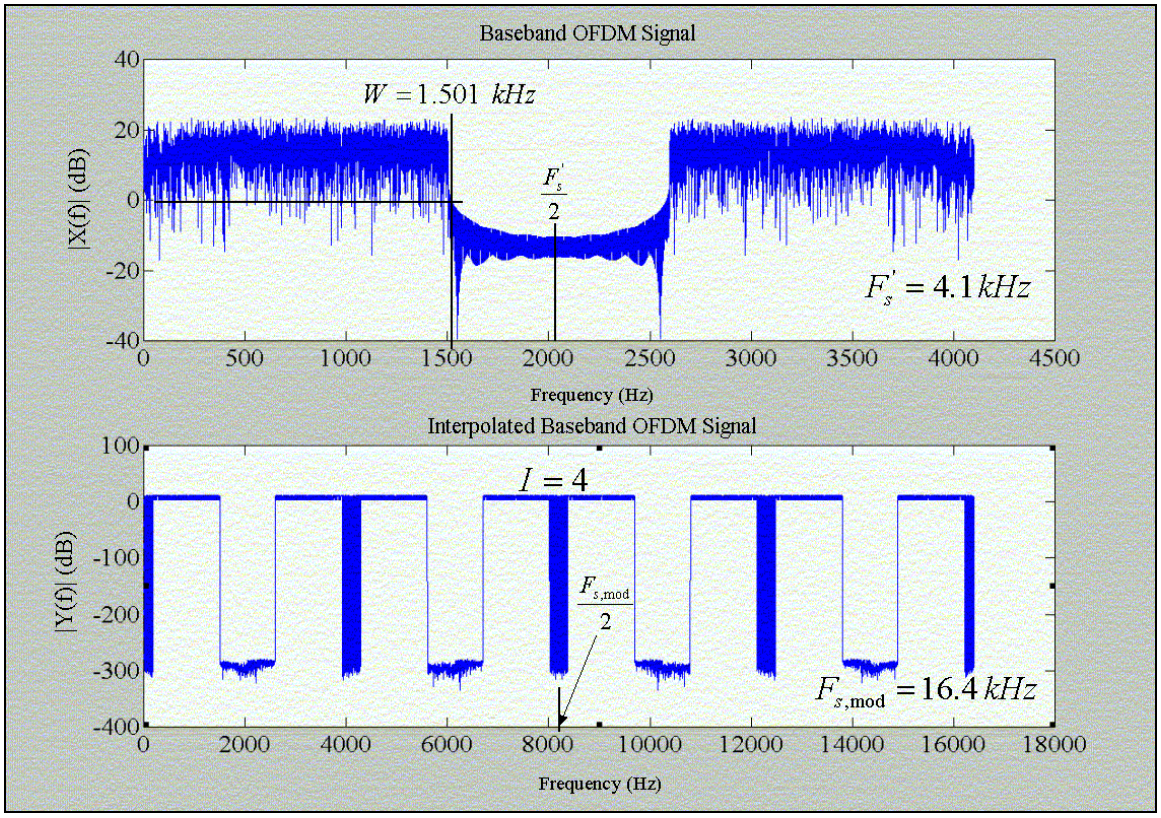


Figure 3.9. Interpolated Baseband OFDM Signal.

The interpolated signal, *y(m)*, must be filtered to remove the unwanted spectral energy above the OFDM signal bandwidth, *W*, as seen in Fig. 3.9. A digital Finite Impulse Response (FIR) filter is used. The FIR filter is designed using a Hamming window with the following parameters.

$$\omega_p = \frac{1.2(2\pi W)}{F_{s.\,\mathrm{mod}}} \equiv \text{pass band digital frequency}$$

$$\omega_s = \frac{1.375(2\pi W)}{F_{s,\mathrm{mod}}} \equiv \text{stop band digital frequency}$$

$$N_h = \frac{8\pi}{\omega_s - \omega_p} \equiv \text{filter order (number of coefficients is } N_h + 1)$$

The filter parameters are shown for a general FIR filter designed using the Hamming window method in Fig. 3.10. The variable ω, is the digital frequency defined as

$$\omega = \frac{2\pi f}{F_s} \tag{3.25}$$

where f refers to the analog frequency and F_s is the sampling rate used to sample the continuous time signal. The output of the FIR filter is defined as,

$$y_f(n) = h(n) * y(n) \tag{3.26}$$

where $h(n)$ is the impulse response of the FIR filter and symbol * refers to linear convolution.

Figure 3.11 shows the effect of the FIR filter on the interpolated baseband spectrum shown in the lower graph of Fig. 3.9. The interpolated baseband frequency spectrum is repeated in the upper graph of Fig. 3.11. The lower graph shows the frequency spectrum of the interpolated baseband frequency spectrum following filtering. The transmission parameters are the same as for Fig. 3.9. The frequency response of the FIR filter is shown in Fig. 3.12.

After interpolation and filtering, the signal is ready for modulation to the desired carrier frequency. The modulation is performed by discrete time domain multiplication with a cosine modulating signal. The modulating signal is defined as

$$m(n) = \cos\left(\frac{2\pi f_c n}{F_{s,\mathrm{mod}}}\right) \tag{3.27}$$

where n is defined from 1 to the number of samples contained in the signal to be modulated, $y_f(n)$.

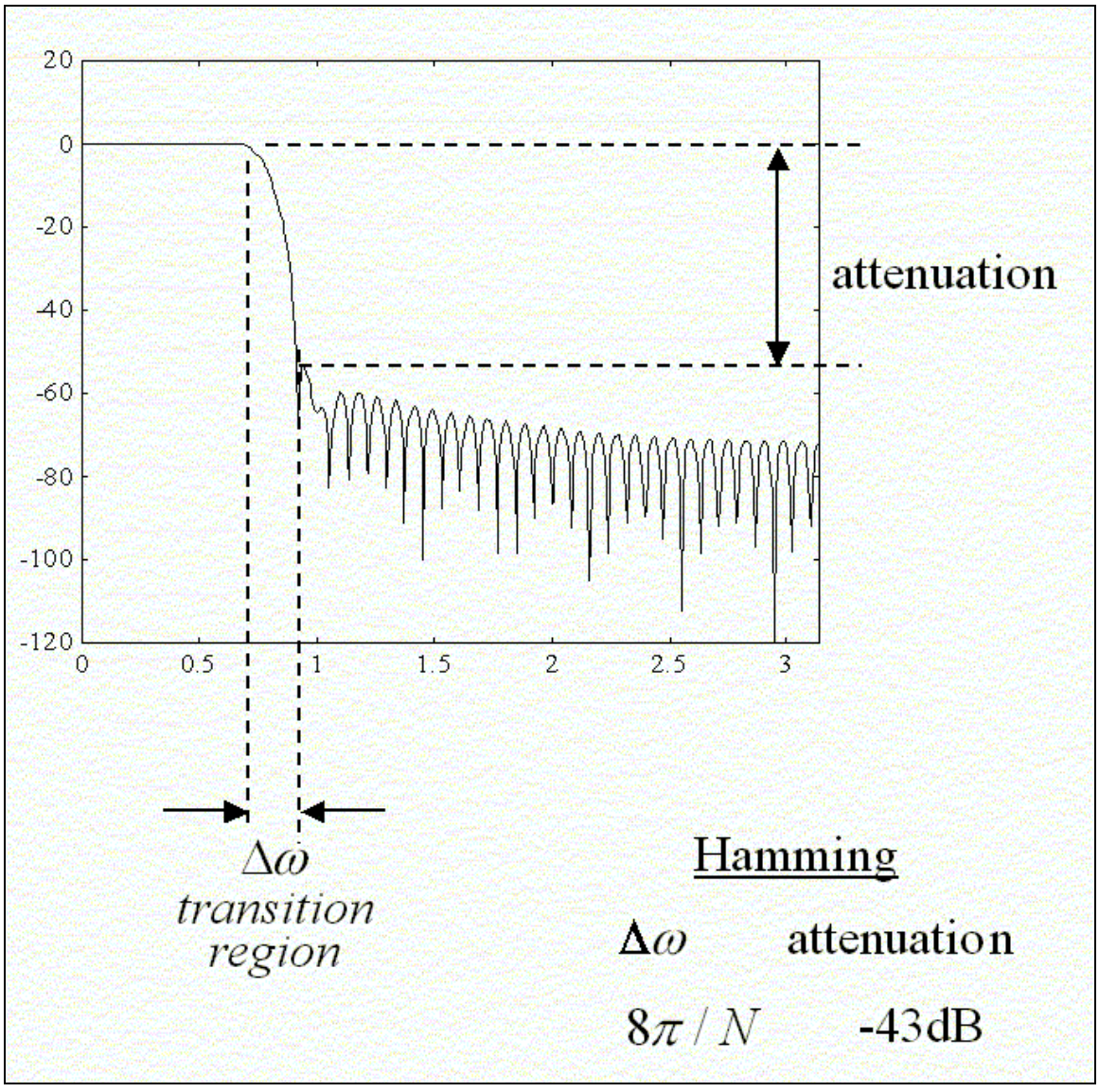


Figure 3.10. General FIR Filter (Hamming Window Design).

The frequency spectrum of the cosine modulating signal consists of dirac delta functions at the absolute value of the carrier frequency on both the real and the imaginary frequency axis. The modulation is expressed in discrete time as

$$y_{\text{mod}}(n) = m(n) \times y_f(n) \tag{3.28}$$

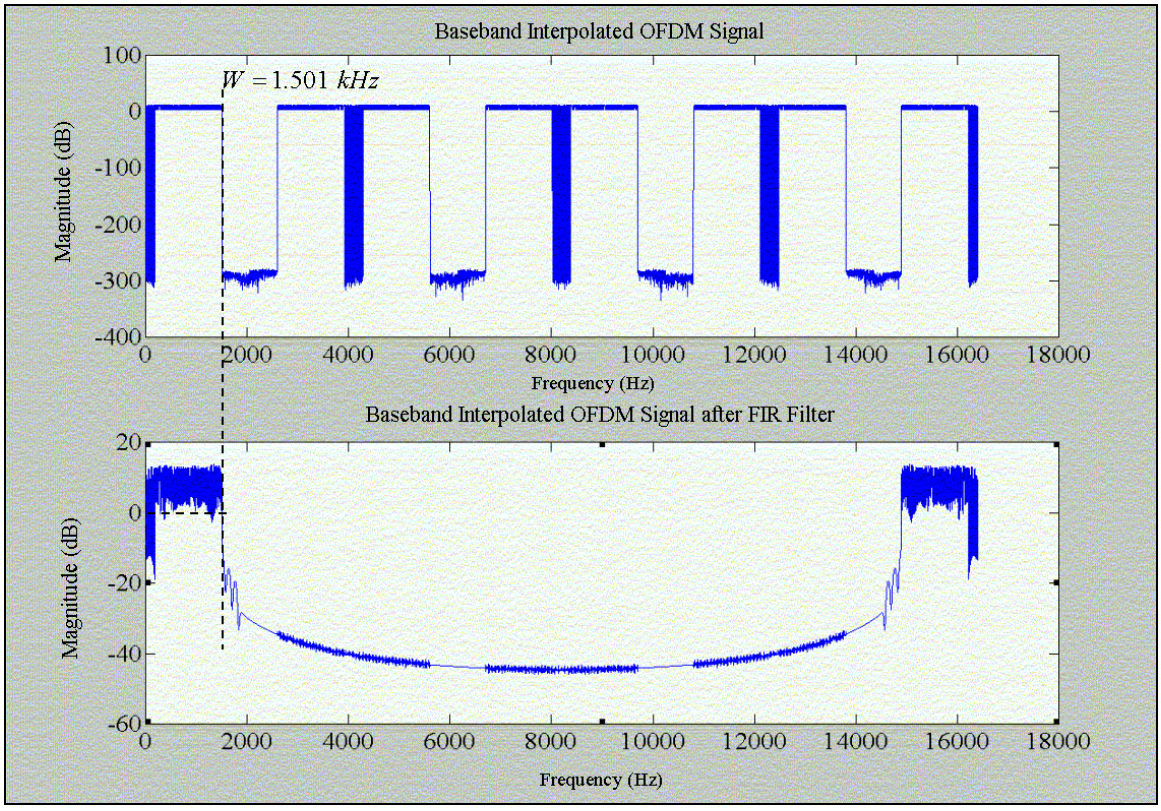


Figure 3.11. Interpolated Baseband OFDM Signal following FIR Filter.

The effect of modulation on y_f is shown in Fig. 3.13. The upper graph in Fig. 3.13 repeats the frequency spectrum of the baseband OFDM signal prior to interpolation. The center graph is the frequency spectrum of the modulating signal which illustrates the dirac delta functions at the carrier frequency. Finally the lower graph is the frequency spectrum of the modulated OFDM signal. This is the form of the signal that is transmitted through the channel. In practice the signal would be applied to a Digital to Analog converter (DAC) and an associated low pass filter (LPF). For the simulation these

components are assumed to be a part of the channel transfer function. Similarly, an Analog to Digital Converter (ADC) and associated LPF would be used in practice to return the signal to discrete time following transmission through the channel. These are also assumed to be included in the channel transfer function.

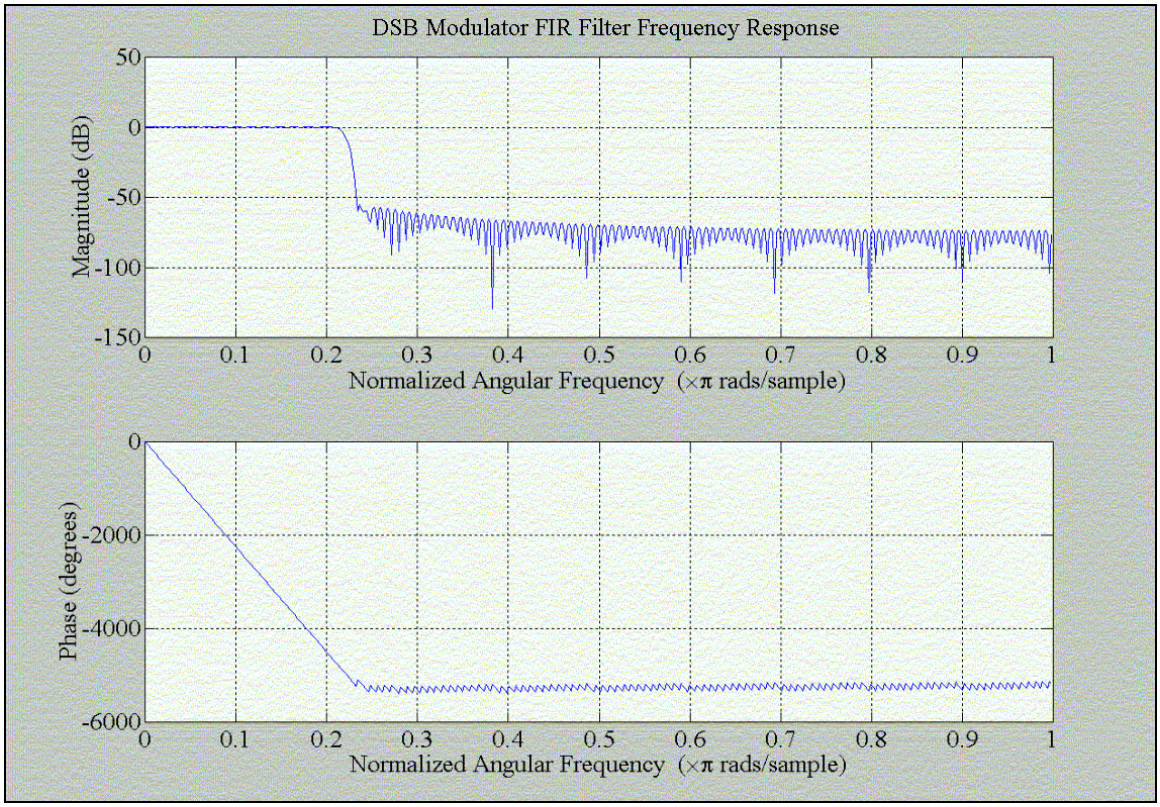


Figure 3.12. FIR Filter Frequency Response.

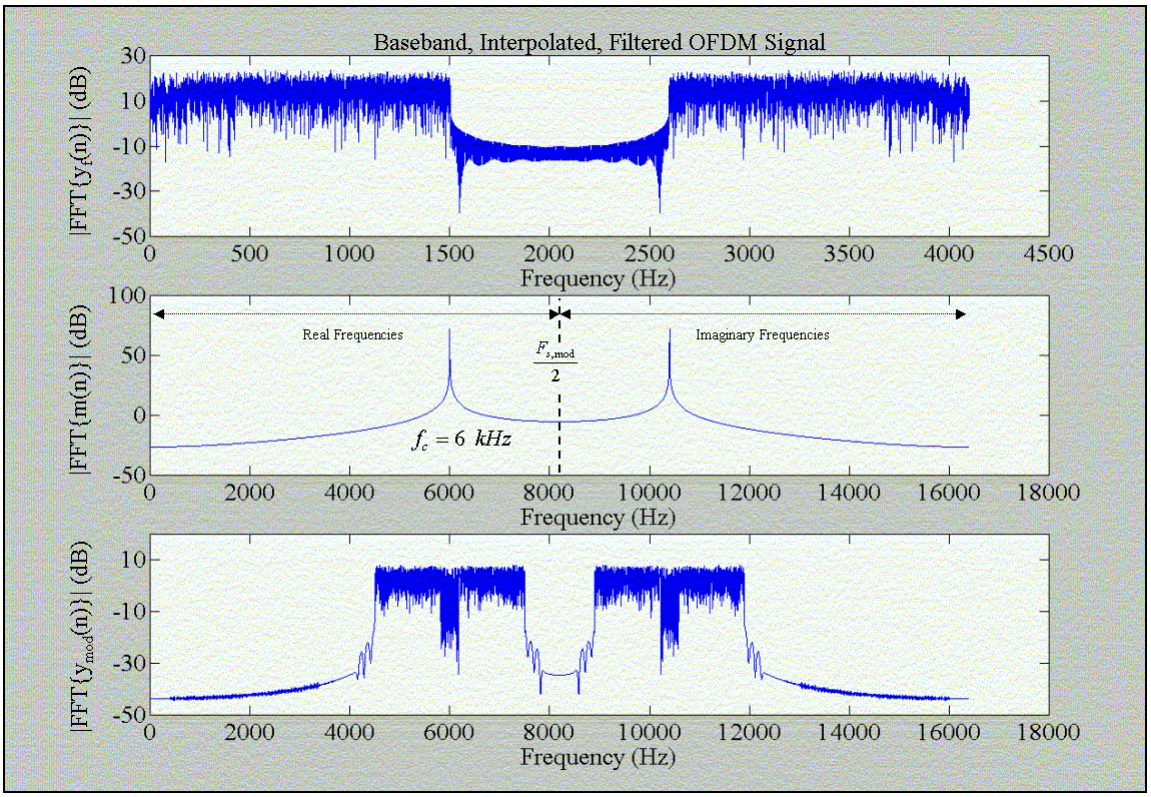


Figure 3.13. Modulation Signals.

The DSB Modulator completes the transmitter for the simulation. The next effect on the signal in practice would be the channel. However in the simulation the channel transfer function is not applied until after demodulation in the receiver for reasons that will be discussed later.

C. MMPE CHANNEL MODEL

MMPE is the Monterey-Miami Parabolic Equation model. It provides an approximate solution to the Helmholtz equation based on the efficient split-step Fourier (SSF) marching algorithm. The Helmholtz equation is derived from the solution of the linear, three-dimensional, lossless, homogeneous wave equation describing the propagation of small amplitude acoustic signals in an ideal fluid [Ref. 6]. The velocity potential is assumed to have a time-harmonic solution, or in other words, the acoustic field is generated by a continuous wave source. The value of a time harmonic field at any

point and time in space is a function of only one frequency. Time harmonic solutions are important because solutions to the wave equation with arbitrary time dependence can be found through application of Fourier transform techniques to the time harmonic solutions [Ref. 7].

The following development follows the work by Houdeshell. [Ref. 8] The linear wave equation is given in Eq. 3.29 with the complex acoustic pressure field given as Eq. 3.30,

$$\nabla^2 P + \frac{1}{c^2}\frac{\partial^2 P}{\partial t^2} = 0 \tag{3.29}$$

$$P\left(\vec{r},t\right) = p\left(\vec{r}\right)e^{-j\omega t} \tag{3.30}$$

where P is the complex acoustic pressure field and p is the amplitude of the acoustic pressure. The Helmholtz equation is then

$$\nabla^2 p\left(\vec{r}\right) - \frac{\omega^2}{c^2} p\left(\vec{r}\right) = 0 \tag{3.31}$$

A cylindrical coordinate system is used in the solution. Therefore the position vector, $\vec{r}$, is a function of range, r, depth, z, and the azimuthal angle, ϕ. The azimuthal dependence of the acoustic field is minor and ignored in the solution. Therefore p is a function of range and depth only, i.e. $p(r,z)$.

Let the complex acoustic pressure be defined as

$$P\left(r, z, \omega t\right) = p\left(r, z\right)e^{-j\omega t} \tag{3.32}$$

and

$$p\left(r,z\right) = \frac{1}{\sqrt{r}}u\left(r,z\right) \tag{3.33}$$

where the $1/\sqrt{r}$ term accounts for azimuthal spreading and *u(r,z)* defines the two dimensional acoustic pressure field. Substituting Eqs. 3.32 and 3.33 into the Helmholtz equation yields the uncoupled azimuthal approximation, which is

$$\boxed{\frac{\partial^2 u}{\partial r^2} + \frac{\partial^2 u}{\partial z^2} + k_o^2\left(n^2 + \frac{1}{4k_o r^2}\right)u = 0}. \tag{3.34}$$

The following operators, denoted by the subscript, *op*, are defined as

$$\boxed{P_{op} = \partial / \partial r} \tag{3.35}$$

and

$$\boxed{Q_{op} = \sqrt{\mu + \varepsilon + 1}} \tag{3.36}$$

where

$$\boxed{\varepsilon = n^2 - 1} \tag{3.37}$$

and

$$\boxed{\mu = \frac{1}{k_o^2}\frac{\partial^2}{\partial z^2}}. \tag{3.38}$$

Then Eq. 3.32 can be rewritten as

$$\boxed{\left(P_{op}^2 + k_o^2 Q_{op}^2\right)u = 0}. \tag{3.39}$$

If *u* is defined as

$$\boxed{u = Q_{op}^{-1/2}\Psi} \tag{3.40}$$

then the outgoing wave can be shown to satisfy [Ref. 9]

$$\boxed{-\,jk_o^{-1}\frac{\partial \Psi}{\partial r} = Q_{op}\,\Psi} \tag{3.41}$$

which is the fundamental equation of all PE models. The numeric solution algorithm is developed by breaking the acoustic field into a slowly modulating envelope or field function and a rapidly varying phase term. The field function, $\psi(r,z)$, is defined as

$$\boxed{\Psi = \psi e^{jk_o r}} \tag{3.42}$$

which can be related to the acoustic pressure as

$$\boxed{p(r,z) = P_{op}\sqrt{\frac{R_o}{r}}Q_{op}^{-1/2}\psi(r,z)e^{jk_o r}}. \tag{3.43}$$

The parabolic equation for the field function is

$$\boxed{\frac{\partial \psi}{\partial r} = -jk_o\psi + jk_o Q_{op}\psi = -jk_o H_{op}\psi} \tag{3.44}$$

where

$$\boxed{H_{op} = 1 - Q_{op}}. \tag{3.45}$$

The solution to the field function can then be marched outward in range according to

$$\boxed{\psi(r+\Delta r) = \Phi(r)\psi(r)} \tag{3.46}$$

where the propagator, $\Phi(r)$, which is a unitary operator, steps the solution out in range. The split-step Fourier method is used to apply the unitary operator. The operator, H_{op}, is separated by using the wide-angle approximation [Ref. 10]of the square root operation such that

$$H_{op} \approx T_{op} + U_{op} \tag{3.47}$$

where

$$U_{op} = -(n-1) \tag{3.48}$$

and

$$T_{op}(k) = 1 - \sqrt{1 + \frac{1}{2k_o^2}\frac{\partial^2}{\partial z^2}}. \tag{3.49}$$

Therefore the propagator may be expressed as [Ref. 11]

$$\Phi(r) = e^{-jk_o \frac{\Delta r}{2} U_{op}(r+\Delta r)} e^{-jk_o \Delta r T_{op}} e^{-jk_o \frac{\Delta r}{2} U_{op}(r)} \tag{3.50}$$

The PE/SSF method applies the operator

$$e^{-jk_o \Delta r \hat{T}_{op}} \tag{3.51}$$

in the k_z domain where the operator, $\hat{T}_{op}$, is defined as

$$\hat{T}_{op} = 1 - \sqrt{1 - \left(\frac{k_z}{k_o}\right)^2}. \tag{3.52}$$

The following convention is used for the FFT

$$\boxed{\begin{aligned}&\psi(z) = FFT\ \{\psi(k_z)\}\\ &and\\ &\psi(k_z) = IFFT\ \{\psi(z)\}\end{aligned}}. \tag{3.53}$$

Finally, the PE/SSF method is implemented as

$$\boxed{\psi(z,r+\Delta r)=e^{-jk_o\frac{\Delta r}{2}U_{op}(z,r+\Delta r)}FFT\left\{e^{-jk_o\Delta r\hat{T}_{op}(k_z)}IFFT\left\{e^{-jk_o\frac{\Delta r}{2}U_{op}(z,r)}\psi(z,r)\right\}\right\}} \tag{3.54}$$

The MMPE model provides multiple outputs to the user. The primary output for use with the OFDM simulation is the complex acoustic pressure. The MMPE model used in the simulation is a 2D model in range and depth since the azimuthal dependence was neglected thereby omitting the third dimension. The solution is a grid of points from which the complex acoustic pressure can be evaluated. The complex acoustic pressure output for a given point in the grid is a function of frequency. The application of this data to the OFDM signal is discussed in the receiver section covering the application of the acoustic channel.

The source depth for the shallow water (100 m) acoustic channels was 30 m and 100 m for the deep water (340 m) acoustic channels. The receiver depth varied with the results of the acoustic channel evaluation. The receiver depth was chosen by determining the depth with the greatest average acoustic pressure amplitude across the frequency band evaluated. The sound speed profile (SSP) used for all acoustic channels is from the Atlantic Ocean just off the eastern coast and is shown in Figure 3.14.

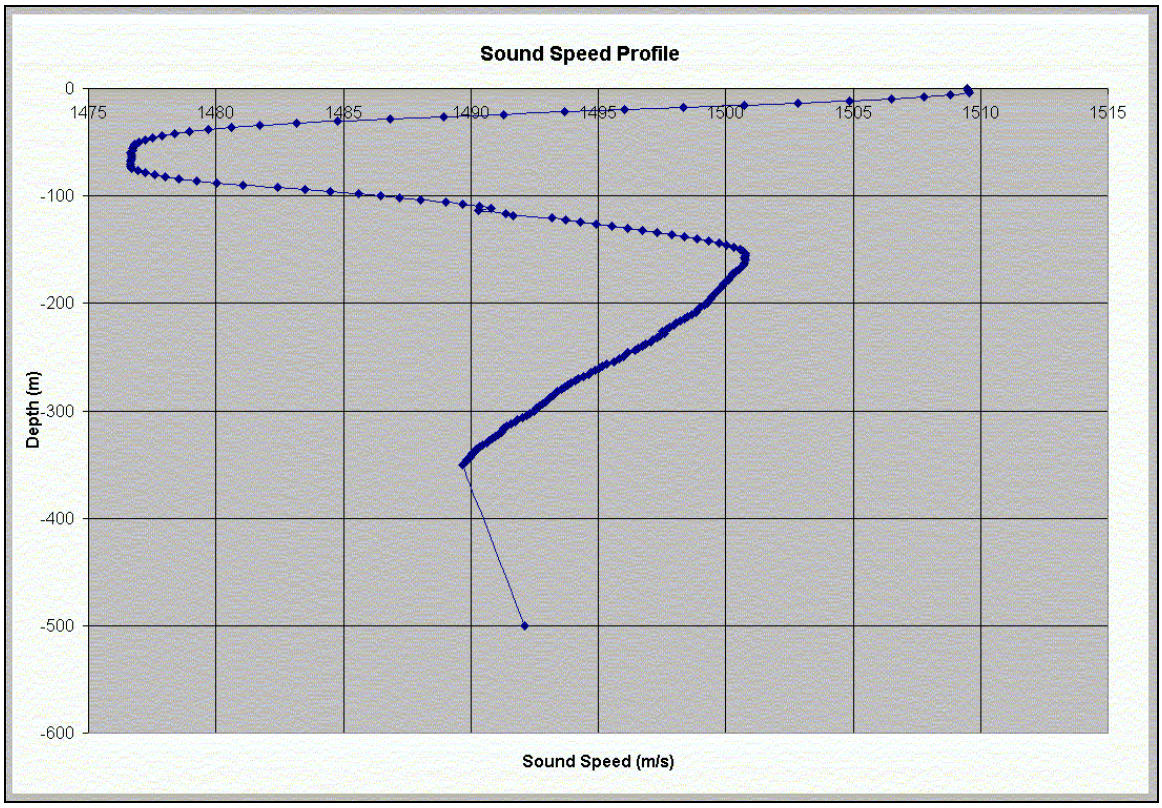


Figure 3.14. Sound Speed Profile.

D. RECEIVER

The functional blocks in the receiver were presented in Fig. 3.1. For the most part the blocks of the receiver simply undo the work of the transmitter to convert the information back to its original form. Additional blocks in the receiver include the channel estimation and equalization blocks, additive white gaussian noise (AWGN), synchronization and bit error analysis.

1. AWGN Addition

The simulation assumes the background noise present in the channel is AWGN. The signal power is calculated as

$$P_s = \frac{\sum_{i=1}^{K} x^2(i)}{K} \tag{3.55}$$

where K is the length of the received signal. The noise power to be added is then calculated as,

$$P_n = \frac{P_s}{10^{\frac{SNR}{10}}} \tag{3.56}$$

where SNR is the desired SNR in dB. The received signal is then

$$x_r(n) = x(n) + w(n) \tag{3.57}$$

where *x(n)* is the noise free signal at the input to the receiver, which is the output of the transmitter and *w(n)* is AWGN.

2. DSB Demodulator

The received signal is first demodulated using a cosine signal, as presented in Eq. 3.26, then the demodulated signal is filtered and decimated. The LPF is designed such that spectral components above the known carrier frequency plus the bandwidth of the transmitted signal are removed. Then decimation by *D* is performed to reduce the number of samples and the sampling rate.

The reader should note that in the current flowchart of the simulation, some processing of the received signal occurs before the acoustic channel model is applied. This is because without source or receiver motion the MMPE model can be applied directly to the complex QAM symbols in each of the OFDM blocks. In reality the channel effects would alter the signal after transmission and before reception. In future work source and receiver motion will be modeled. At that point the application of the

MMPE model will take place between transmission and reception. Therefore the blocks such as the DSB Modulator and Demodulator have been left in the simulation even though they are not essential in the current form.

Signal detection theory is not addressed in this thesis. As such it is assumed that the signal is detected by some means and the stream of data containing the signal is processed. The simulation is designed such that the received signal of finite length contains the transmitted signal of some length less than the length of the received signal. The information is assumed to be received in an AWGN environment. The DSB Demodulator demodulates the entire received signal to baseband. The synchronizer is responsible for identifying where the information lies within the received signal. Only the portion of the received signal containing information, i.e. the transmitted signal, is processed in the remainder of the receiver.

The demodulating signal is expressed as

$$d(n) = \cos\left(\frac{2\pi f_c n}{F_{s,\mathrm{mod}}}\right) \tag{3.58}$$

where n is defined from 1 to the length of the received signal, $x_r(n)$. The demodulated signal is then the product of the received signal with the demodulating signal, that is,

$$x_{r,d}(n) = x_r(n) \times d(n). \tag{3.59}$$

The signal is baseband and real at this point. Next the signal must be decimated to reduce the number of samples and the sampling rate by the decimation factor, D. First the signal must be passed through an antialiasing LPF to remove spectral components above π/D. The LPF is again designed using the Hamming window method with the following parameters:

$$\omega_p = \frac{1.1(2\pi W)}{F_{s.\,\mathrm{mod}}} \equiv \text{pass band digital frequency}$$

$$\omega_s = \frac{1.25(2\pi W)}{F_{s,\mathrm{mod}}} \equiv \text{stop band digital frequency}$$

$$N_h = \frac{8\pi}{\omega_s - \omega_p} \equiv \text{filter order (number of coefficients is } N_h + 1)$$

where reference can be made to Fig. 3.10 for the general filter characteristics. The output of the filter is

$$x_{r,f}(n) = h(n) * x_{r,d}(n) \tag{3.60}$$

where *h(n)* is the impulse response of the LPF and * represents linear convolution. The final step in the DSB Demodulator is decimation of the signal. Decimation by a factor *D* results in the expression

$$x_{r,fd}(m) = x_{r,f}(mD). \tag{3.61}$$

The effects of the DSB Demodulator are illustrated in Fig. 3.15. The signal parameters of Fig. 3.15 are the same as those of Fig. 3.9. The bit rate is 5 kbps using 16 QAM and the decimation factor is *D* is 4. The baseband bandwidth is 1.501 kHz.

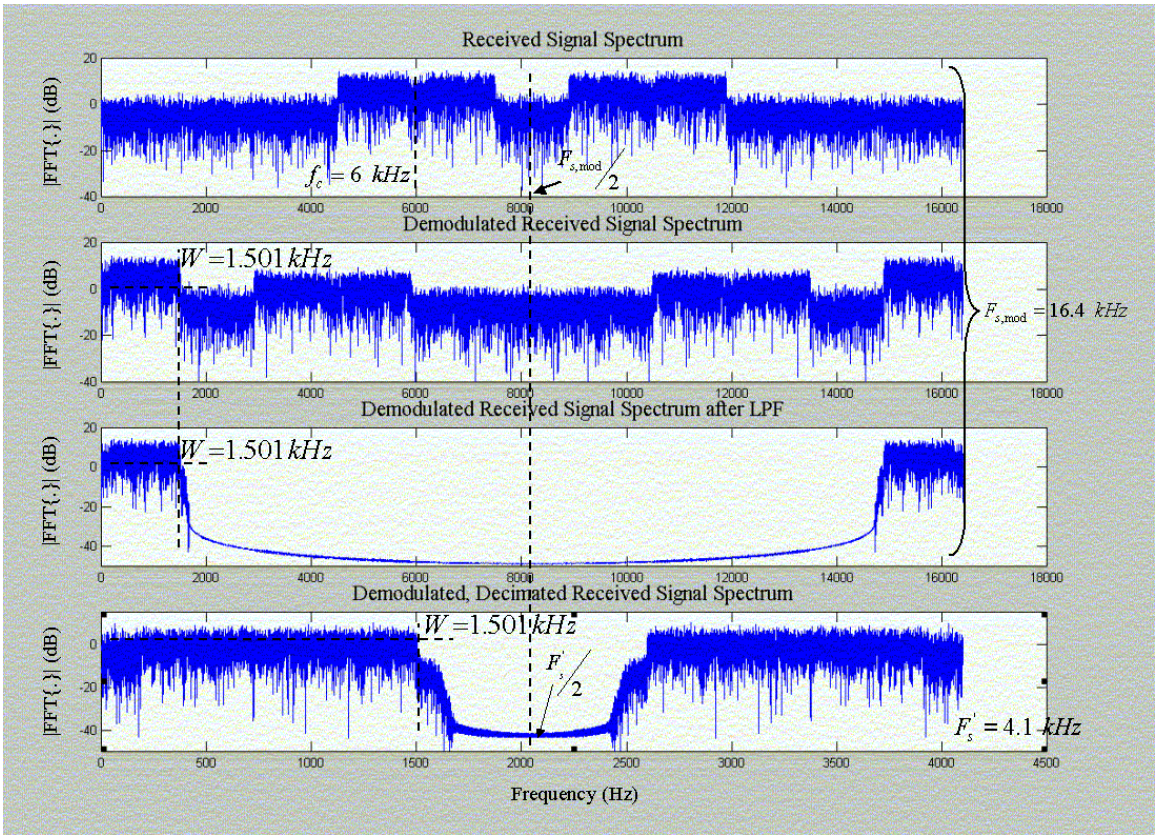


Figure 3.15. DSB Demodulator Signal Processing Effects.

3. Synchronizer

In order to demodulate the subcarriers an OFDM receiver must first perform synchronization. There are two tasks related to synchronization. The first is determining the location of the OFDM symbol blocks. The second is estimating and correcting for frequency offset of the subcarriers. In addition coherent receivers also require the carrier phase of the DSB Modulator to be synchronized. In the simulation the phase of the DSB Modulator is assumed to be synchronized by the OFDM receiver.

Recall from the discussion on OFDM theory that the subcarriers will remain orthogonal in the receiver only if the transmitter and receiver use exactly the same frequencies. If not then ICI will occur. Practical oscillators possess phase noise, which results in varying subcarrier center frequencies since the frequency is the time derivative of the phase. In practical systems with non-ideal oscillators this is an unavoidable source of ICI in OFDM systems. However Van Nee and Prasad [Ref. 4] indicate that this source

can be minimized to acceptable levels through a well-designed system. In the simulation phase noise is not modeled and therefore assumed negligible. The sensitivity of OFDM systems to phase noise and frequency offset is one of the major disadvantages of OFDM relative to single carrier systems. In single carrier systems phase noise and frequency offset degrade SNR but do not produce interference [Ref. 4]. Van Nee and Prasad [Ref. 4] provide a detailed discussion of these two effects in OFDM systems.

Frequency offset is caused not only by phase noise of local oscillators in the transmitter but also by Doppler shift due to non-zero relative motion between the source and the receiver. The simulation assumes the transmitter and receiver are stationary. Therefore frequency offsets are not addressed in this thesis.

While OFDM systems are relatively sensitive to frequency issues they are quite robust to timing errors. Symbol timing may vary by as much as the duration of the guard interval, t_g, without producing ISI or ICI. However an optimal timing instant exists which provides for maximum multipath robustness. Any deviation from the optimal timing instant reduces the delay spread tolerance the system was designed to handle. OFDM systems attempt to minimize the timing error relative to the guard interval.

The cyclic extension is used to perform symbol synchronization in the simulation. Recall that the first t_g samples received will be the same as the last t_g samples for each OFDM block due to the cyclic extension. Therefore correlation streams of data t_g samples long with streams of the same length but delayed by the duration of the FFT interval will identify the location of the OFDM blocks.

The synchronization block diagram is presented in [Ref. 4] and modified slightly for the purposes of the simulation to the form shown in Fig. 3.16.

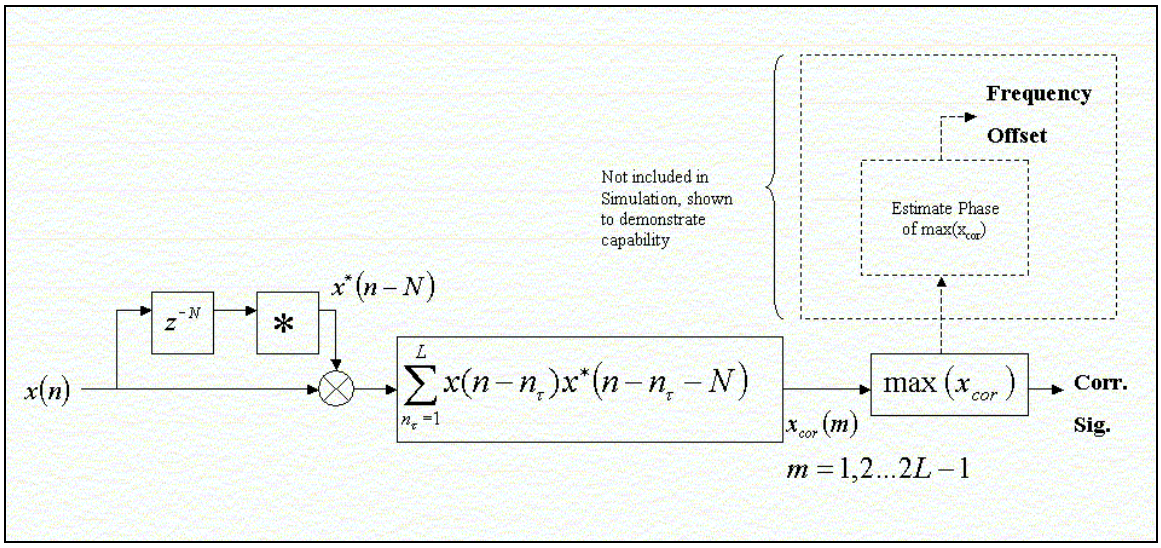


Figure 3.16. Synchronization Flowchart [after Ref. 4].

Recall that t_g is the guard time in continuous time and L is the discrete time number of samples corresponding to t_g for the applicable sampling rate. Figure 3.17 illustrates the structure of an OFDM signal in serial form to clarify the cross correlation used in the synchronizer. Fig. 3.17 identifies the observation intervals of length $N+L$ which slide over the received signal in discrete time. The first and last L samples in the observation interval will be the same when the interval falls exactly across an OFDM symbol labeled as frames in Fig. 3.17. At this point the cross correlation terms will be the greatest. When the observation interval includes samples from two different frames such that the first and last L samples are independent the cross correlation function will have a peak much lower than when the interval is exactly over one frame. In fact in the limit as the number of samples over which the cross correlation is performed is very large the ratio of the sidelobe-to-peak amplitude will go to zero [Ref. 4]. Figure 3.18 provides a realization of the correlation signal identified in Fig. 3.16 for an OFDM signal with six OFDM blocks where the first two blocks are the channel estimation blocks.

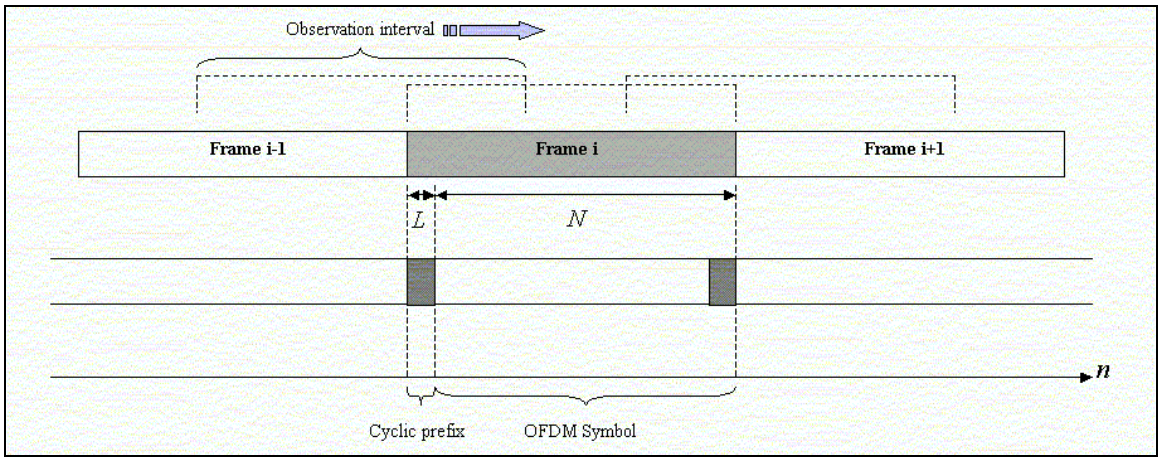


Figure 3.17. OFDM Signal Structure with Synchronizer Observation Intervals.

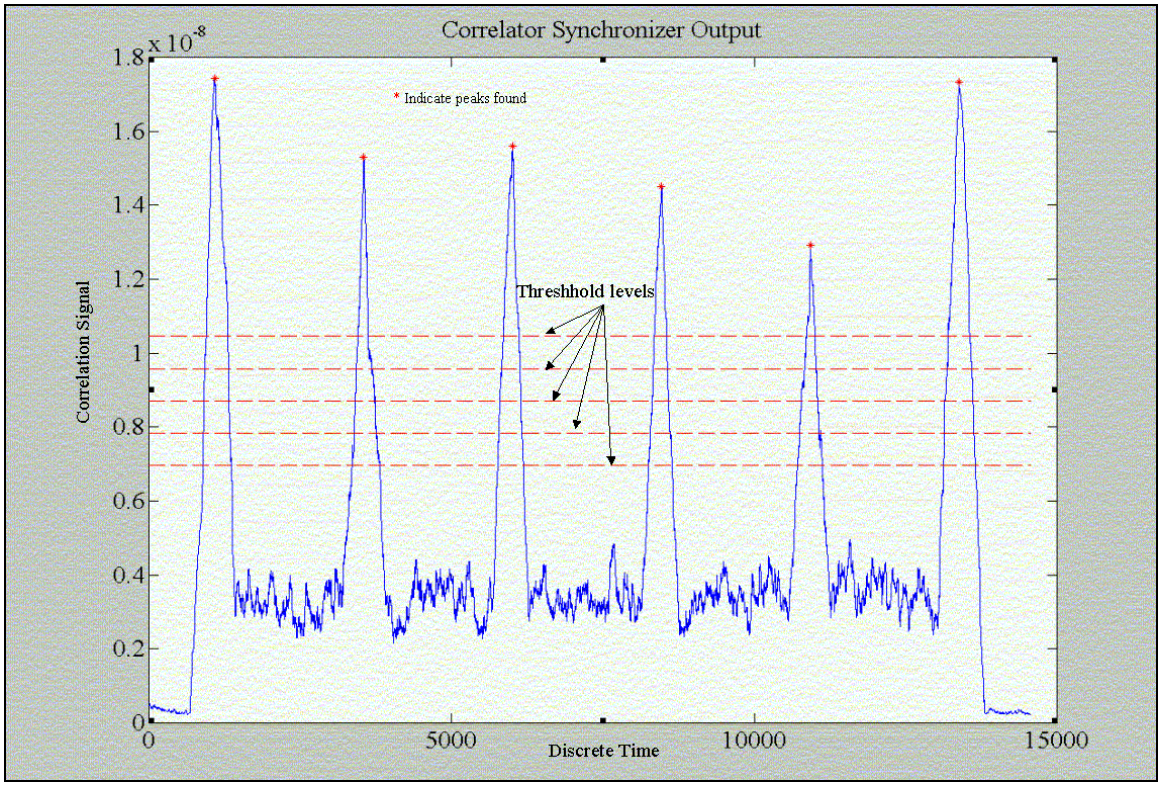


Figure 3.18. Correlation Signal Realization, *SNR* = 17.413 dB, *N*=2048, *R* = 5 kbps.

The peak finding algorithm uses five threshold levels to determine the existence of peaks in the correlation signal. The threshold levels are adjustable and based on the maximum value found in the correlation signal. In the realization of Fig. 3.18 the

threshold levels are 0.4, 0.45, 0.50, 0.55 and 0.60 of the maximum of the correlation signal. The algorithm requires that for a value to qualify as a peak it must first define an inflection point in the correlation signal. Second the value must register as a peak at all threshold levels. The impulse response and frequency response of the acoustic channel model applied to the transmitted signal in realizing the correlation signal in Fig. 3.18 is shown in Fig. 3.19. The impulse response of Fig. 3.19 clearly indicates the existence of three distinct receptions due to the multipath environment. Also note that the frequency response of Fig. 3.19 illustrates frequency selective fading and nearly linear phase response.

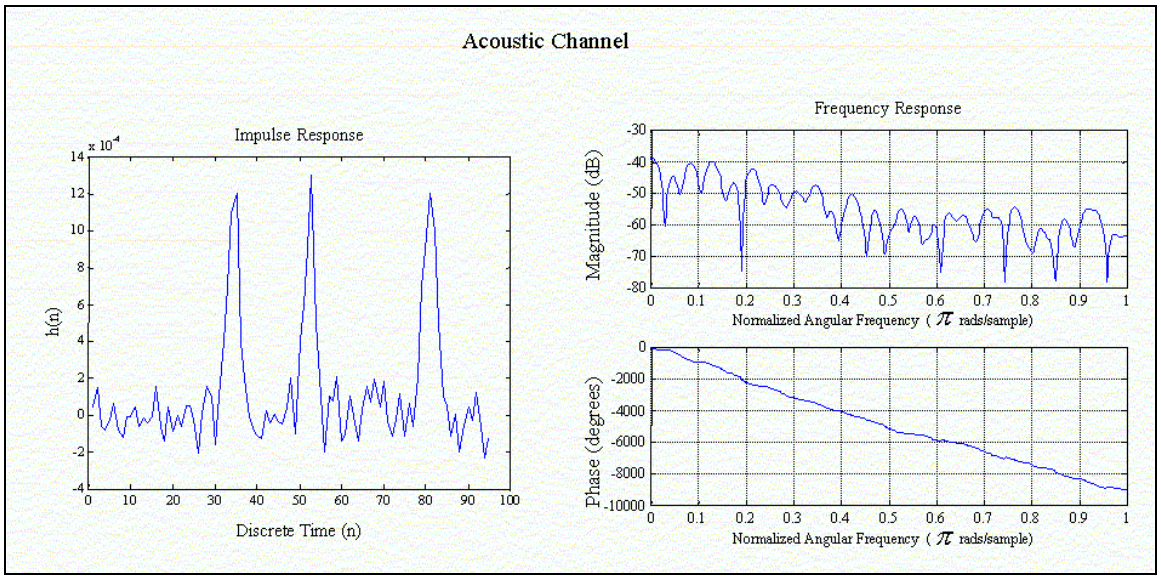


Figure 3.19. Acoustic Channel Filter Plots for Realization of Fig. 3.17.

Van Nee and Prasad [Ref. 4] discuss optimal timing by analyzing the correlation signal power. The idea is to take the samples that lie within the OFDM symbol that also have the greatest power and therefore the greatest SNR. In the simulation the signal power is not analyzed since the correlator synchronizer, while computationally intensive, was very successful at identifying the boundaries of the OFDM symbols. In most cases the correct symbol boundaries were found to within a few percent error relative to the length of the cyclic prefix, *L*. Future work might look at the benefits of using the optimal timing instant. Additionally, the correlator synchronizer used here is computationally intensive and a less demanding algorithm would be desirable for practical application. In

that case the optimal timing instant would be of greater value as a less intense timing synchronization method would likely be less accurate. A low complexity synchronization method that seems to be of value in OFDM is presented by van de Beek et al.[Ref. 12]. The synchronization is performed by the Matlab functions peakfinder.m and synchronizer.m contained in Appendix C.

4. Composite Description of Serial to Parallel Conversion through FFT/OFDM Demodulation

The serial to parallel (S/P) converter takes the reduced length signal from the synchronizer and converts it to parallel form. The resulting matrix of samples will have the number of rows corresponding to the block length determined by the synchronizer and the number of columns will be the number of OFDM symbols found by the synchronizer. Recall that each peak in the correlation signal corresponds to the start of an OFDM symbol. The matrix of real time samples from the S/P converter contain the cyclic prefix which is removed from each OFDM symbol, i.e. column in the matrix, by the cyclic extension removal block.

The OFDM Demodulator simply performs an N point FFT of the matrix from the cyclic extension removal block. The FFT can be expressed as

$$\boxed{X(k) = \sum_{1}^{N} x(n) W_N^{kn} \qquad for\ \mathrm{k} = 1,2 \ldots N} \quad . \tag{3.62}$$

The FFT can also be expressed in matrix form as

$$\boxed{\underline{X}_N = \underline{W}_N \underline{x}_N} \tag{3.63}$$

where $\underline{W}_N \in C^{N \times N}$ and $\underline{x}_N \in R^{N \times M}$ therefore $\underline{X}_N \in C^{N \times M}$. The resulting matrix of complex samples represent the transmitted complex QAM symbols. The OFDM symbols making up each column of the matrix are conjugate symmetric and only the active subcarriers are of interest for receiving the information in the signal.

5. Acoustic Channel Application

The output of the MMPE model used in the simulation is in the form of a complex transfer function. The transfer function is applied directly to the complex QAM symbols present after the OFDM demodulator. The generic output from the MMPE model is a grid of points in depth and range. The desired output is chosen for a given point. The complex transfer function represents the complex acoustic pressure as a function of frequency present at the chosen point in the grid.

The transfer function contains values for the frequencies between f_c and $f_c + \overline{N}\Delta f_{sc}$. This band represents the frequencies that would lie in the real frequency region of the baseband signal corresponding to the signal modulated to f_c. Recall that only the active subcarriers carry non-zero signal power and therefore determine the bandwidth of the transmitted signal. In order to apply the transfer function to the complex QAM symbols it must be made conjugate symmetric by application of Eq. 3.12. The complex QAM symbols after application of the transfer function is

$$\boxed{\underline{Y} = \underline{X} .* \underline{H}} \quad . \tag{3.64}$$

where $\underline{X} \in C^{N\times(N_{block}+2)}$ and $\underline{H} \in C^{N\times 1}$ therefore $\underline{Y} \in C^{N\times(N_{block}+2)}$. The variable $\underline{X}$ represents the matrix of complex QAM symbols after the OFDM demodulator, $\underline{H}$ represents the complex transfer function and $\underline{Y}$ is the output. The .* in Eq. 3.64 indicates array multiplication such that each column of $\underline{X}$ is weighted by the vector $\underline{H}$. The number of columns in $\underline{X}$ and $\underline{Y}$ is $N_{block} + 2$ due to the 2 blocks of complex QAM symbols used for channel estimation and equalization.

6. Channel Equalization

The channel equalization scheme simply attempts to invert the effects of the channel such that the complex QAM symbols after equalization will be the same as those

that were generated in the transmitter. The first two OFDM symbols are used to estimate the channel transfer function. The generation of the channel estimation blocks was discussed in the QAM Modulator section. The channel estimate is

$$H_{est} = \frac{X_r}{X} \tag{3.65}$$

where X_r represents the received complex QAM symbols in the channel estimation blocks, X is the complex QAM symbols present in the QAM Modulator of the transmitter and H_{est} is an N by 2 matrix of complex values representing estimates of the magnitude and phase of the channel transfer function at frequencies corresponding to the subcarrier center frequencies. Array division is implied in Eq. 3.65. The values of X are assumed to be known at the receiver. Each row of H_{est} is averaged to give a vector of values representing the estimate of the channel transfer function. That is,

$$H'_{est}(r) = \frac{H_{est}(r,1) + H_{est}(r,2)}{2} \qquad for\ r = 1,2...N \tag{3.66}$$

In the simulation the estimates of the channel transfer function are performed over the active subcarriers only rather than over all N as using the zero subcarriers would result in division by zero in H_{est}.

Finally the complex QAM symbols are equalized by defining

$$X_{eq} = \frac{X_r}{H'_{est}} \tag{3.67}$$

where the division is array division such that the columns of X_r are weighted by the inverse of H'_{est}.

A realization of the channel estimates is shown in Fig. 3.20. The magnitude of the channel estimate for the two channel estimation blocks and the mean of the two estimates is plotted versus the FFT bin number which is analogous with subcarrier number or

frequency. The plot is discontinuous at high and low bin numbers due to division by zero that would result where the zero symbols are interspersed to evenly shape the power distribution as discussed in the QAM Modulator section. The plots are over the active subcarriers only. Note that H_{mean} in Fig. 3.20 refers to the mean of the two channel estimates $H_{sys}(1)$ and $H_{sys}(2)$.

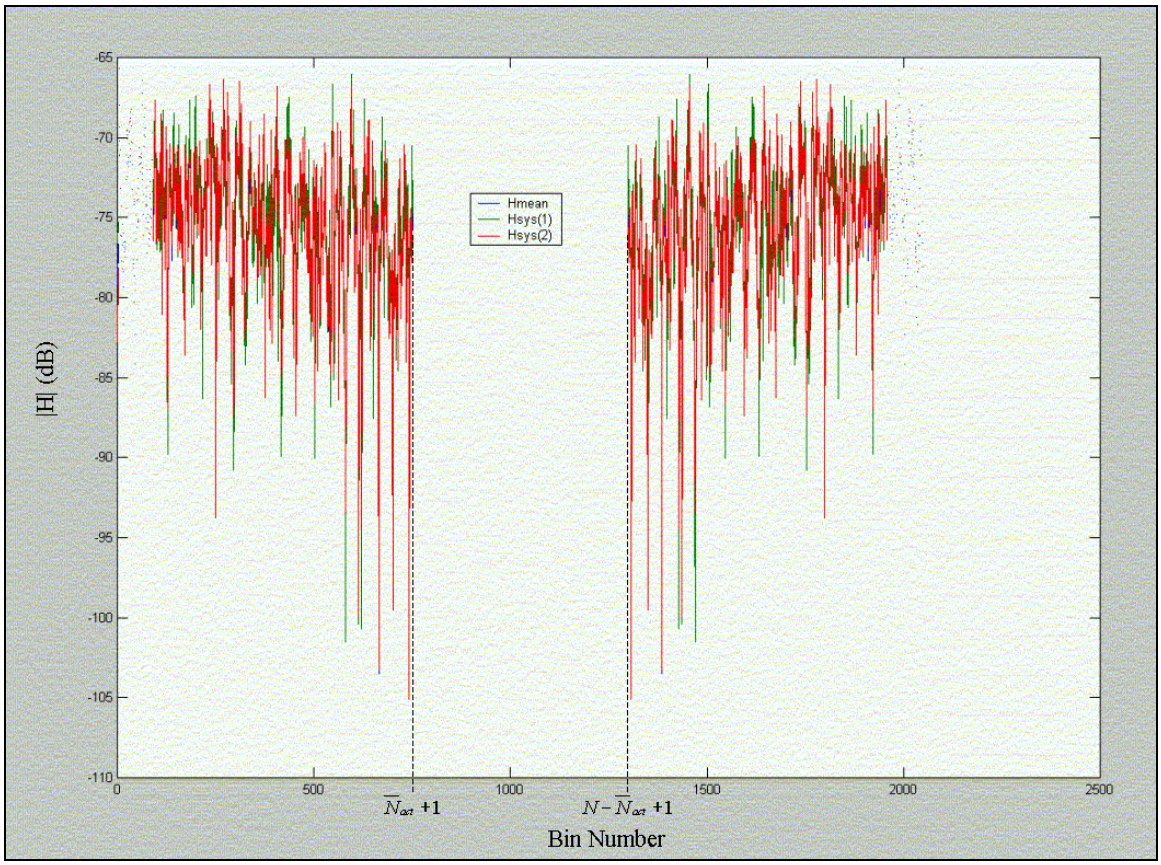


Figure 3.20. Channel Estimates, N=2048, 16 QAM, 5 kbps, f_c = 6 kHz, $\overline{N}_{act} = 750$.

7. QAM Demodulator

The QAM demodulator transforms the received complex QAM symbols to points on a Q-ary QAM constellation. Therefore the output will be integers from 0 to Q-1.

The first step is to strip the first $\overline{N}_{act}+1$ symbols from each block as these are the symbols that carry the transmitted information. This implies that the receiver knows the number of active subcarriers. This information is assumed to be contained in protocol

information that would be carried as overhead in the transmission system. Recall that the zero symbols that were added in the QAM modulator are for data management purposes and could just as well be protocol information. Protocol is not addressed in this thesis but is assumed to be present in making some assumptions that are required in the receiver.

The zero symbols are removed and the magnitude and phase of each symbol is determined and converted to Cartesian coordinates. The real and imaginary components of the received symbols correspond to the in-phase and Quadrature components respectively. The operation of the QAM Demodulator up to this point is illustrated in Fig. 3.21.

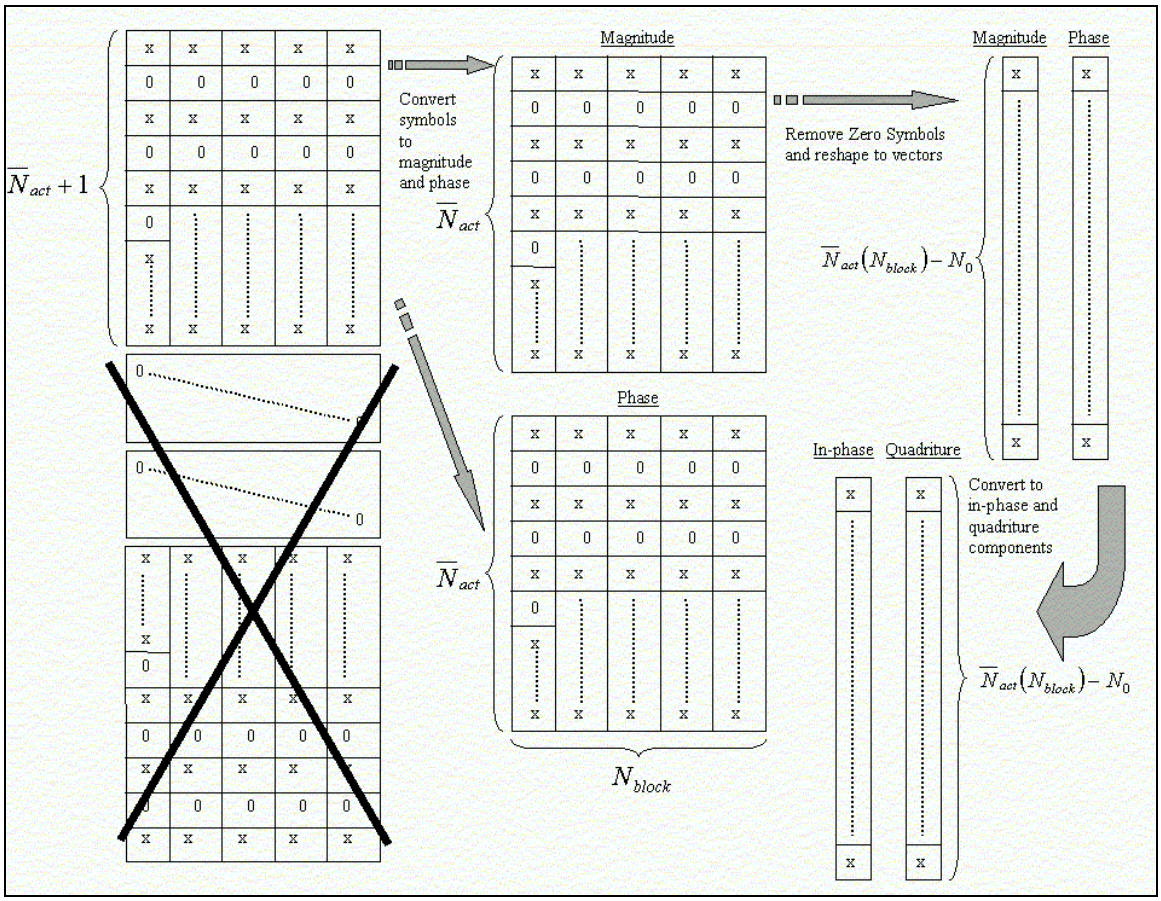


Figure 3.21. QAM Demodulator.

The next step is to map the in-phase and Quadrature components to Q-ary QAM constellation points. The mapping of constellation points to in-phase and Quadrature components was explained in the QAM modulator. In that case the constellation point

was exact and resulted in integer values for the in-phase and Quadrature components. However in mapping back to QAM constellation points the in-phase and Quadrature components will not in general be integer values. Therefore an algorithm is required to determine the constellation point to which the received symbol most closely corresponds.

This operation is performed by the Matlab function qaskdecomod.m contained in Appendix C. A 2 by Q matrix containing the in-phase and Quadrature values corresponding to each point in a Q-ary QAM constellation is created. The in-phase and Quadrature components of the received symbols are compared to the vectors in the constellation matrix to determine the constellation point that is nearest to the received symbol. The received symbol is mapped to this constellation point by its corresponding integer value. The process is illustrated in Fig. 3.22. The example in Fig. 3.22 is for a constellation size of 4 for simplicity although 4-ary QAM constellations are not used for data transmission in the simulation. The error analysis is performed for each received symbol resulting in a mapping of an integer from 0 to Q-1 for each in-phase and Quadrature component pair. Therefore the output of the QAM Demodulator is a vector of integer values of length $\overline{N}_{act} * N_{block} - N_0$.

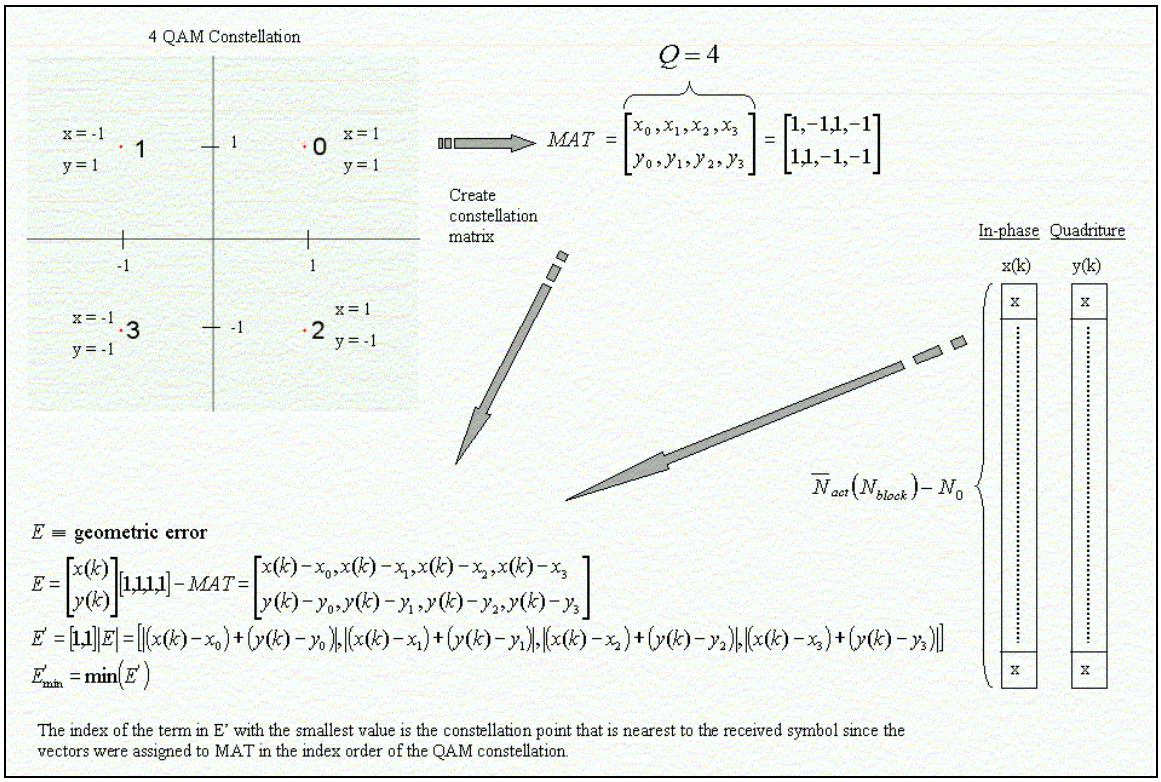


Figure 3.22. Received Symbol QAM Decoding Process.

8. Composite Description of Block De-Interleaving through Error Analysis

The Block De-interleaver simply undoes the interleaving of data performed by the Block Interleaver in the transmitter. It is assumed that the receiver knows the code word length of the Reed Solomon code as the code vector supplied to the deinterleaver is reshaped into a matrix with *n* columns in preparation for Reed Solomon decoding.

The Reed Solomon decoder applies an algorithm in the Matlab communications toolbox, rsdeco.m, in decoding. Recall the code word length is *n* and the message length is *k*. Therefore the input matrix has *n* columns and the output matrix has *k* columns where each row corresponds to a codeword when coded and a message when decoded.

The integers representing QAM symbols must be converted to binary to continue the signal reconstruction. Each integer is converted to a binary word with *q* bits where *q*

is the number of bits per QAM symbol. The output of this block is a vector of binary digits.

The quantization buffer simply reshapes the bit stream to a matrix with eight columns, since 8-bit quantization was performed in the transmitter. As stated earlier the level of quantization is adjustable and determined by the variable *bits* in the transmitter. In general the bit stream is reshaped to a *bits* column matrix in preparation for signal reconstruction.

Signal reconstruction consists of two steps. The first is the conversion of the binary words of length *bits* to integers. Second the integer values are mapped to the corresponding value of the information signal in the transmitter. Refer back to Fig. 3.4 which is the illustration of the quantization process.

The BER is determined by comparing the transmitted and received bit streams. An error occurs when the received bit is different than the corresponding transmitted bit.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. RESULTS

The results of the thesis are divided into 2 major sections. The first section develops the theoretical performance of an OFDM system. The second presents the results of the OFDM simulation for several combinations of system and acoustic channels parameters.

A. SYSTEM THEORETICAL PERFORMANCE

The theoretical performance of the system is first analyzed by considering a single carrier system. An OFDM system is equivalent to a set of independent and ISI free single carrier QAM systems. Therefore a large part of the analysis of an OFDM system can be performed by analyzing a single carrier QAM system. The single carrier analysis is extended to the multicarrier, OFDM, system to complete the analysis.

1. Single Carrier Analysis

The energy per two-dimensional symbol in a square QAM constellation is given by

$$\boxed{\varepsilon = \frac{Q-1}{6} d^2} \qquad (4.1)$$

where d is the distance between points in the QAM constellation and Q is the size of the QAM constellation. All points in the constellation are equally likely and the constellation is centered at the origin and therefore has zero mean. Equation 4.1 is exact only for square QAM constellations but is still a good approximation for the average transmit energy for non-square QAM constellations [Ref. 1]. For an ISI free channel with gain, $|H|$, the probability of two dimensional symbol error is approximately

$$\boxed{P_e \leq 4 Q_{erf}\left(\frac{d_{\min}}{2\sigma}\right)} \qquad (4.2)$$

where d_{min} is the minimum distance between QAM constellation points at the output of the channel and $Q_{erf(x)}$ is the error function, each of which are defined as[Ref. 3]

$$d_{\min} = d^2 \left| H \right|^2 \tag{4.3}$$

and

$$Q_{erf}(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-\frac{u^2}{2}} du \tag{4.4}$$

The probability of symbol error per dimension, which is $P_e/2$, for communication systems should be less than 10^{-4}. Therefore the following requirement must be met,

$$\left(\frac{d_{\min}}{2\sigma} \right)^2 = 11.8(dB) + \gamma_m(dB) - \gamma_c(dB) \tag{4.5}$$

The variable γ_m is the margin, which provides for additional performance in order to overcome unpredictable channel degradations and γ_c is the coding gain. Note that when $\gamma_m = \gamma_c = 0$ the system is uncoded and has no margin. Therefore 11.8 dB is the argument required in the Q(.) function to get $P_e/2 = 10^{-4}$. Coding increases d_{min} and therefore reduces the 11.8 dB by the amount of coding gain. Using a margin increases the 11.8 dB required by the amount of margin desired. [Ref. 3]

Equation 4.1 can be rewritten as

$$Q = 1 + \frac{6\varepsilon \left| H \right|^2}{d_{\min}^2} \tag{4.6}$$

which leads to the definition of SNR gap, or normalized SNR, Γ,

$$3\Gamma = \frac{d_{\min}^{2}}{4\sigma^{2}} \tag{4.7}$$

where σ^2 is the variance or power of the AWGN.

Therefore, for the target of $P_e/2 = 10^{-4}$, the following expression can be written for the SNR gap,

$$3\Gamma(dB) = 11.8 + \gamma_m - \gamma_c \ (dB) \tag{4.8}$$

Then by taking the base 2 log of Eq. 4.6 and substituting into Equation 4.7 for d_{min}, the number of bits that can be carried by QAM at the target $P_e/2$ is calculated as

$$b_{P_e} = \log_2(Q) = \log_2\left(1 + \frac{SNR}{\Gamma}\right) \tag{4.9}$$

where the channel output SNR is defined as

$$SNR = \frac{\varepsilon |H|^2}{2\sigma^2} \tag{4.10}$$

Note that Eq. 4.9 is the Shannon-Hartley theoretical channel capacity with the SNR reduced by a factor of the SNR gap. As the SNR gap goes to 1 (0 dB) the achievable data rate of the system approaches capacity. As can be seen, the SNR gap is a measure of the degradation of the system from optimal. The number of bits that can be carried by a QAM system as computed using Eq. 4.9 will not be an integer but is typically rounded down to the nearest integer. If a square QAM constellation is desired then it is rounded down to the nearest even integer. The SNR gap for any QAM system with probability of one dimensional symbol error equal to 10^{-4} is

$$\Gamma = 7.029 + \gamma_m - \gamma_c \tag{4.11}$$

2. OFDM (Multi-Carrier) Analysis

With the single carrier analysis complete, multi-carrier analysis can now be performed as an extension of the single carrier results. In a generic multicarrier system the probability of error is an average of the probability of error for each of the subchannels. Therefore the overall probability of error will be dominated by the weakest subcarriers. [Ref. 3]

This weakness was pointed out earlier in the thesis. Forward error correction coding with block interleaving is used to average out the probability of error over the active subcarriers. Therefore in the following analysis the probability of error is assumed equal over all subchannels. Directly applying the single carrier analysis yields the following results for the ith subchannel

$$3\Gamma = \frac{d_{\min,\,i}^{2}}{4\sigma_{i}^{2}} = \frac{\left|H_{i}\right|^{2} d_{i}^{2}}{4\sigma_{i}^{2}} \tag{4.12}$$

where the subscript *i* denotes any values that are subchannel dependent. The maximum number of bits that can be carried on the i^{th} subchannel with a given margin and coding gain is

$$b_{P_e,i} = \log_2\left(1 + \frac{SNR_i}{\Gamma}\right) \tag{4.13}$$

where

$$SNR_i = \left|H_i\right|^2 \frac{\varepsilon_i}{2\sigma_i^2}. \tag{4.14}$$

The subsymbol energy, ε_i , is held constant over all subchannels that are active and is zero on the inactive subchannels. The water pouring distribution (see Proakis [Ref. 13] or Gallager [Ref. 14]), is a better energy distribution but Cioffi has determined that

the on/off energy distribution is very close to optimal yet is much easier to compute and implement. [Ref. 3]

Figure 4.1 plots curves of b_{Pe} versus received channel SNR for three different values of one-dimensional probability of symbol error, P_{e1}. Figure 4.1 assumes that the coding gain is equal to the margin which may be zero. Therefore actual results in simulation or that may be seen in real world application may shift left or right as a function of the margin and coding gain. If the coding gain is greater than the margin the curves will shift left thereby predicting greater numbers of bits per subcarrier at a given received SNR. The coding gain achieved in the simulation is not calculated but is assumed to be around 6 to 8 dB which will be shown to be accurate in the following results.

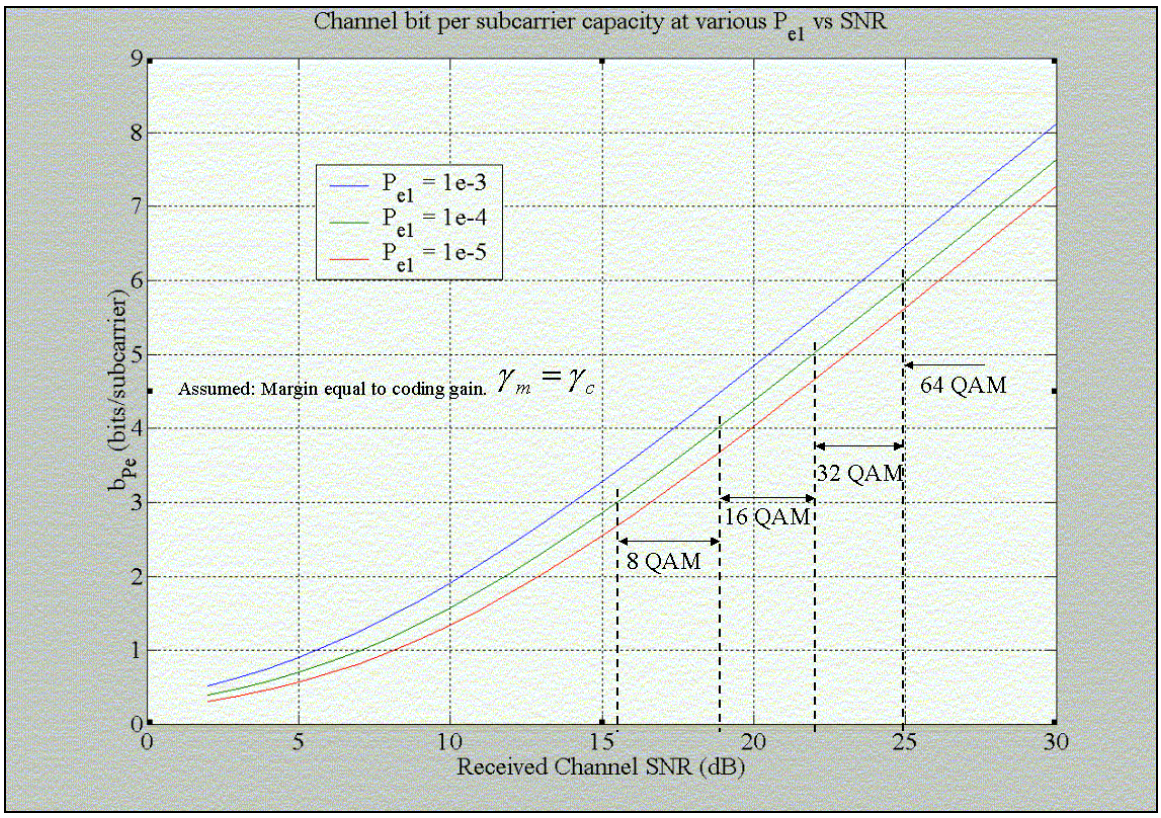


Figure 4.1. Channel Bit per Subcarrier Capacity.

The bit rate, R, of the total system is then simply the product of the active subcarriers with the number of bits per subcarrier divided by the OFDM symbol period,

T. Since all active subcarriers have the same number of bits, b_{sc}, which is rounded down to the nearest integer below $b_{P_e,i}$, the system bit rate is

$$R = \frac{\sum_{i=1}^{\overline{N}_{act}} b_{sc}}{T} = \frac{b}{T}. \tag{4.15}$$

Figure 4.2 plots system bandwidth versus bit rate for several sizes of QAM constellations.

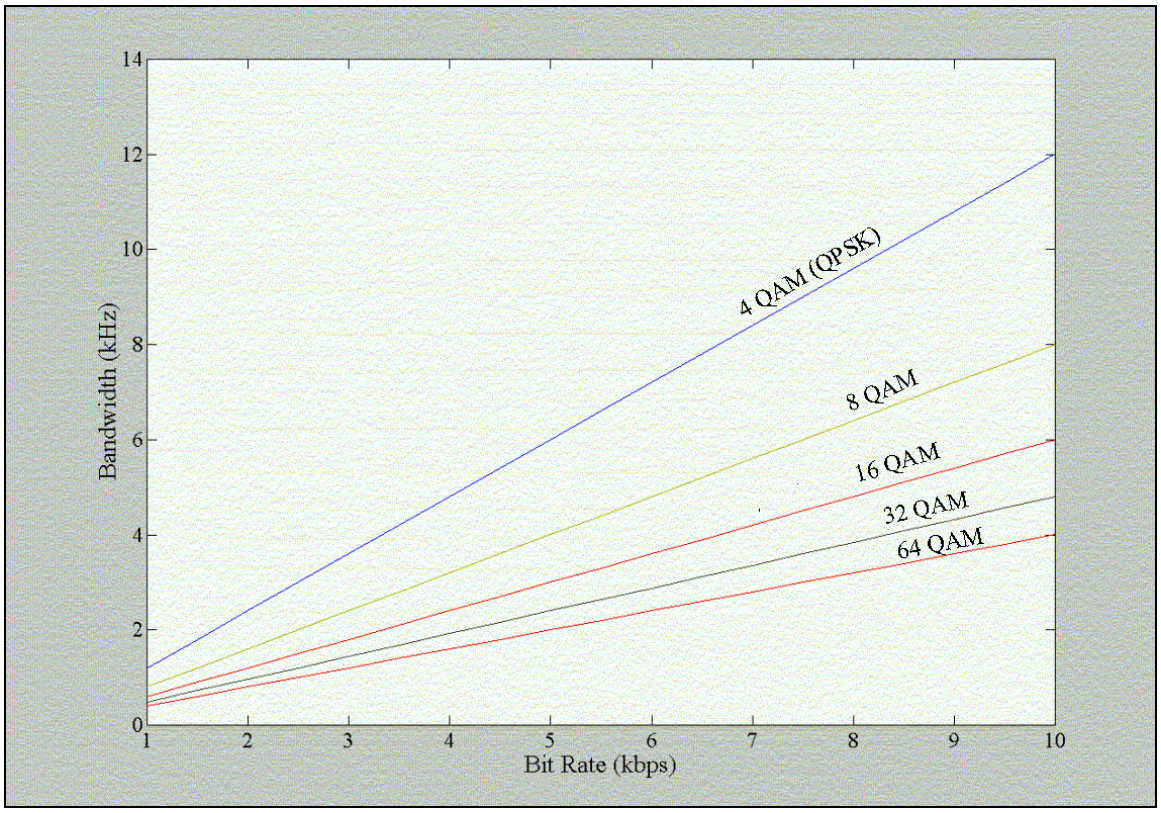


Figure 4.2. Bandwidth vs Bit Rate for Q-ary QAM OFDM Signals.

B. SIMULATION RESULTS

The remainder of this chapter presents the results of the simulation over the acoustic channel models developed. The main parameters of the acoustic channel are the water depth, the bottom roughness in meters of standard deviation, the range between the

receiver and the transmitter and the frequency band over which the model is evaluated. The water depths evaluated are either 100 m or 340 m. Three different bottom roughness values of 0, 2 and 4 m standard deviation were evaluated. The ranges evaluated were either 2 km or 4 km.

Two of the available forms of output from the MMPE model are included for every acoustic model evaluated. The first is a plot of transmission loss as a function of frequency and depth. The second is a plot of transmission loss versus depth and time. These plots show the ray paths present in the water column over time such that for a given depth the multipath arrivals are apparent.

While each acoustic model was evaluated for ranges of 2 and 4 km, only the plots for 2 km ranges are provided. Figure 4.3 shows the transmission loss for the acoustic channel that is 100 m deep and has a bottom roughness of 4 m. The effect of the bottom roughness on the acoustic energy is clearly seen in Figure 4.3 and can be compared with the transmission loss curves of the same acoustic channel with less bottom roughness, which are Figures B.1 and B.3, to further illustrate the effect. Note the interference patterns and blending of transmission near the interface that is much less pronounced as the bottom roughness goes down. Figure 4.4 shows the acoustic energy arrival as a function of time and depth for the same acoustic channel. The severe multipath environment present in shallow water acoustic channels is readily apparent in Figure 4.4. All other acoustic channel plots are found in Appendix B.

The simulation was performed for several combinations of bit rate, center frequency and constellation sizes. Each combination was simulated 10 times and then the mean of the result was taken. The primary output of the simulation and this thesis are plots of BER versus received SNR. Recall that the simulation is performed with the margin, γ_m, set equal to zero. Accepted values for γ_m range from 6 to 12 dB, with Cioffi [Ref. 3] recommending 6 dB.

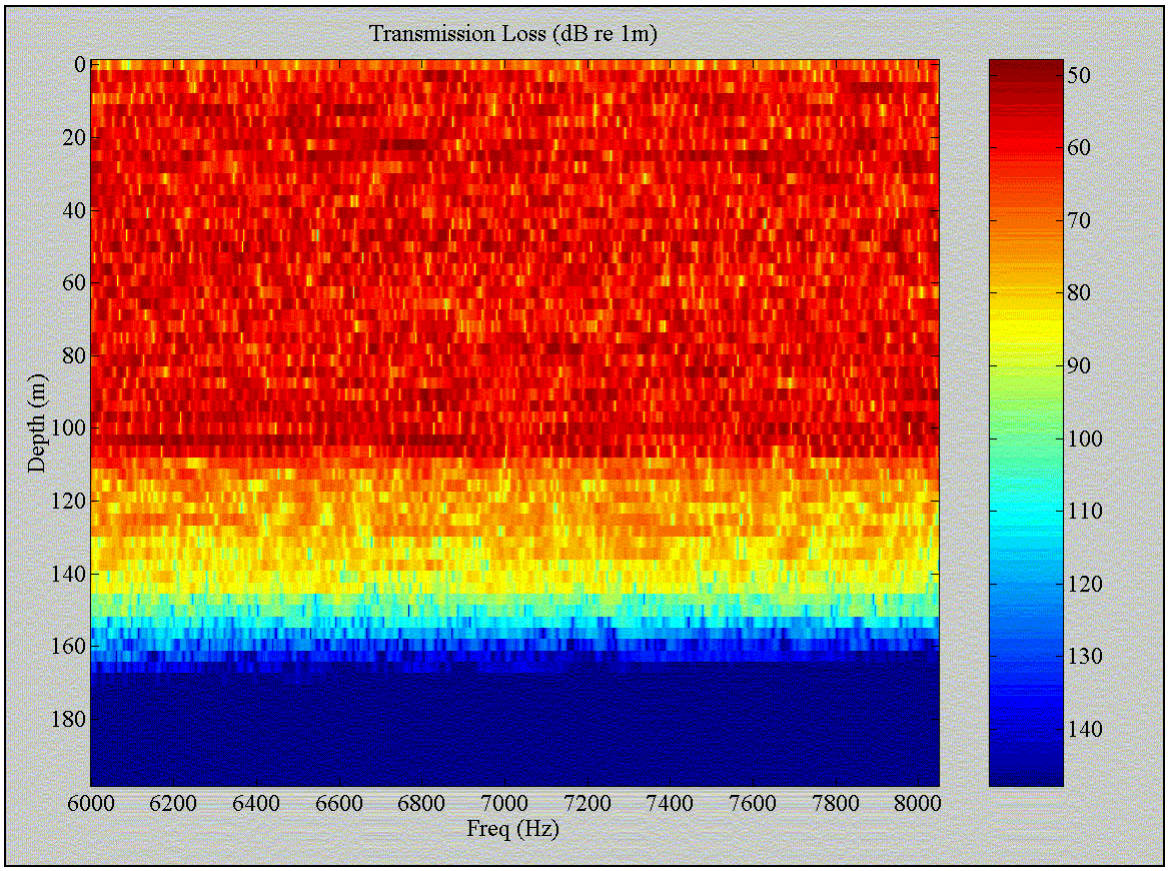


Figure 4.3. Transmission Loss, Bottom Roughness 4 m, Water Depth 100 m.

The margin can be considered as an adjustment for simulations to shift the results towards what can be expected in reality. Recall also that the coding gain, γ_c, is estimated to be 4 to 6 dB. Therefore, since the coding gain is included in the results of the simulation, in order to check the results versus the theoretical prediction of Fig. 4.1 the BER versus received SNR must be shifted to the right by approximately 11 dB.

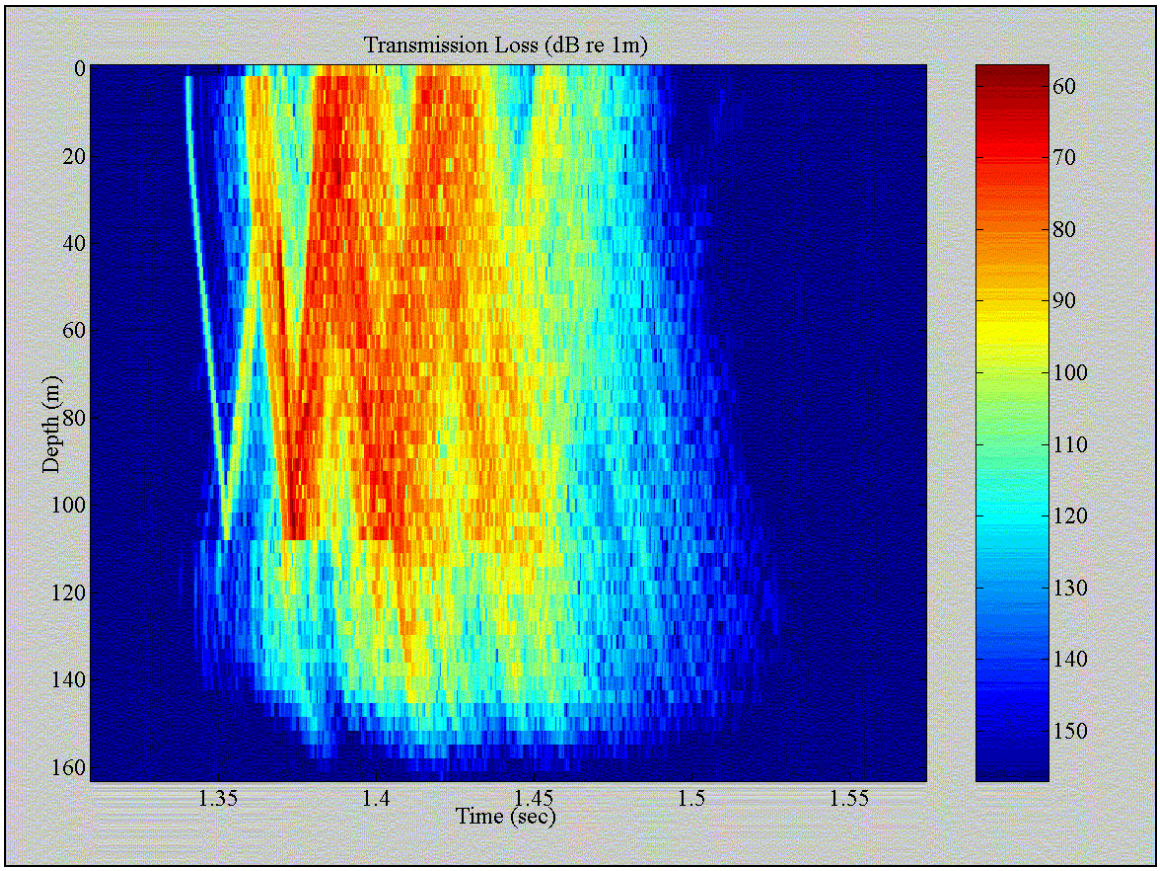


Figure 4.4. Acoustic Energy Arrival, Bottom Roughness 4 m, Water Depth 100 m.

The raw results are provided in Appendix A. In these plots each data point indicates the result of one run of the simulation. The solid red line is the mean of the 10 runs performed. The composite results are provided in Figures 4.5 through 4.8 for QAM constellation sizes of 8, 16, 32 and 64. Each solid line in the figures is the mean of the corresponding series of 10 runs plotted individually in Appendix A. Figure 4.5 presents the results for all evaluations using 8-ary QAM.

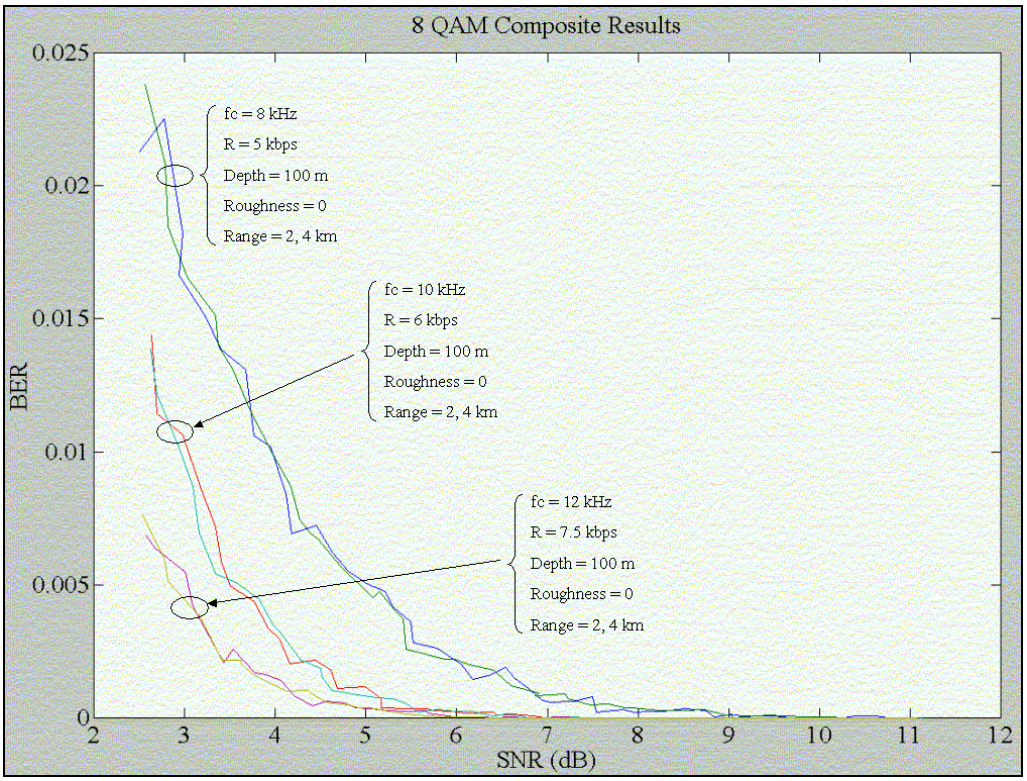


Figure 4.5. 8 QAM Composite Results.

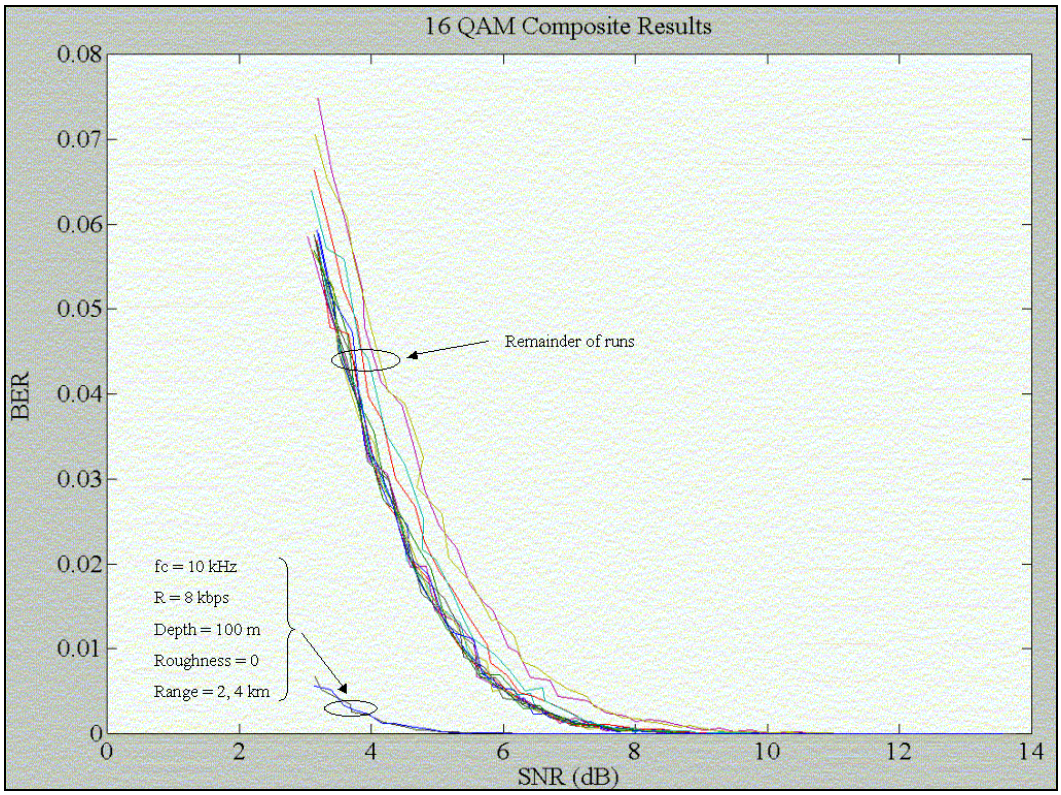


Figure 4.6. 16 QAM Composite Results.

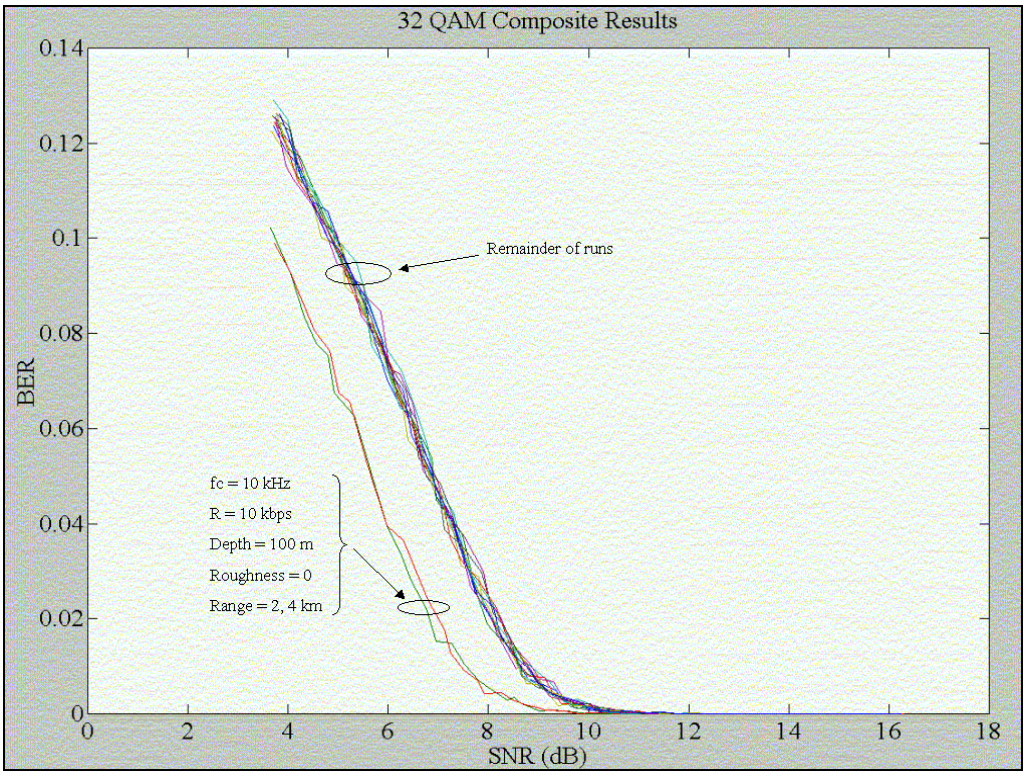


Figure 4.7. 32 QAM Composite Results.

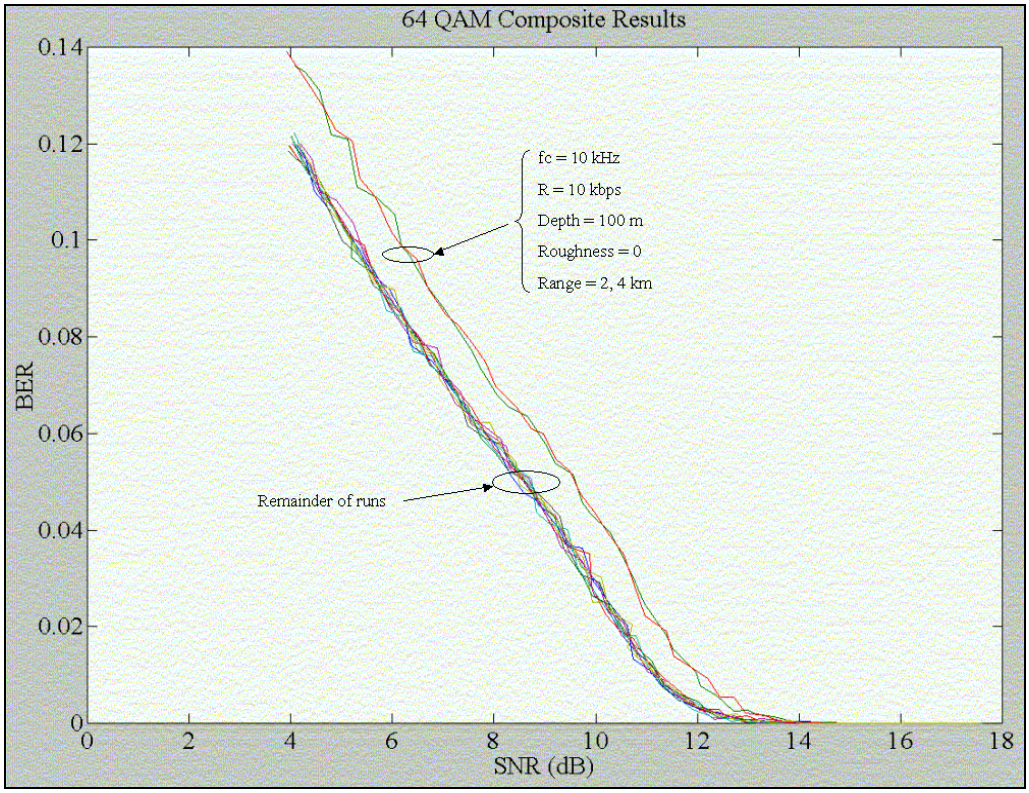


Figure 4.8. 64 QAM Composite Results.

Figure 4.9 provides an appreciation for the nature of the decoded signal and its frequency spectrum with respect to the information signal and its frequency spectrum.

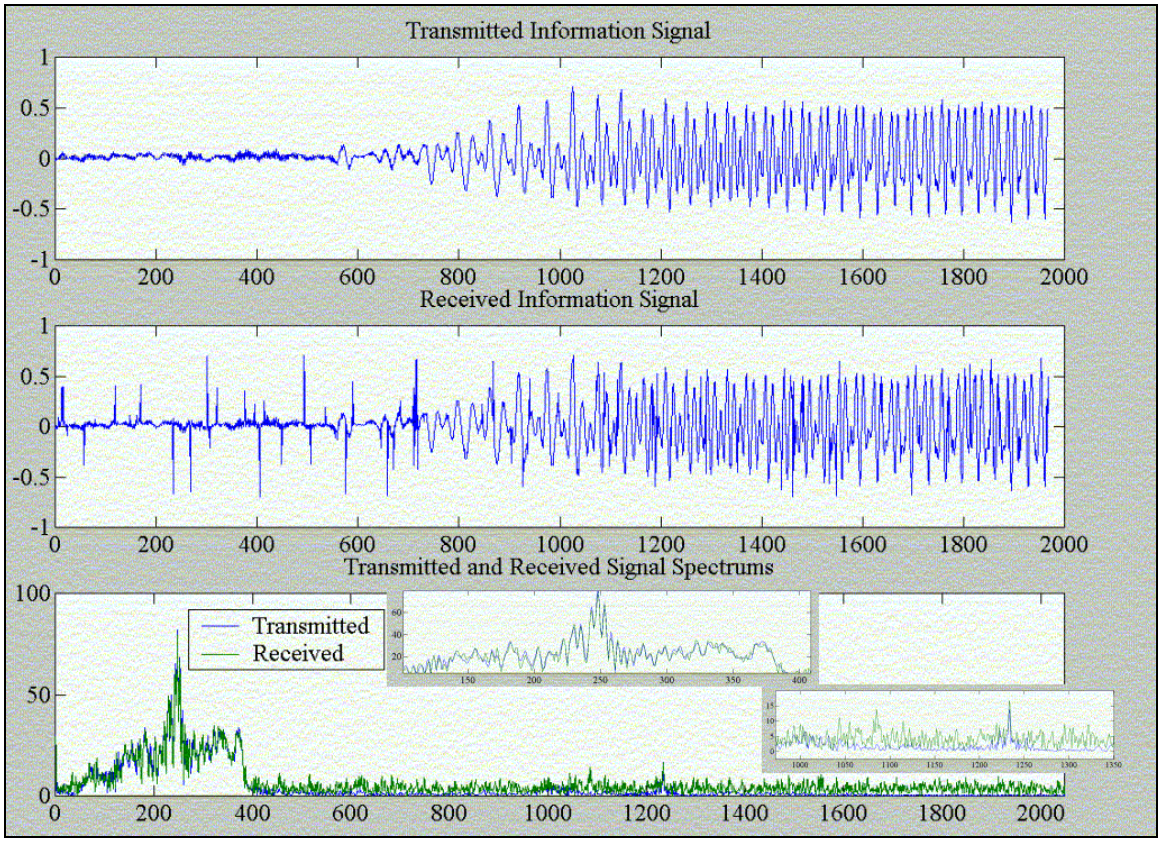


Figure 4.9. 16 QAM Signal Realization, f_c = 6 kHz, R = 5 kbps, SNR = 5.3 dB BER = 1.6 E-2.

The BER is unacceptable by communication system standards. The information signal for the simulation is a voice stream. Playback of the received signal at the BER of Figure 4.9 contains minor static but is clearly understandable. The zoomed-in subplots show the deviation of the received information signal spectrum from the transmitted information signal spectrum. One can see that where there is significant power in the frequency spectrum of the transmitted information signal the received spectrum is very near that of the transmitted spectrum. Comparing the time domain plots of the transmitted and received information signals one can see the deviations of the received signal relative to the original information signal. Figure 4.10 shows the received QAM constellation for

the realization of Figure 4.9 after channel estimation and equalization but prior to Reed Solomon decoding.

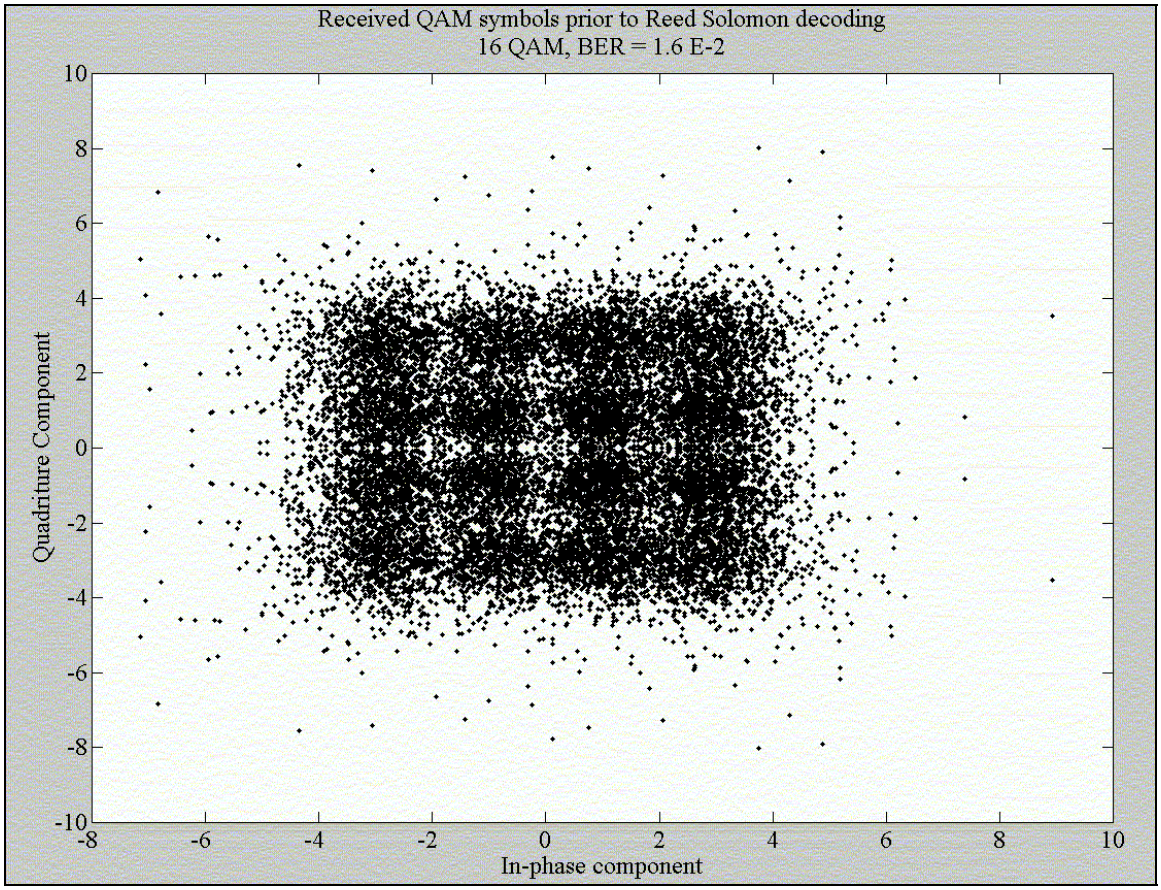


Figure 4.10. 16 QAM Signal Realization, f_c = 6 kHz, R = 5 kbps, SNR = 5.3 dB BER = 1.6 E-2.

Figures 4.11 and 4.12 provide a contrast to the realizations of Figures 4.9 and 4.10 at a greater received SNR of 7.4 dB and the resulting smaller BER of 2.5 E-4. The BER is still greater than the acceptable limit of 1 E-4. However one can see that the received signal is essentially an exact replica of the transmitted information signal. Also the powerful effect of the FEC can be seen from the noisy constellations of Figures 4.10 and 4.12.

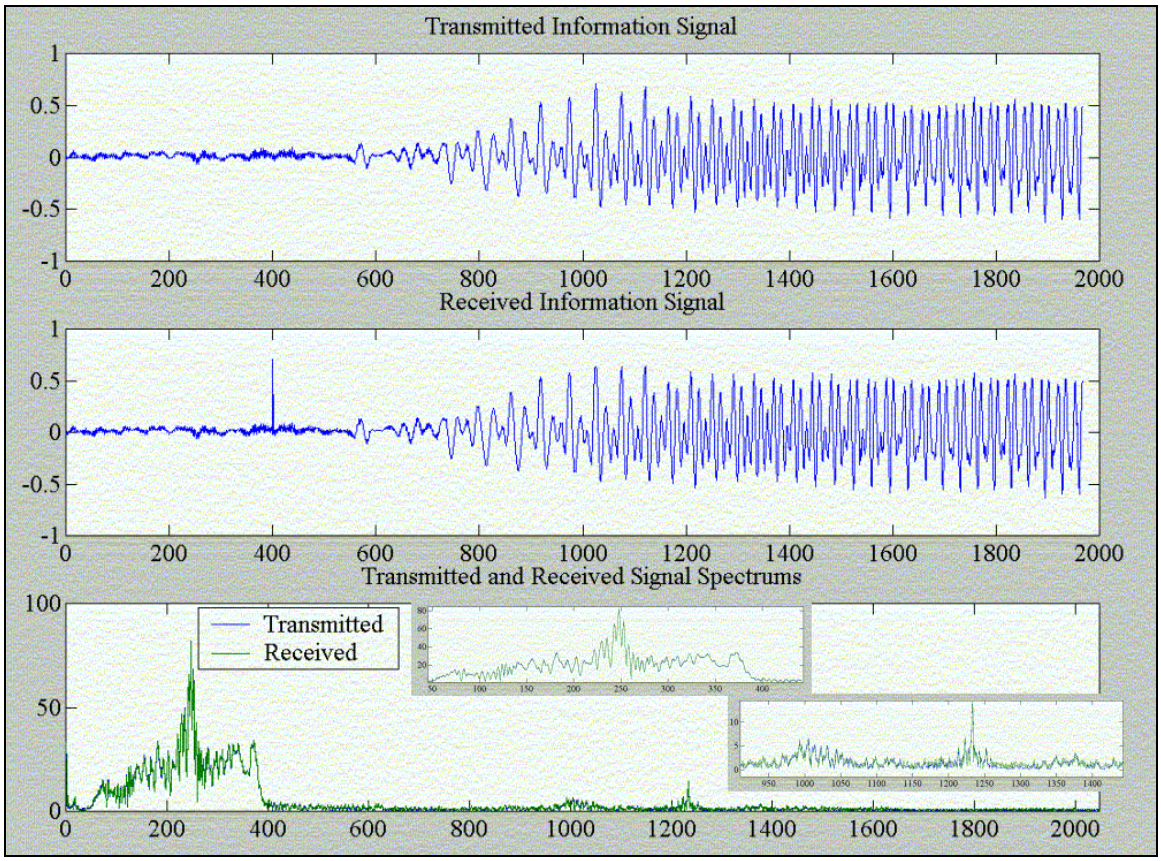


Figure 4.11. 16 QAM Signal Realization, f_c = 6 kHz, R = 5 kbps, SNR = 7.4 dB BER = 2.5 E-4.

Note that the zoomed plots of the frequency spectrums of Fig. 4.11 show that the received spectra is essentially the same as that of the transmitted information signal spectrum even in the areas of the spectrum where the transmitted information signal has relatively low power.

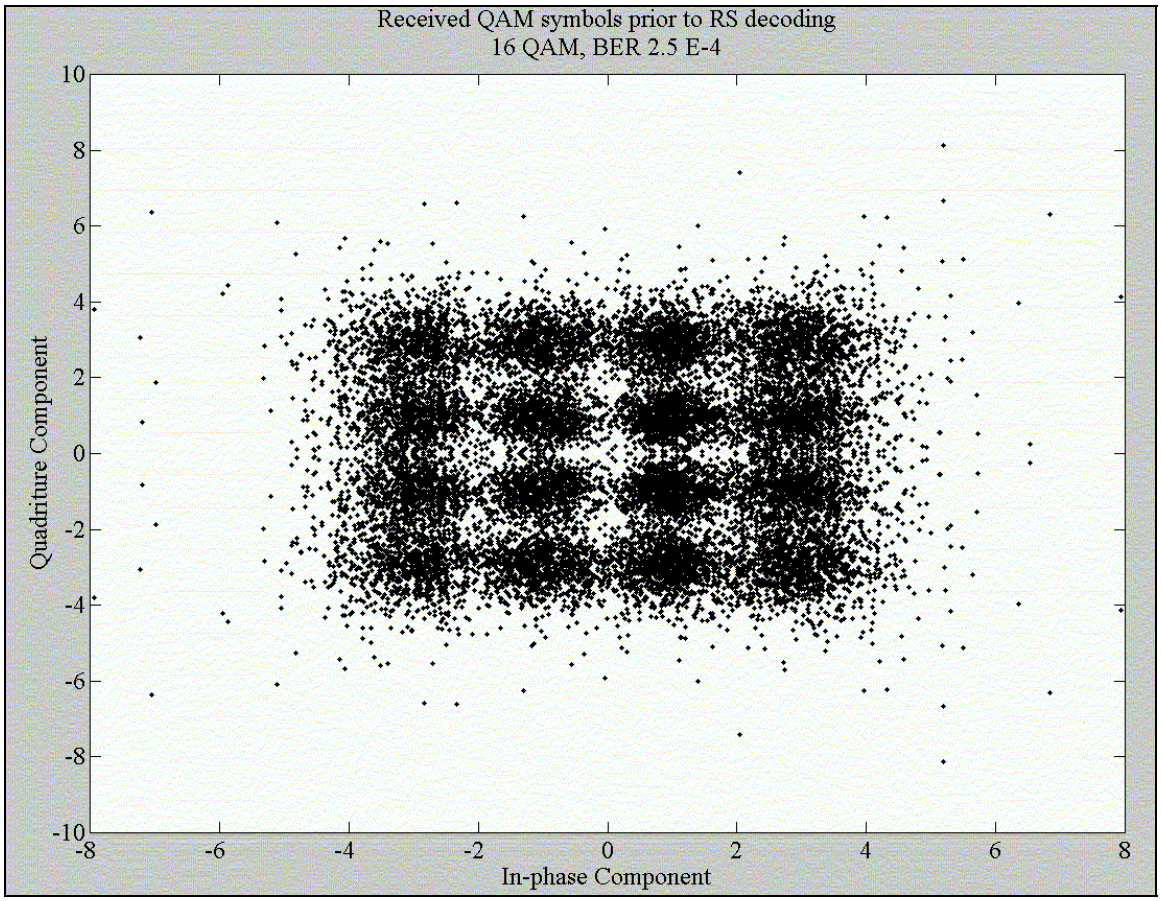


Figure 4.12. 16 QAM Signal Realization, f_c = 6 kHz, R = 5 kbps, SNR = 7.4 dB BER = 2.5 E-4.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS

The primary thrust of this thesis was to develop a computer simulation of an OFDM system for underwater acoustic communication using a PE based model of the acoustic channel. The simulation was validated by comparison of the results of the simulations with the known theoretical results established in this thesis and those from ocean experiments. The comparison of the results with what is expected from the theory was made in conjunction with presenting the results. Coatelan and Glavieux [Ref. 5] experimented with a OFDM communication system in shallow water acoustic channels with results as shown in Table 5.1. The QAM constellation size used by Coatelan and Glavieux [Ref. 1] is not known. Note that the water depth of the experiments is much shallower than for the acoustic channels used in the simulations of this thesis. The FEC code employed in the experiments, which uses rate ½ convolutional coding with a constraint length of seven as well as a soft Viterbi decoder working on a 64 state trellis [Ref. 1], is more advanced than that of the simulation, however the coding gain is likely within a couple of dB of the coding gain for the simulation as more than a few dB of coding gain improvement beyond 5 dB is difficult to obtain. Comparing the composite results presented earlier for the 8 and 16 QAM constellations and applying a margin of 6 to 12 dB one can see that simulation results are comparable to the experimental results of Table 5.1.

Transmission distance	Water depth	Seafloor type	Emitted number of bit	BER after demodulation	BER after decoding	estimated average level E_b/N_o
650 m	25 m	sludge	18500	$3,7.10^{-3}$	$< 1.10^{-4}$	27,5 dB
1920 m	27 m	sand	18500	$2,4.10^{-2}$	$< 1.10^{-4}$	16 dB

Table 1. ***Multicarrier transmission performances during two stations***

Table 5.1. Experimental Multicarrier Results [from Ref. 1].

The benefits of the work are the verification of the feasibility of using OFDM as a method of underwater acoustic communication and to provide a test-bed for future work to develop improved methods for the blocks that make up the OFDM system.

There is significant opportunity for future work in this area. Many of the algorithms applied in the OFDM system are simple and could be improved to enhance the performance of the OFDM system.

The modulation is DSBSC which uses twice the bandwidth of an Upper Side Band (USB) system. There are no obvious reasons why USB modulation could not be used to improve the spectral efficiency. This idea also has benefit in working on a multi-user application of OFDM.

Communication system protocols must be established to make the system useful for real world application and appears to have been neglected so far in the area of underwater acoustic communication. The IEEE standard 802.11 and the European digital terrestrial TV broadcast uses OFDM and has established protocols that may be applicable to underwater communication systems.

Synchronization algorithms have significant merit for future study and improvement. This study does not involve Doppler effects since it assumes that the transmitter and receiver are stationary.

The application of the channel transfer function must be modified in modeling source or receiver motion. Frequency offset due to wave action and phase noise are not simulated either. Both of these are real concerns in development of an actual system and are worthy of simulation and analysis. The synchronization method must be modified in the simulation to account for phase issues. In addition the synchronization method is very computationally intensive and simplification of the system would benefit the simulation and any resulting real system.

Channel estimation and equalization assumes a linear time invariant acoustic channel. The channel is known to be time varying with the rate of variance usually low but still of concern. Therefore a method of updating the channel estimation is necessary. Also while the estimation method used in the simulation is very robust there may be

better methods that use less bandwidth. FEC coding applied is elementary and can be improved.

Thus far work to improve the system and the degree of reality of the simulation have been discussed. Of equal importance is the study of the key parameters on the system performance. The peak to average power ratio of the transmit signal is of concern in OFDM and has not been addressed in this thesis.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. SIMULATION RESULTS

The figures in this appendix represent the results of the simulation for all of the acoustic channels evaluated. The parameters of the system are provided in each figure as well as the main channel parameters. Each figure includes a zoomed inset which illustrates the BER performance near the target level of 10^{-4}.

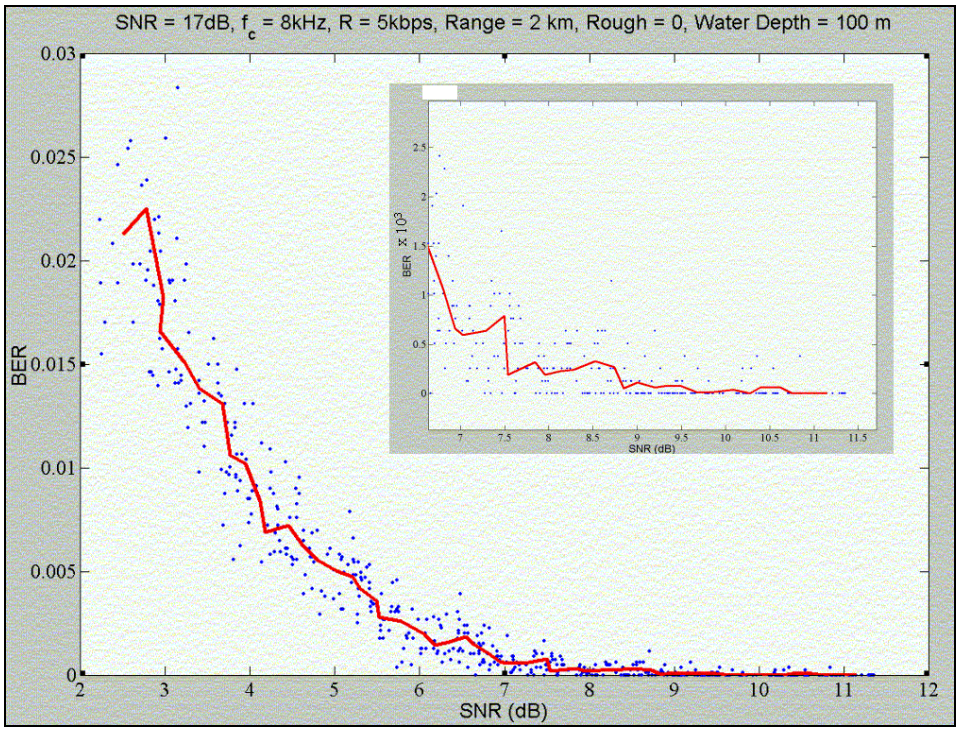


Figure A.1. BER vs SNR for 8 QAM, fc = 8 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.

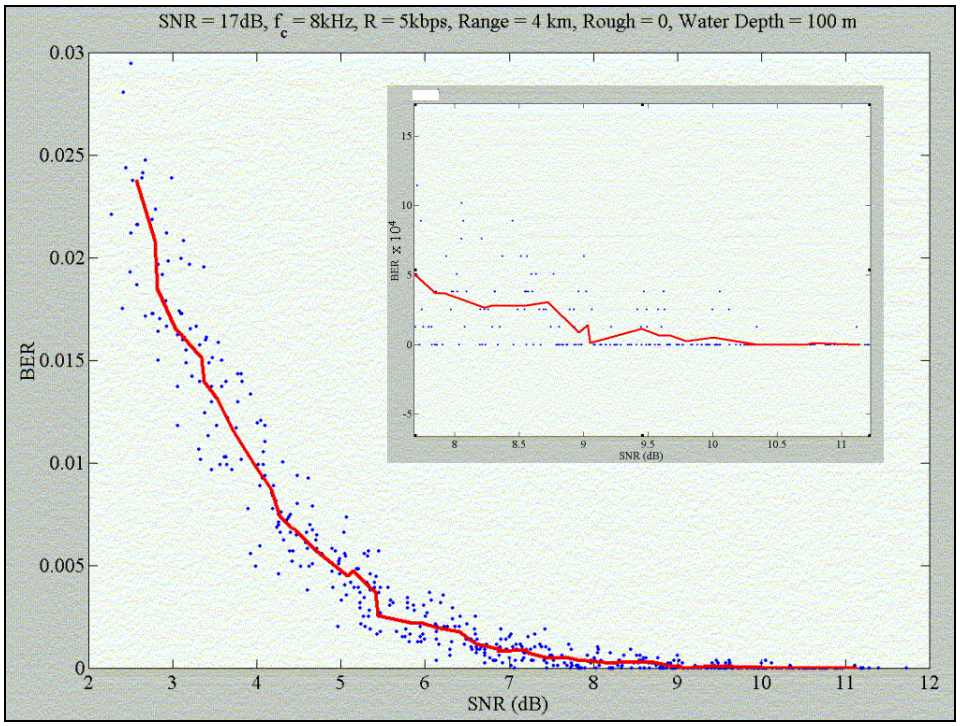


Figure A.2. BER vs SNR for 8 QAM, fc = 8 kHz, R = 5 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.

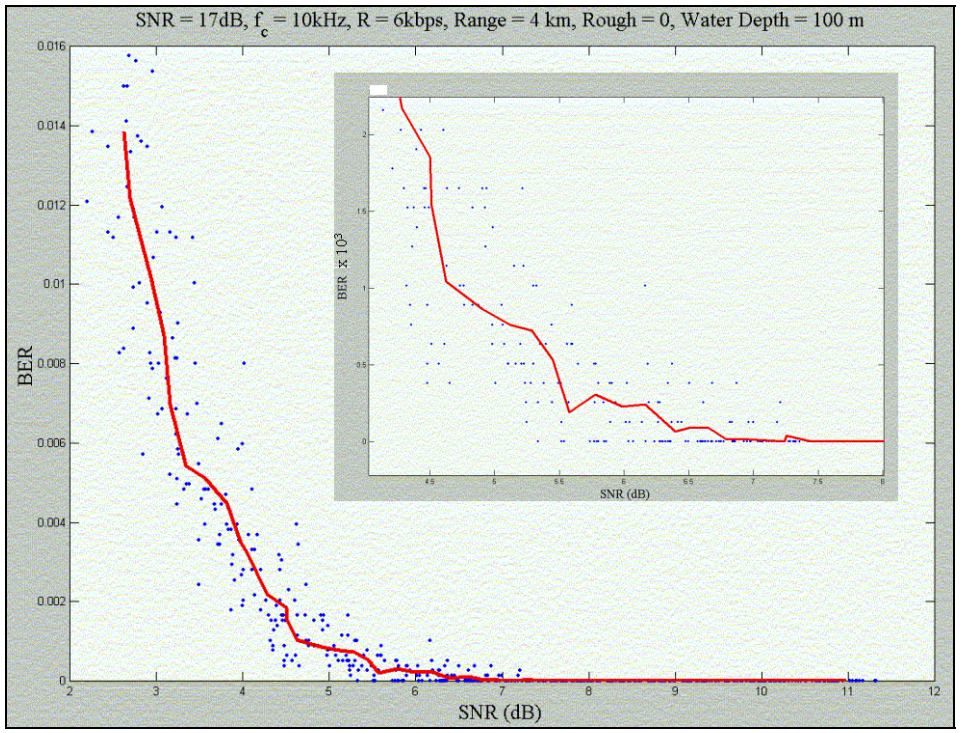


Figure A.3. BER vs SNR for 8 QAM, fc = 10 kHz, R = 6 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.

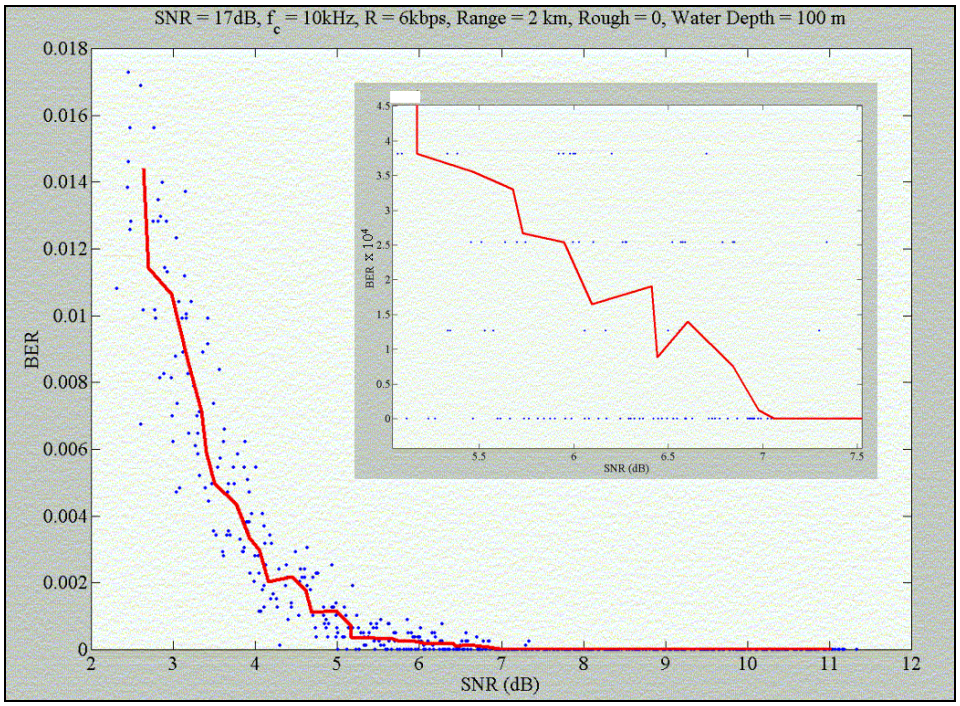


Figure A.4. BER vs SNR for 8 QAM, fc = 10 kHz, R = 6 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.

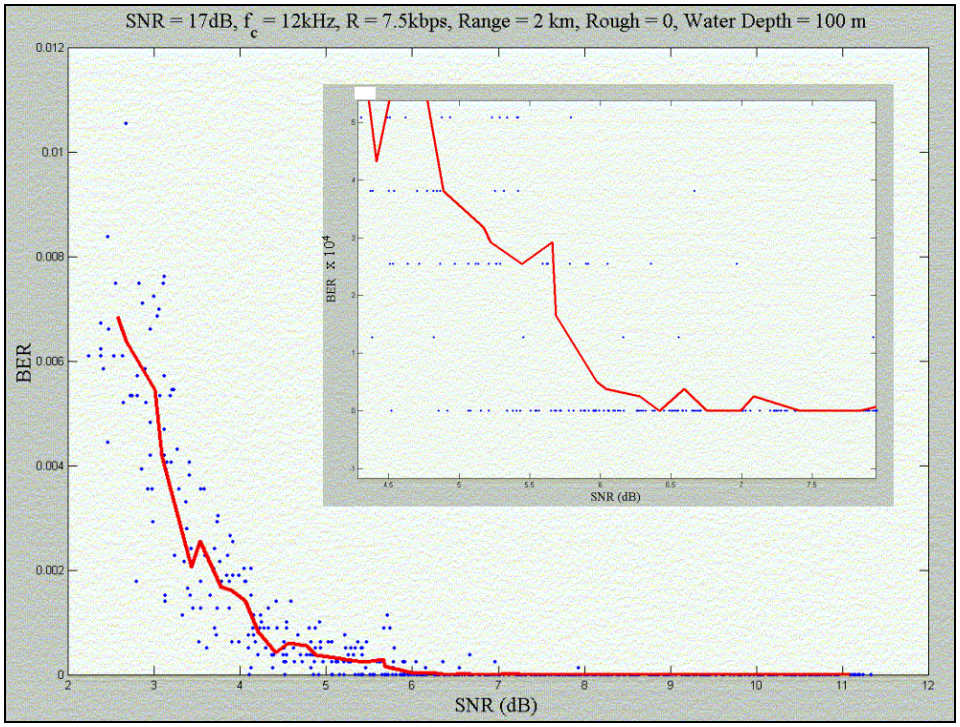


Figure A.5. BER vs SNR for 8 QAM, fc = 8 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.

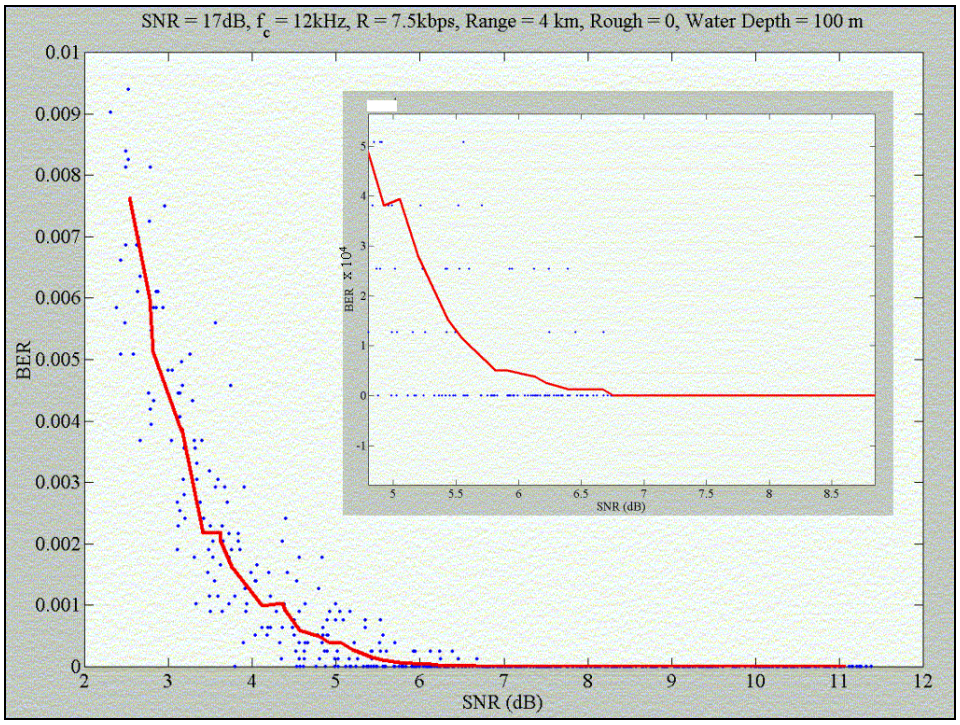


Figure A.6. BER vs SNR for 8 QAM, fc = 12 kHz, R = 7.5 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.

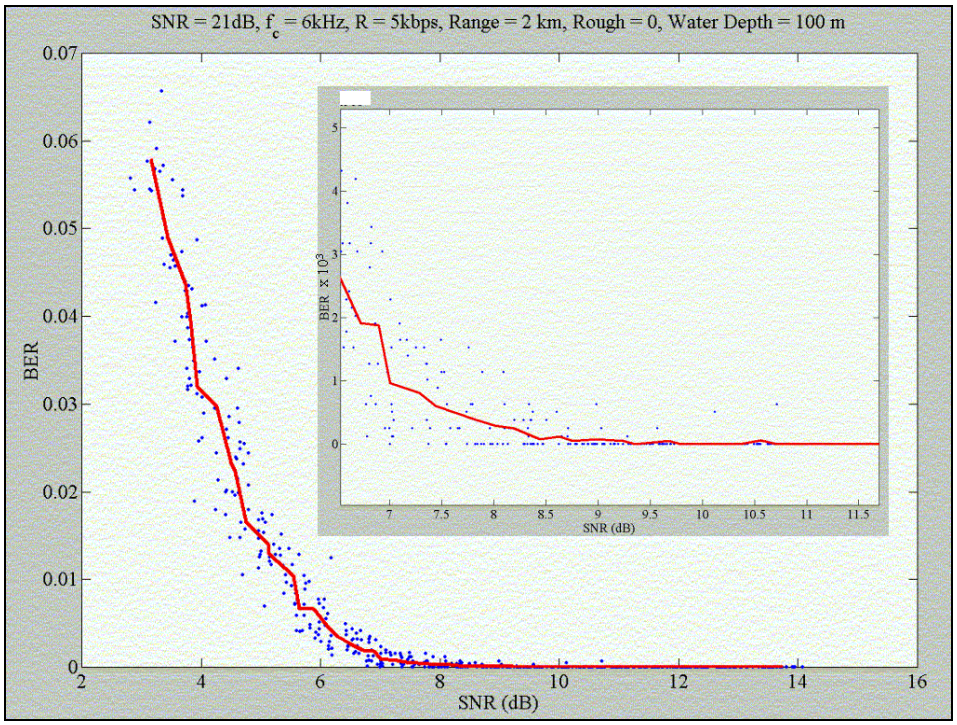


Figure A.7. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.

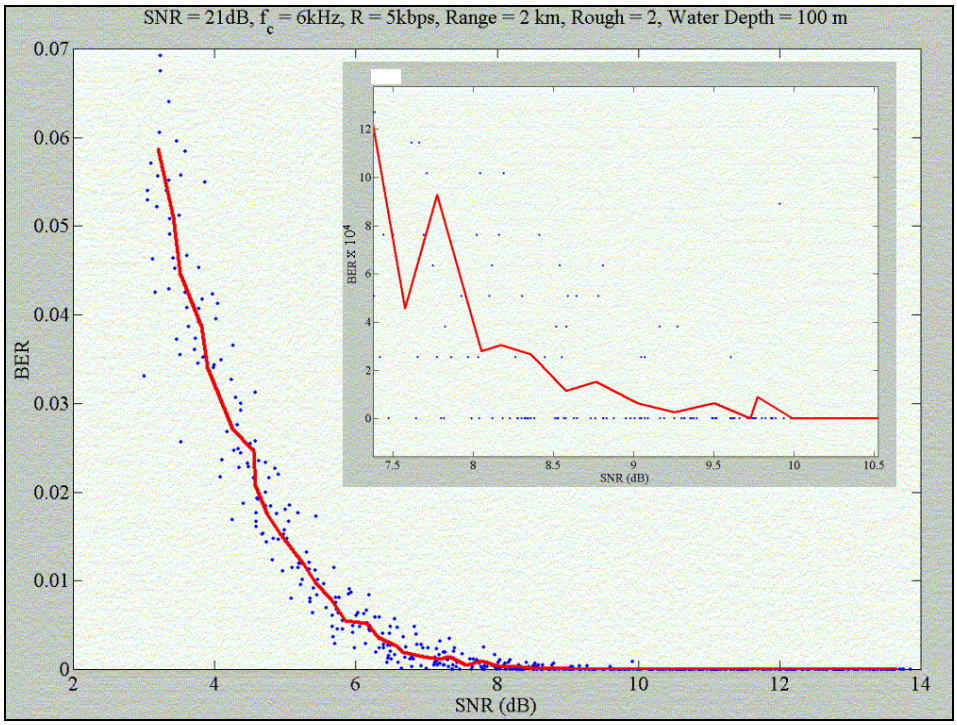


Figure A.8. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 2m, Water Depth = 100m.

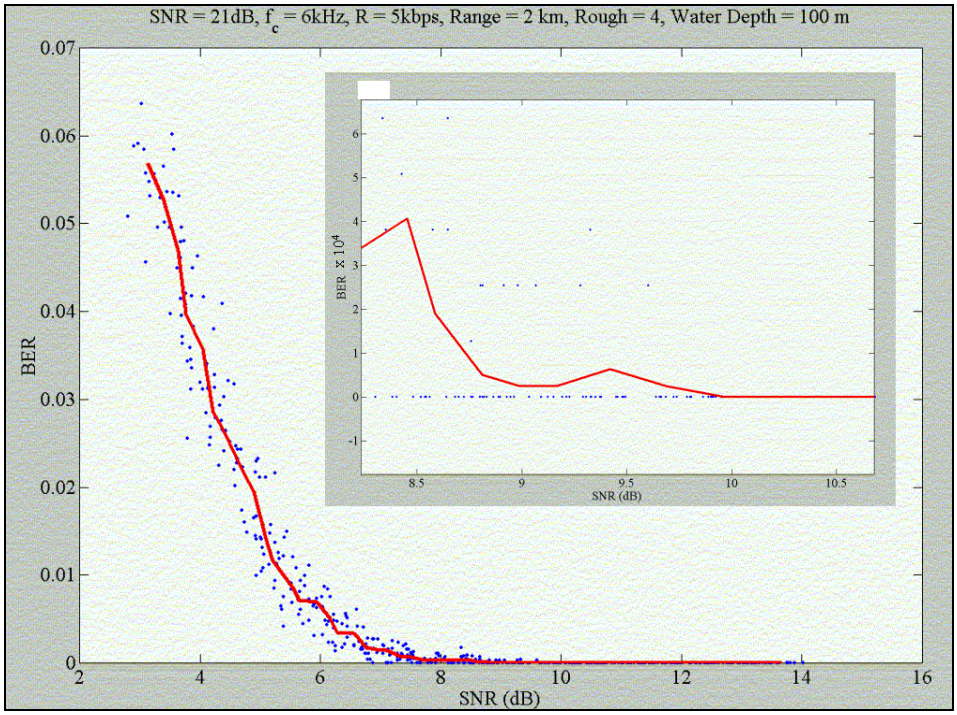


Figure A..9 BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 4m, Water Depth = 100m.

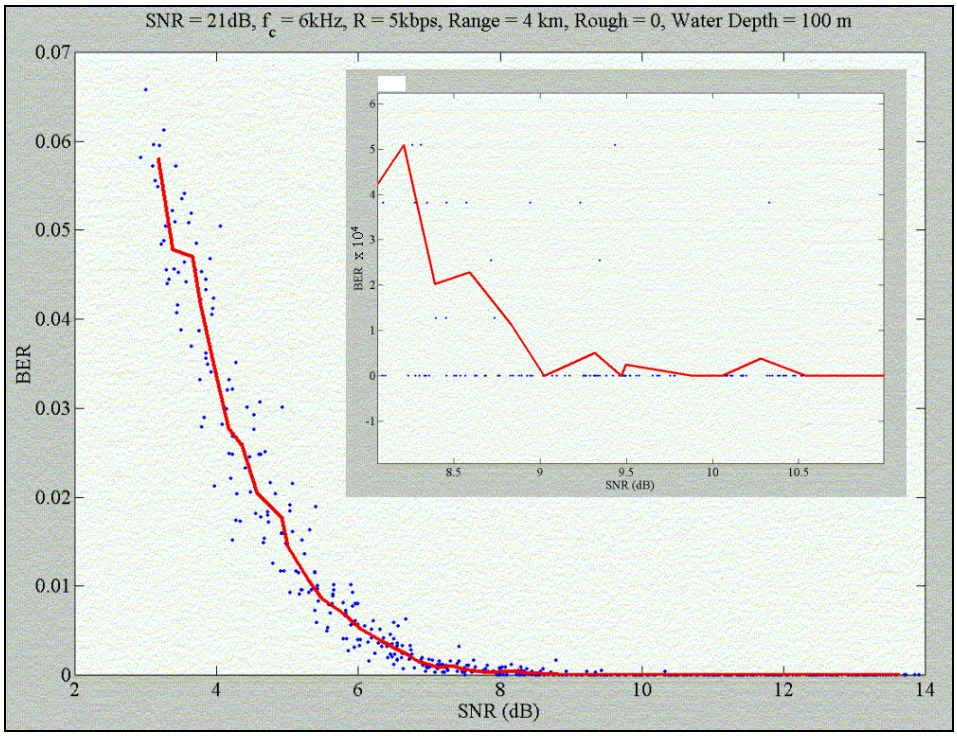


Figure A.10. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.

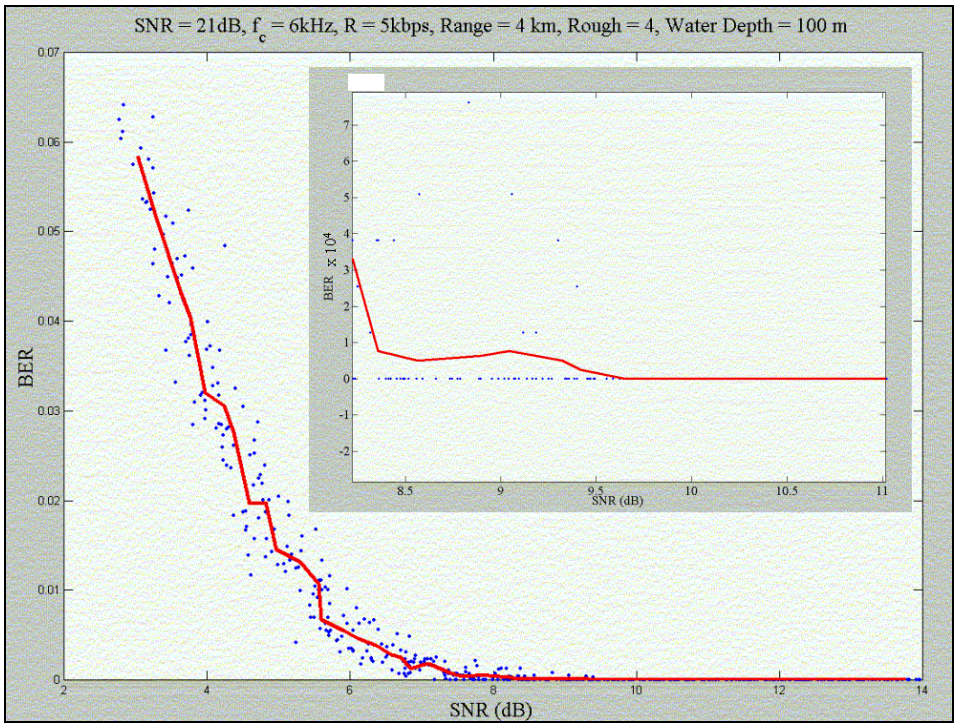


Figure A.11. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 4m, Water Depth = 100m.

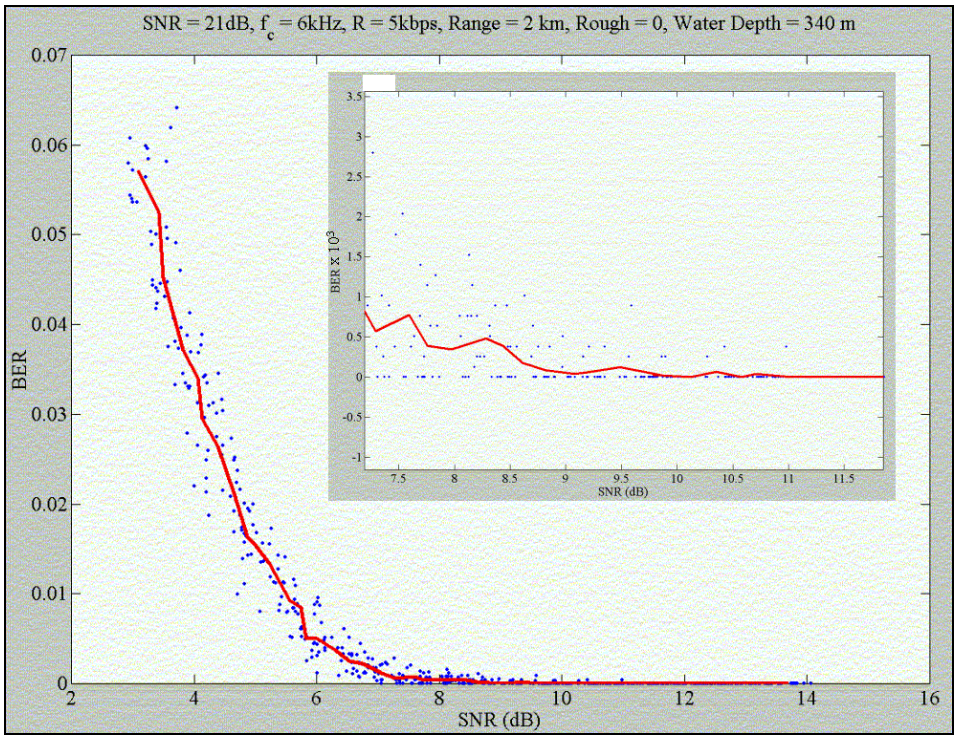


Figure A.12. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 340m.

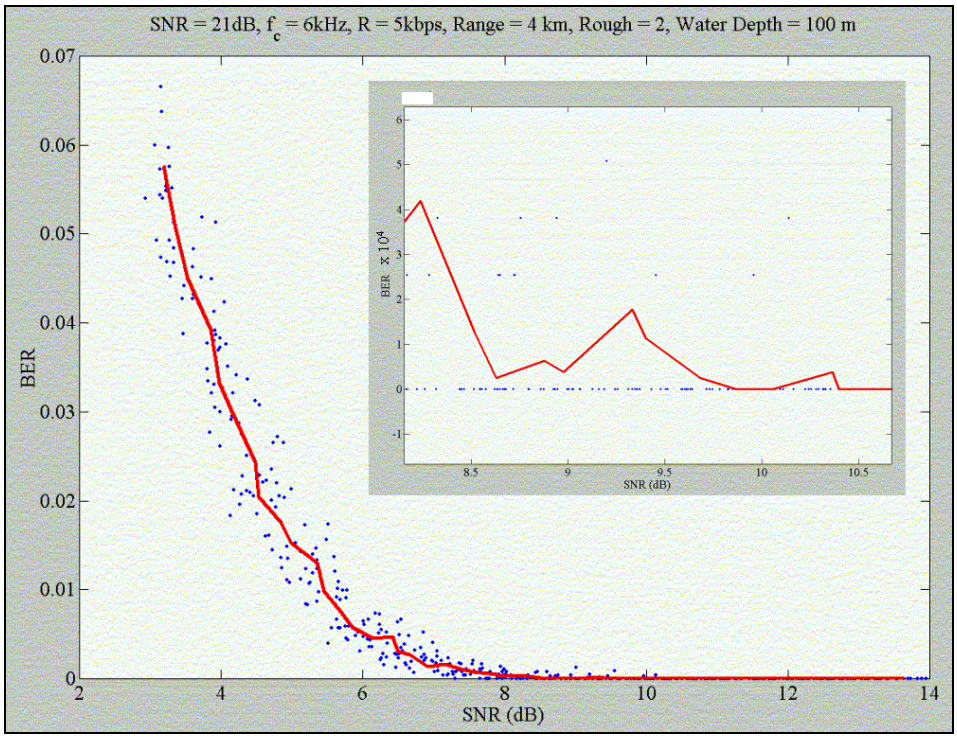


Figure A.13. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 2m, Water Depth = 100m.

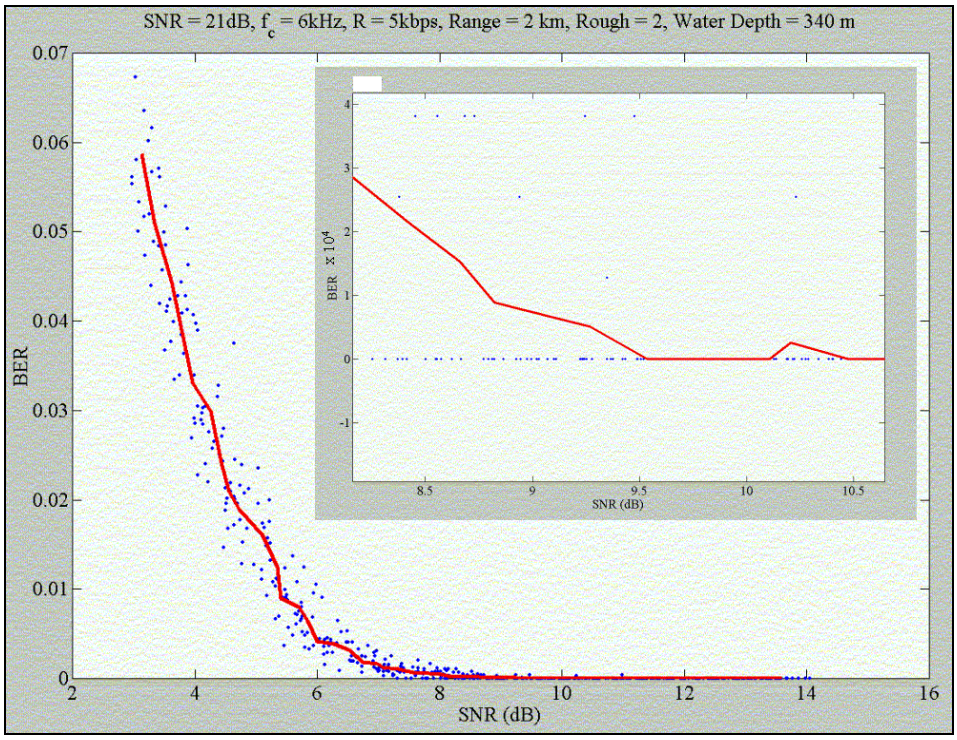


Figure A.14. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 2m, Water Depth = 340m.

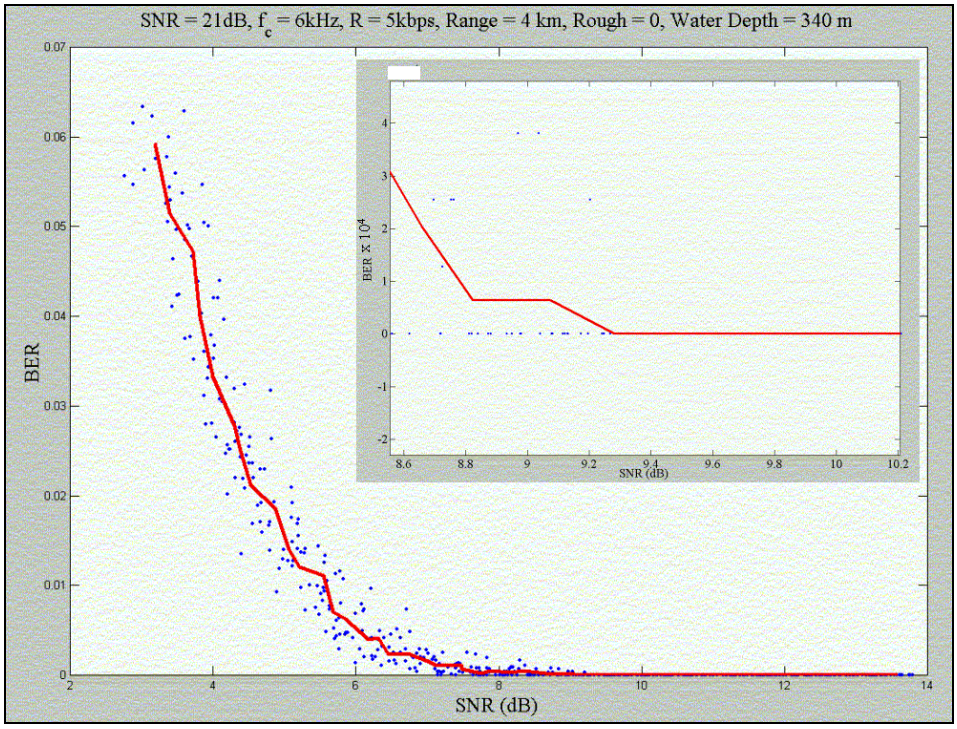


Figure A.15. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 0m, Water Depth = 340m.

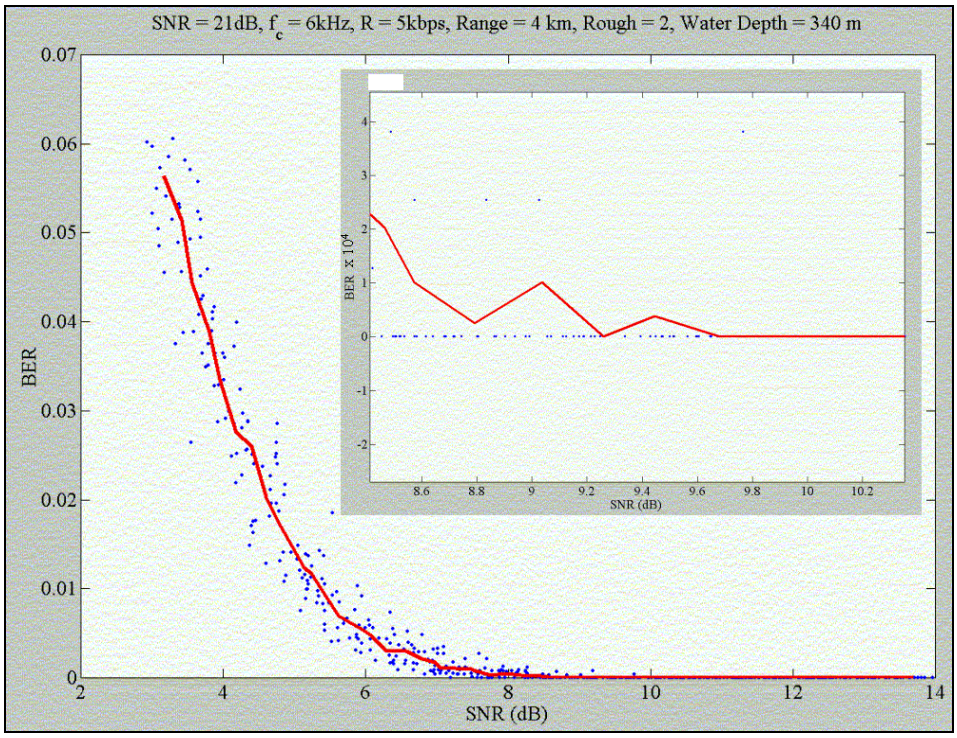


Figure A.16. BER vs SNR for 16 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 2m, Water Depth = 340m.

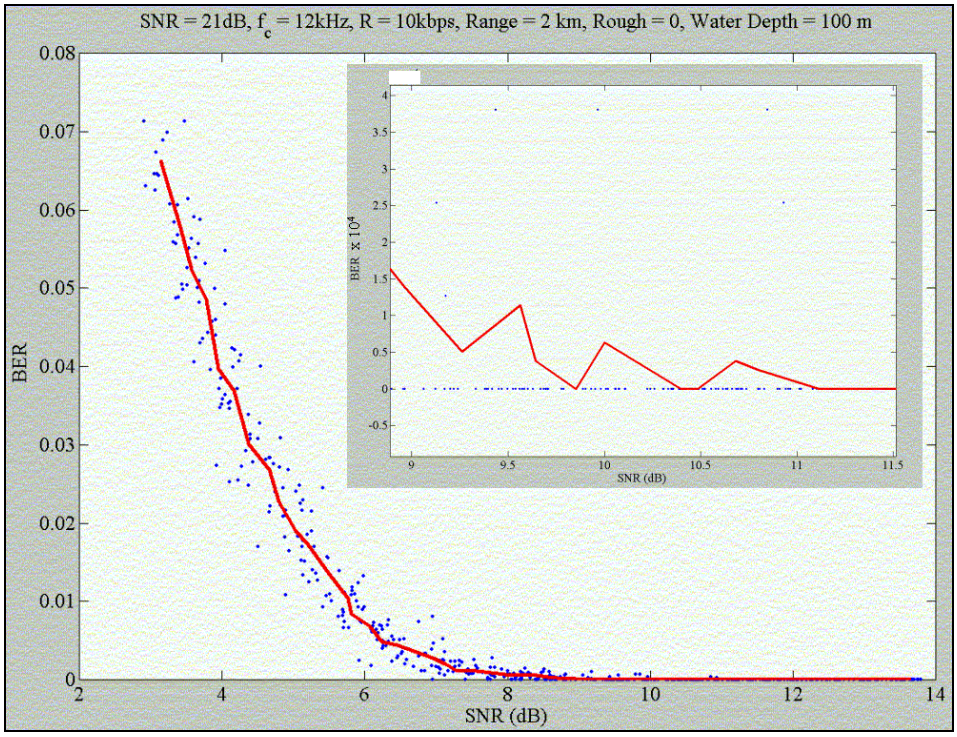


Figure A.17. BER vs SNR for 16 QAM, fc = 12 kHz, R = 10 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.

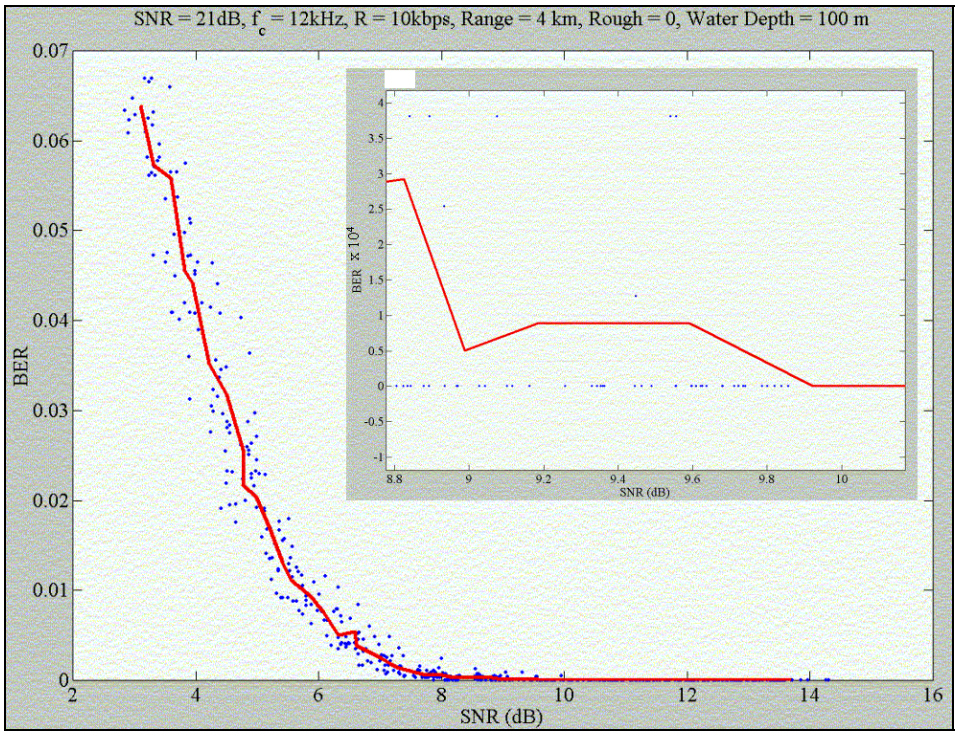


Figure A.18. BER vs SNR for 16 QAM, fc = 12 kHz, R = 10 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.

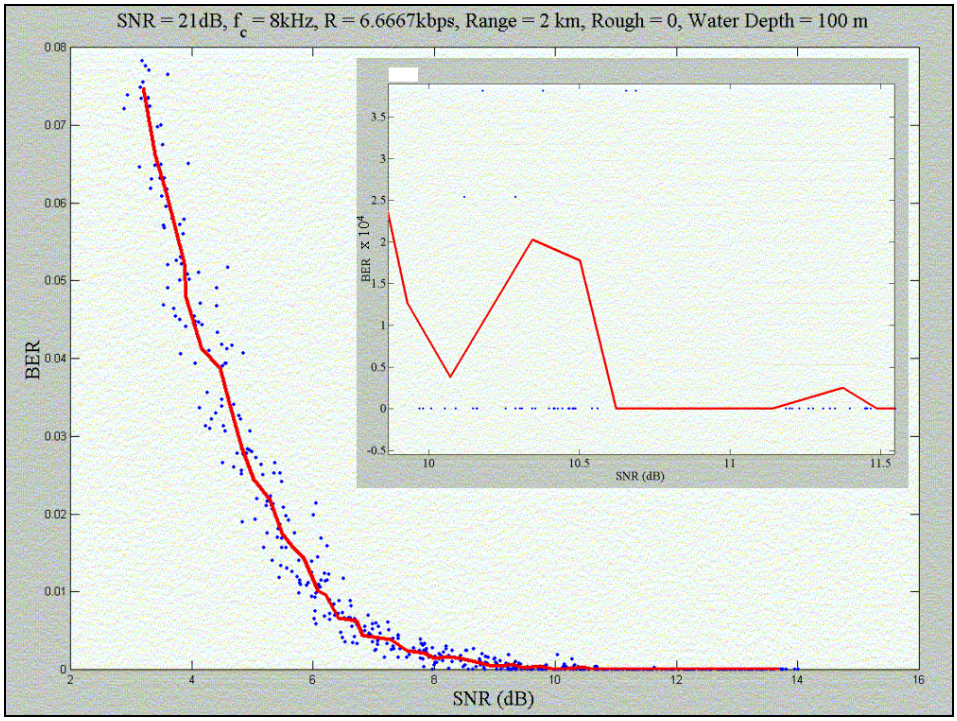


Figure A.19. BER vs SNR for 16 QAM, fc = 8 kHz, R = 6.67 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.

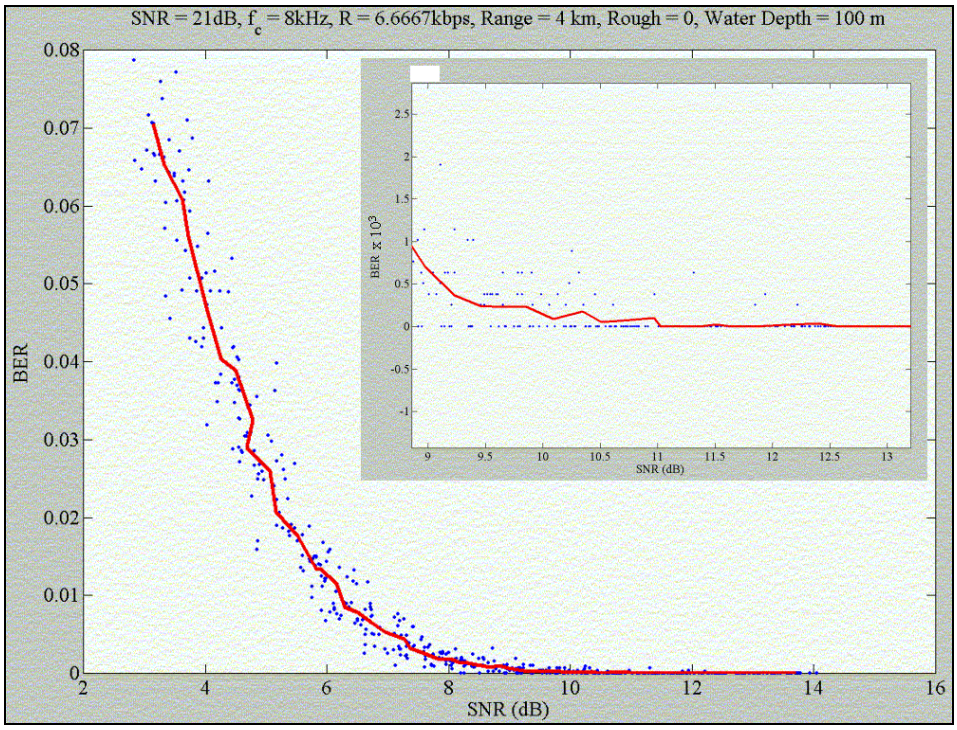


Figure A.20. BER vs SNR for 16 QAM, fc = 8 kHz, R = 6.67 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.

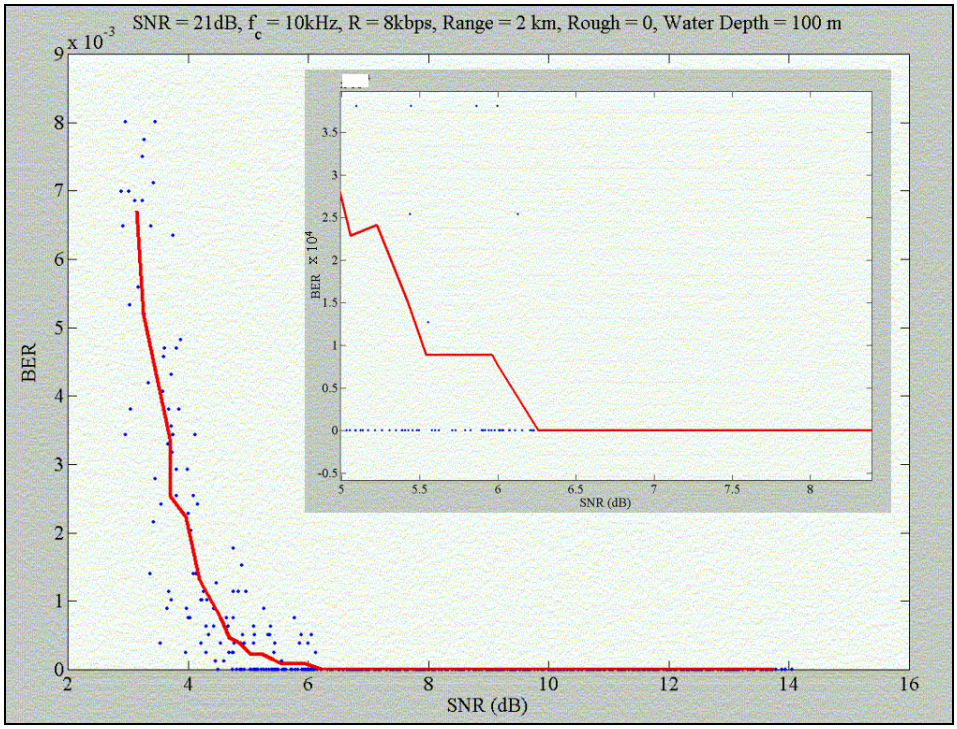


Figure A.21. BER vs SNR for 16 QAM, fc = 10 kHz, R = 8 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.

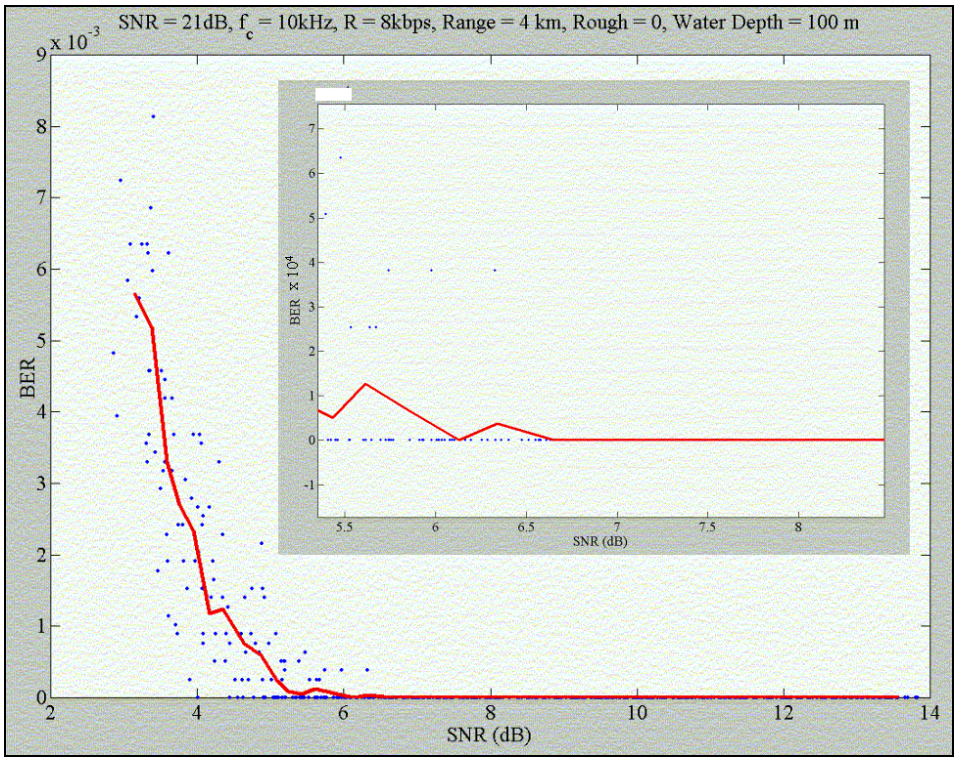


Figure A.22. BER vs SNR for 16 QAM, fc = 10 kHz, R = 8 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.

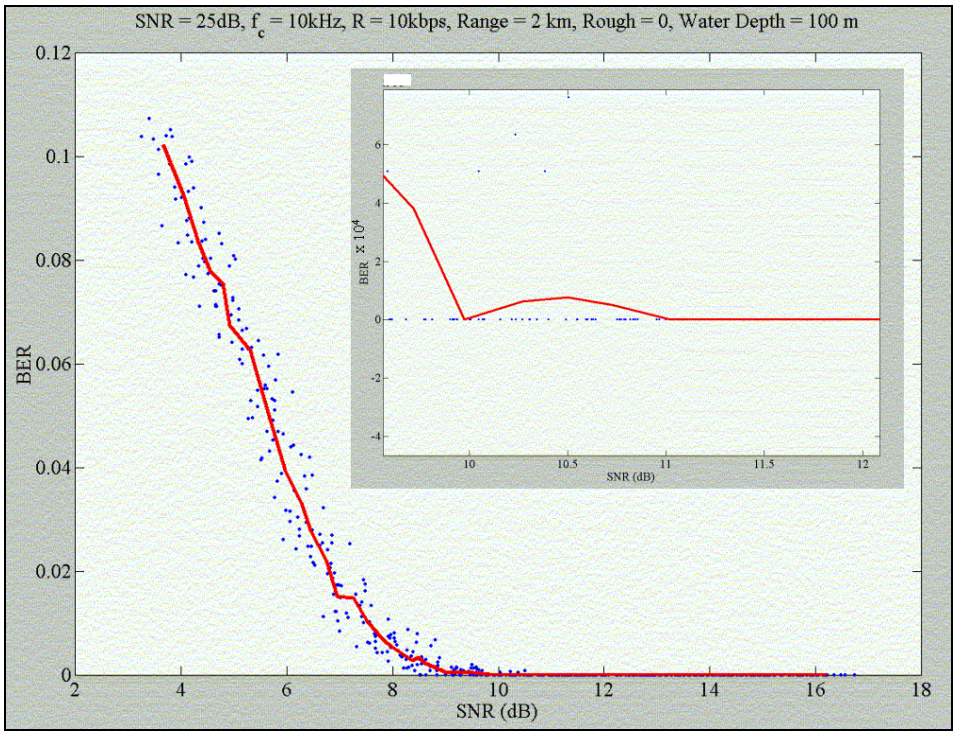


Figure A.23. BER vs SNR for 32 QAM, fc = 10 kHz, R = 10 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.

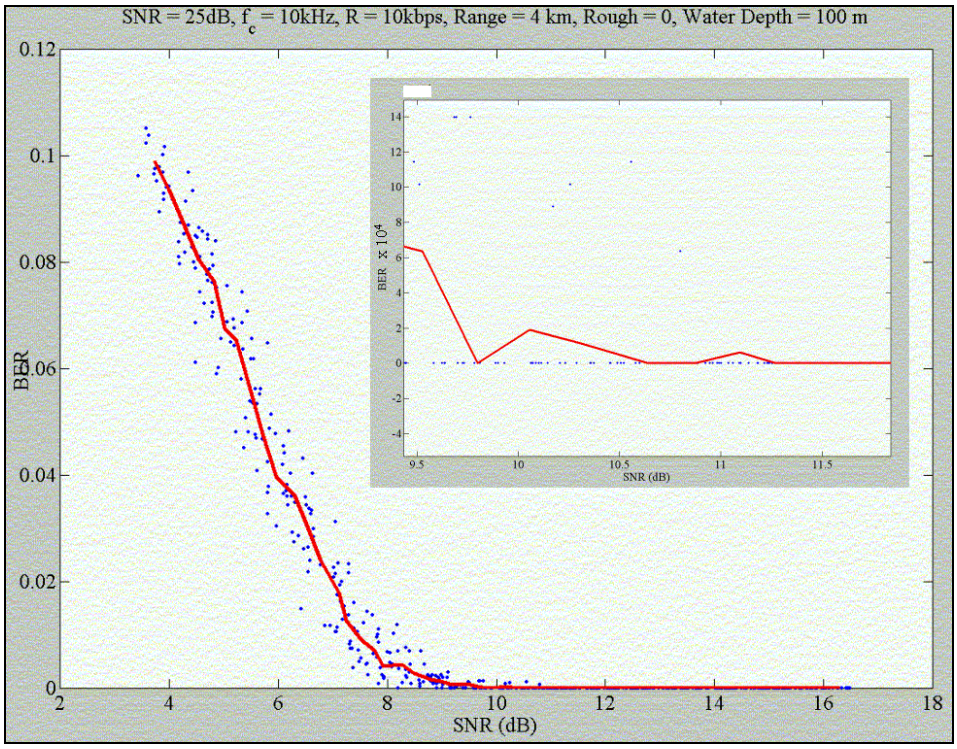


Figure A.24. BER vs SNR for 32 QAM, fc = 10 kHz, R = 10 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.

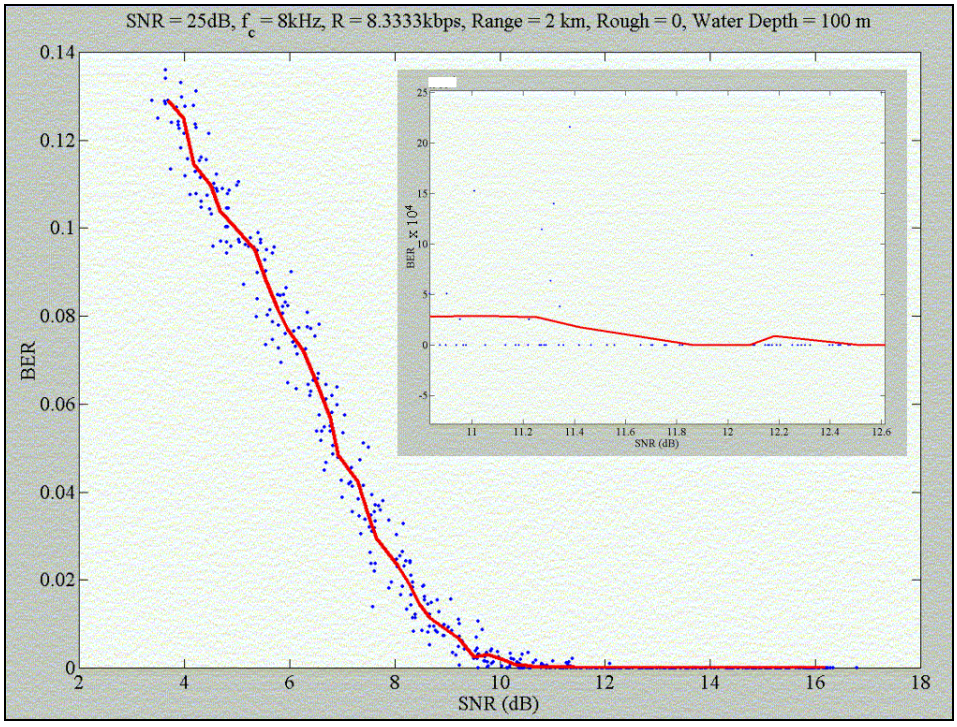


Figure A.25. BER vs SNR for 32 QAM, fc = 8 kHz, R = 8.33 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.

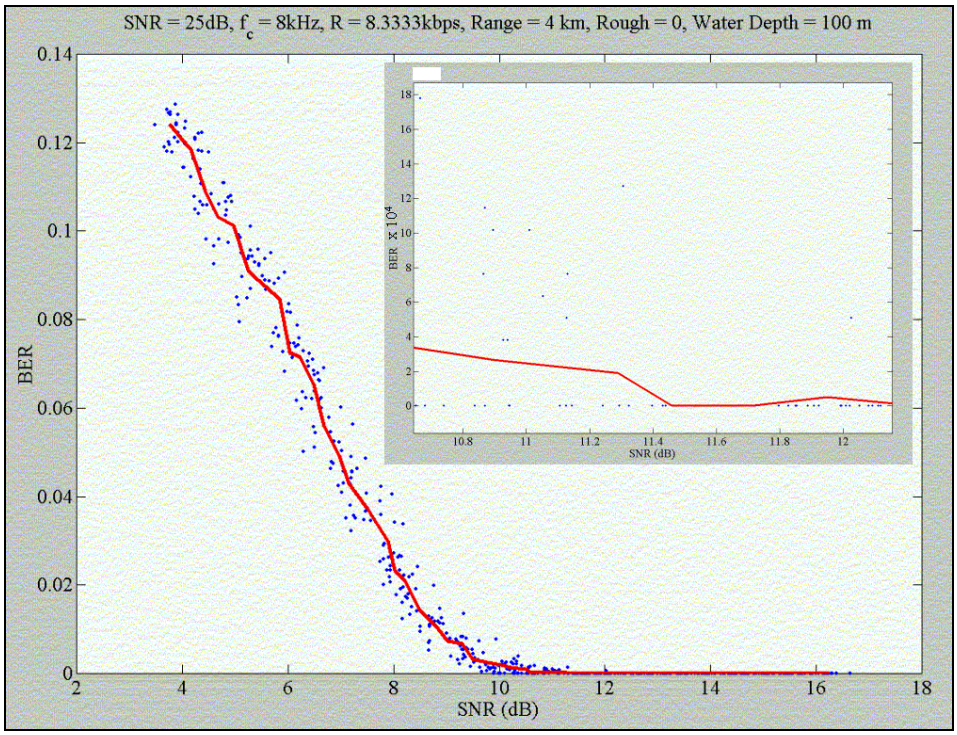


Figure A.26. BER vs SNR for 32 QAM, fc = 8 kHz, R = 8.33 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.

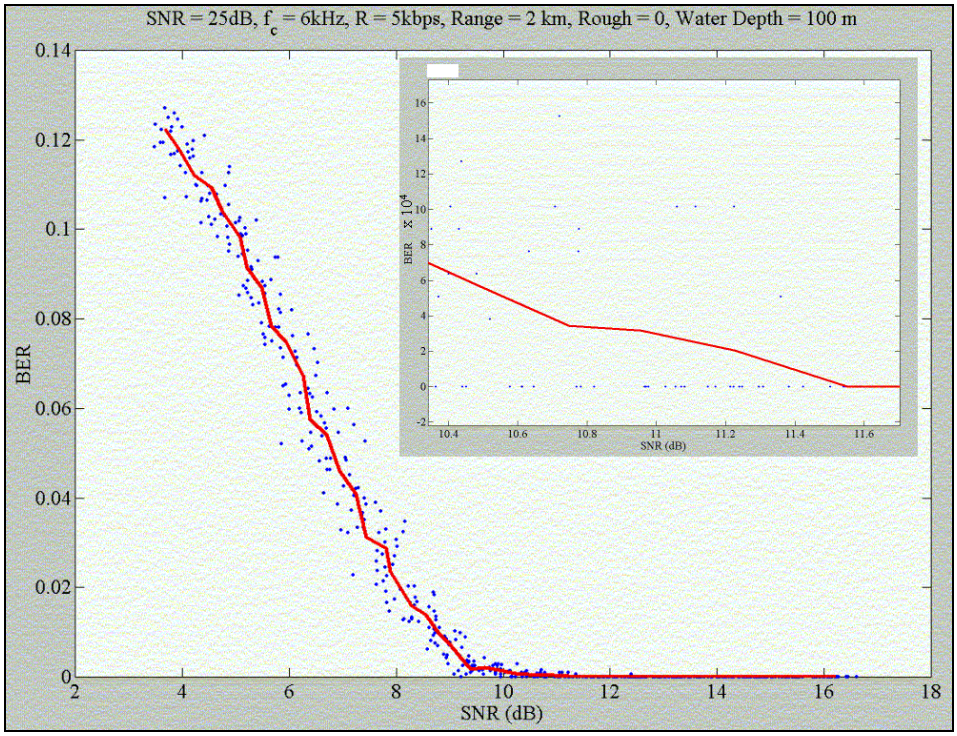


Figure A.27. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.

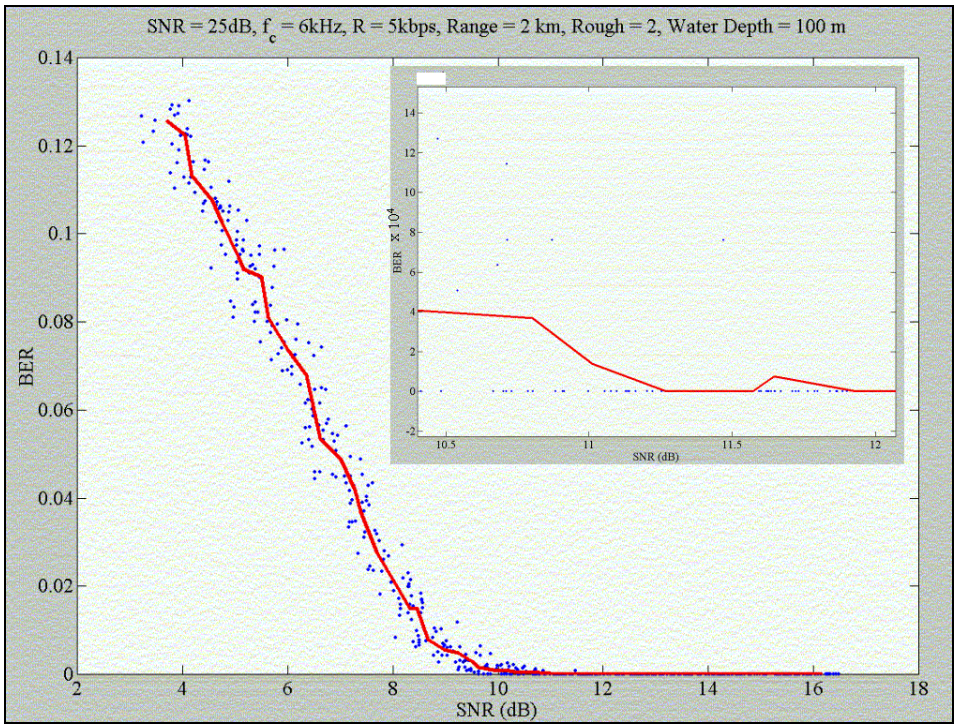


Figure A.28. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 2m, Water Depth = 100m.

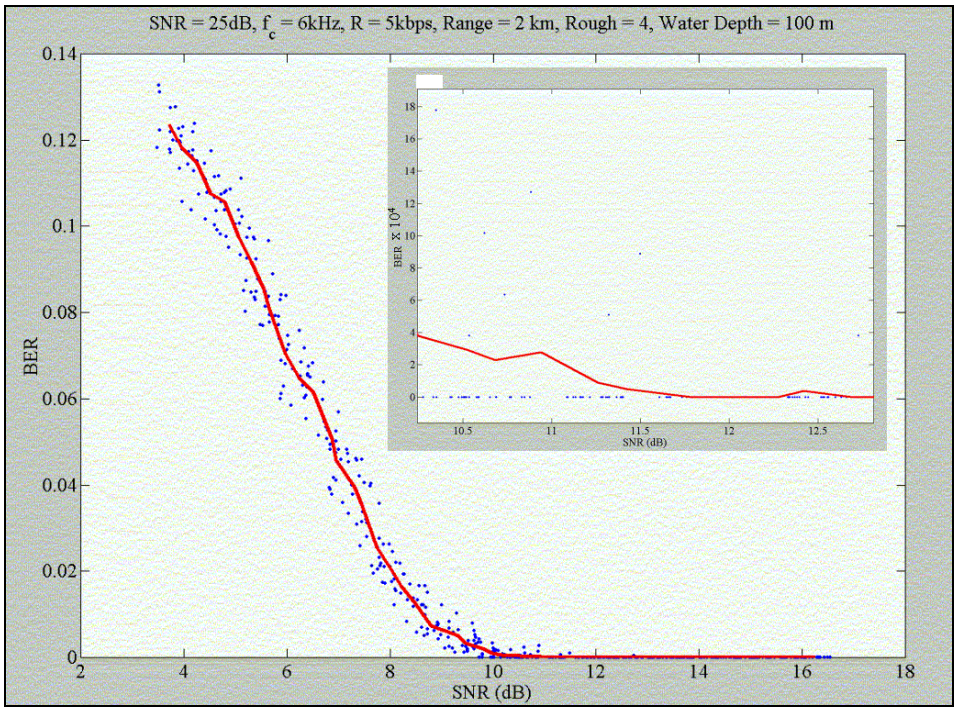


Figure A.29. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 4m, Water Depth = 100m.

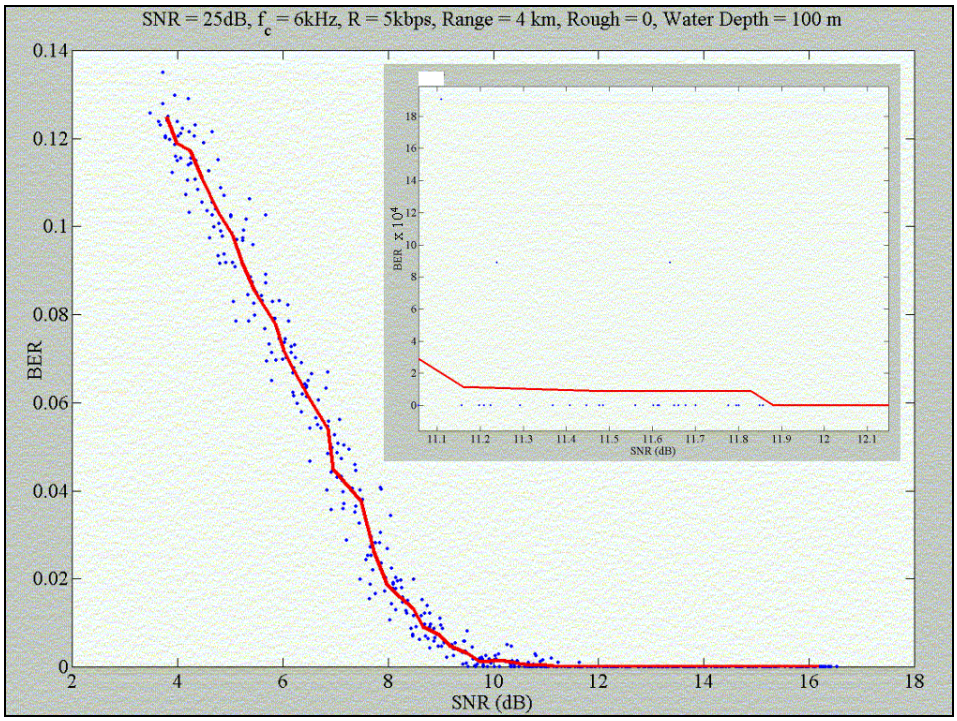


Figure A.30. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.

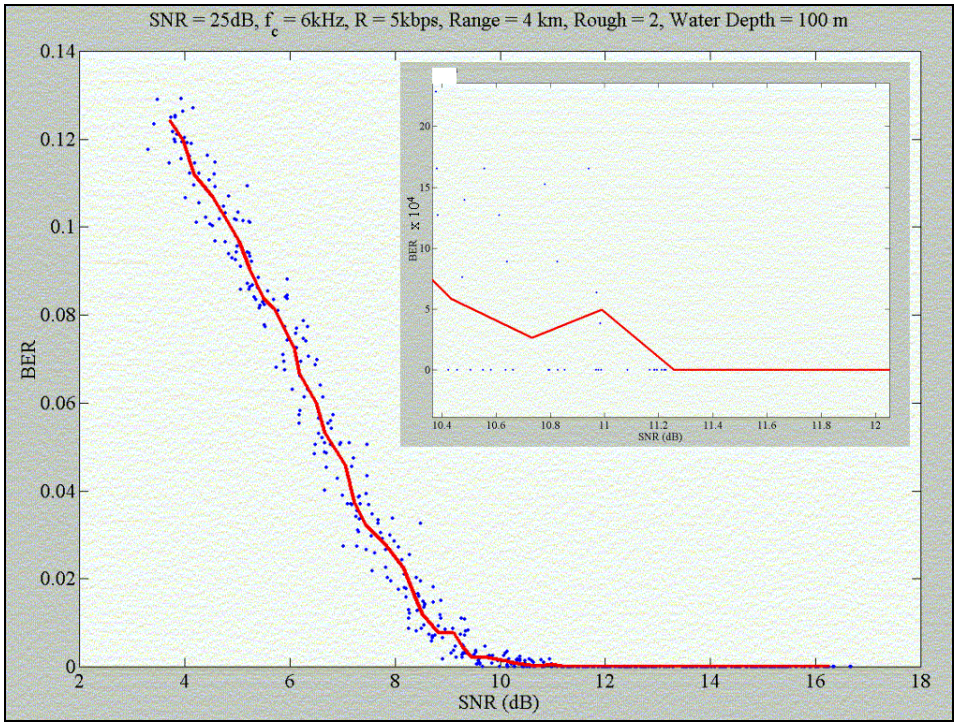


Figure A.31. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 2m, Water Depth = 100m.

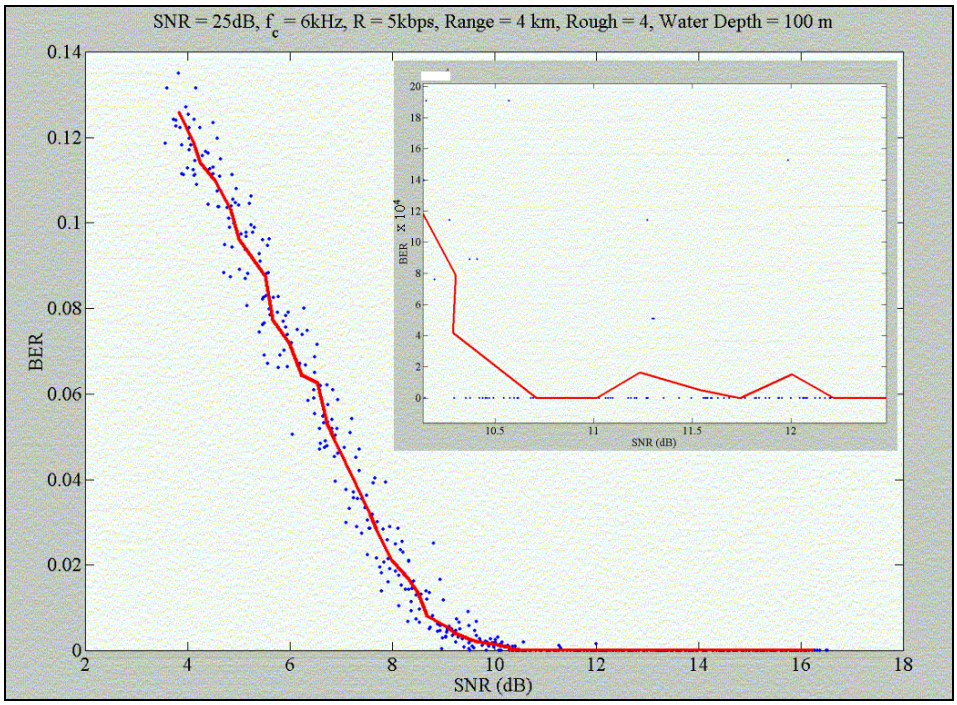


Figure A.32. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 4m, Water Depth = 100m.

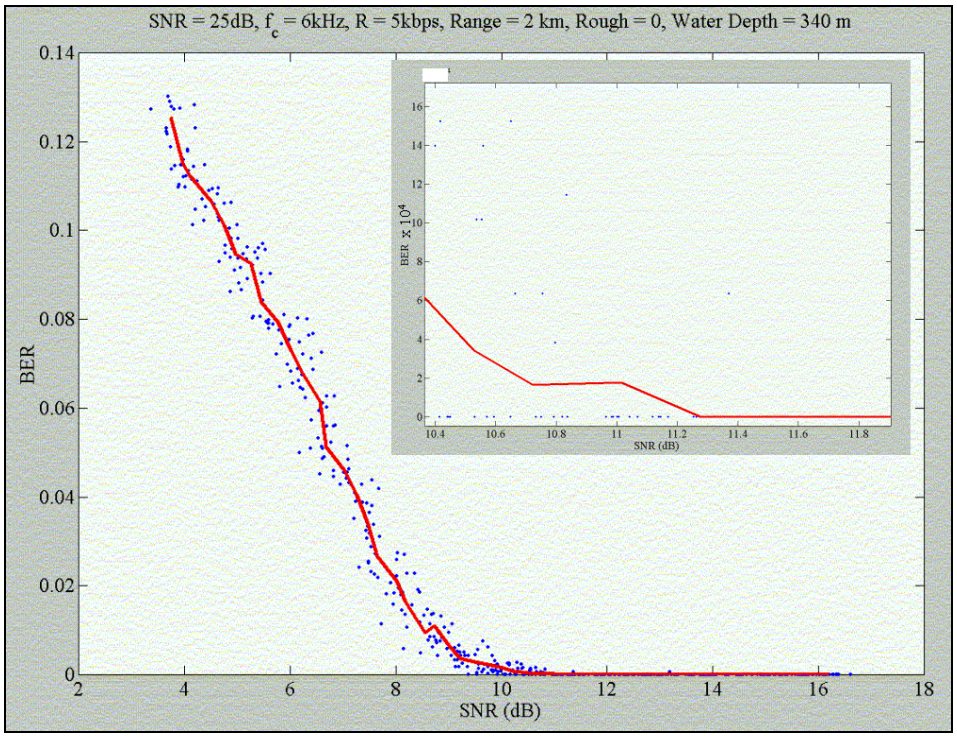


Figure A.33. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 340m.

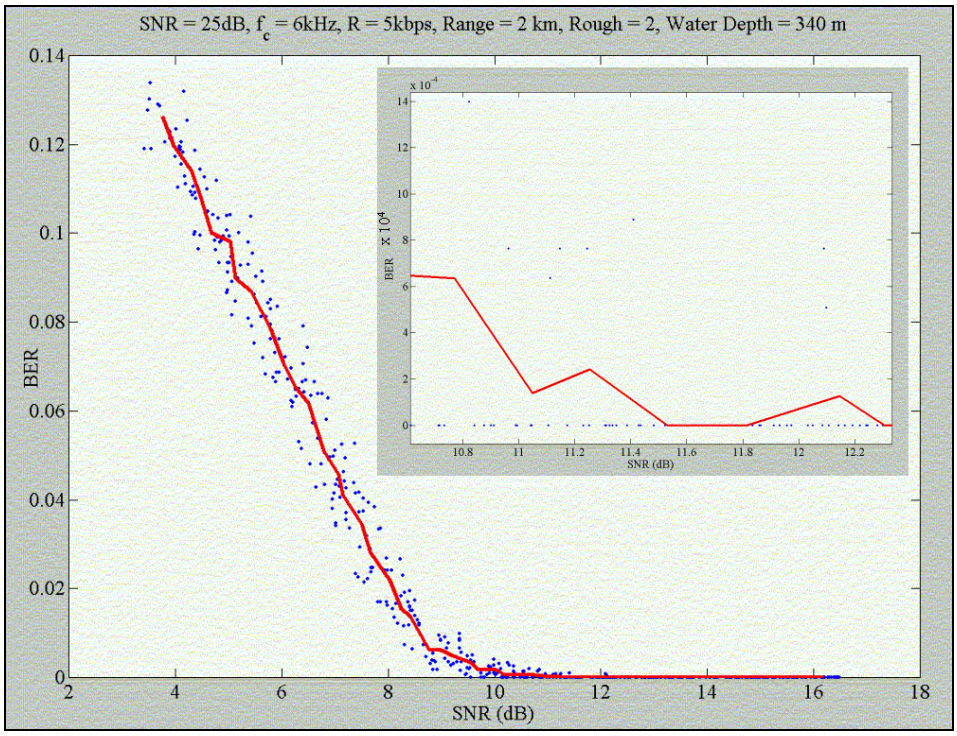


Figure A.34. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 2m, Water Depth = 340m.

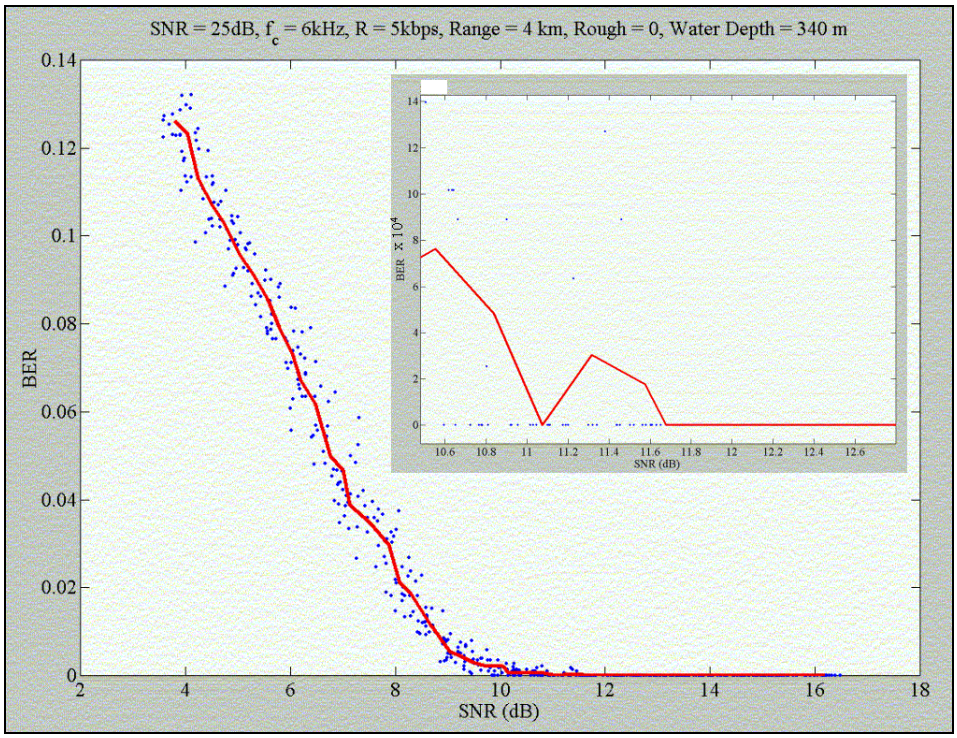


Figure A.35. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 0m, Water Depth = 340m.

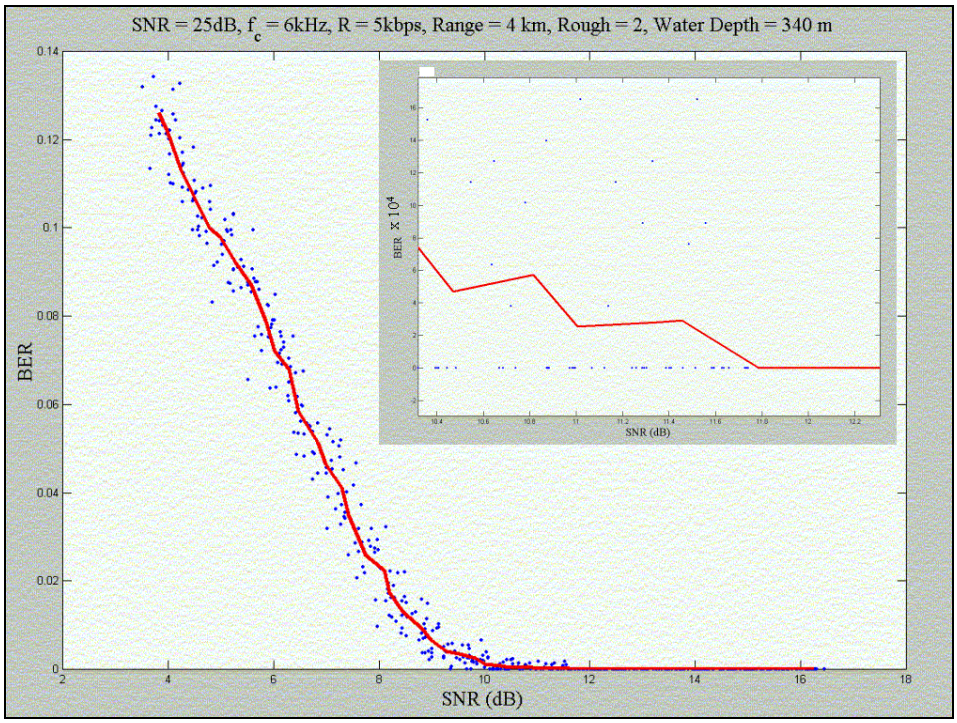


Figure A.36. BER vs SNR for 32 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 2m, Water Depth = 340m.

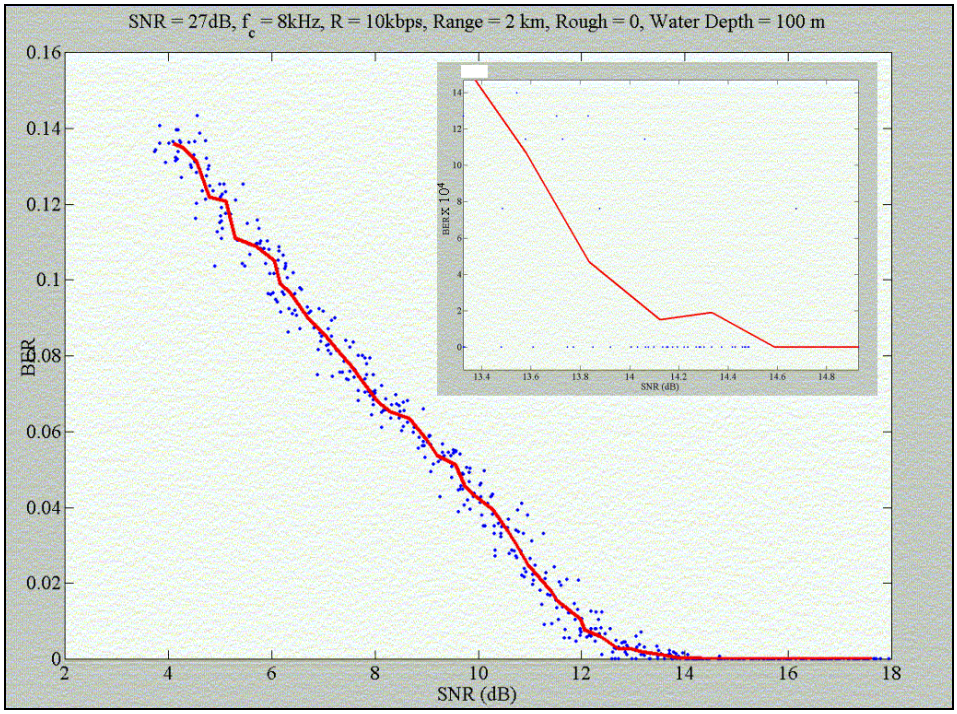


Figure A.37. BER vs SNR for 64 QAM, fc = 8 kHz, R = 10 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.

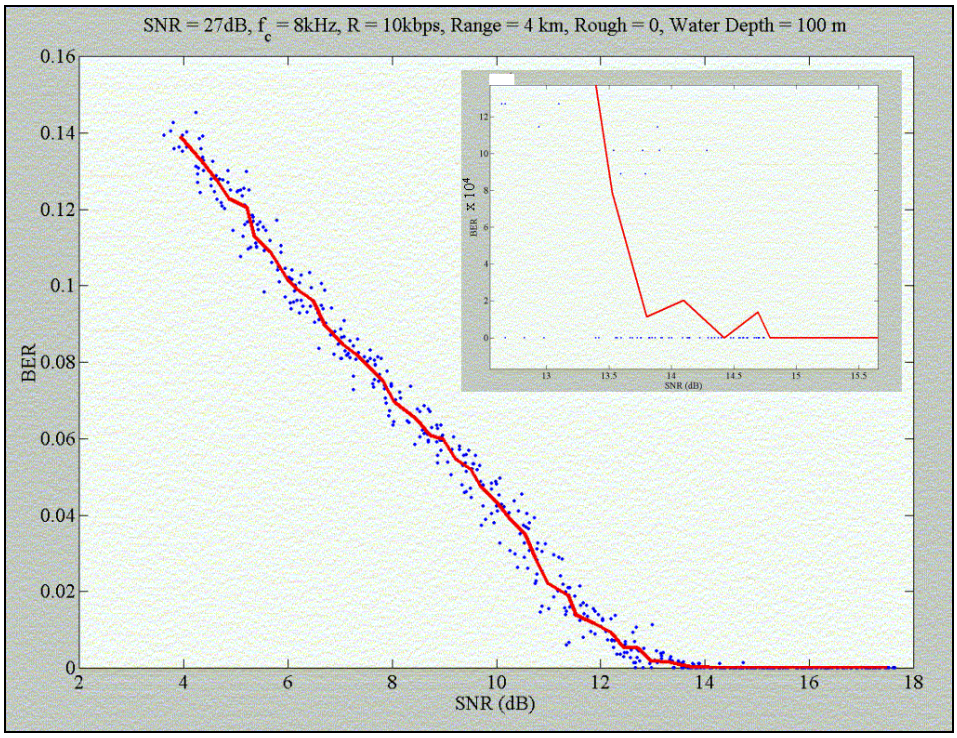


Figure A.38. BER vs SNR for 64 QAM, fc = 8 kHz, R = 10 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.

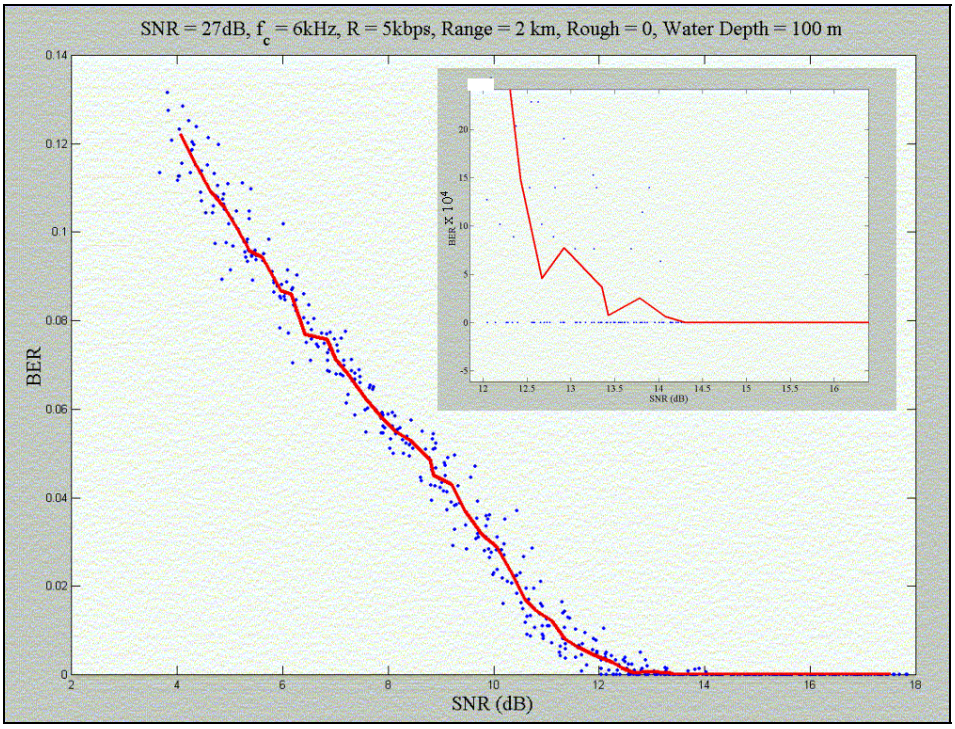


Figure A.39. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.

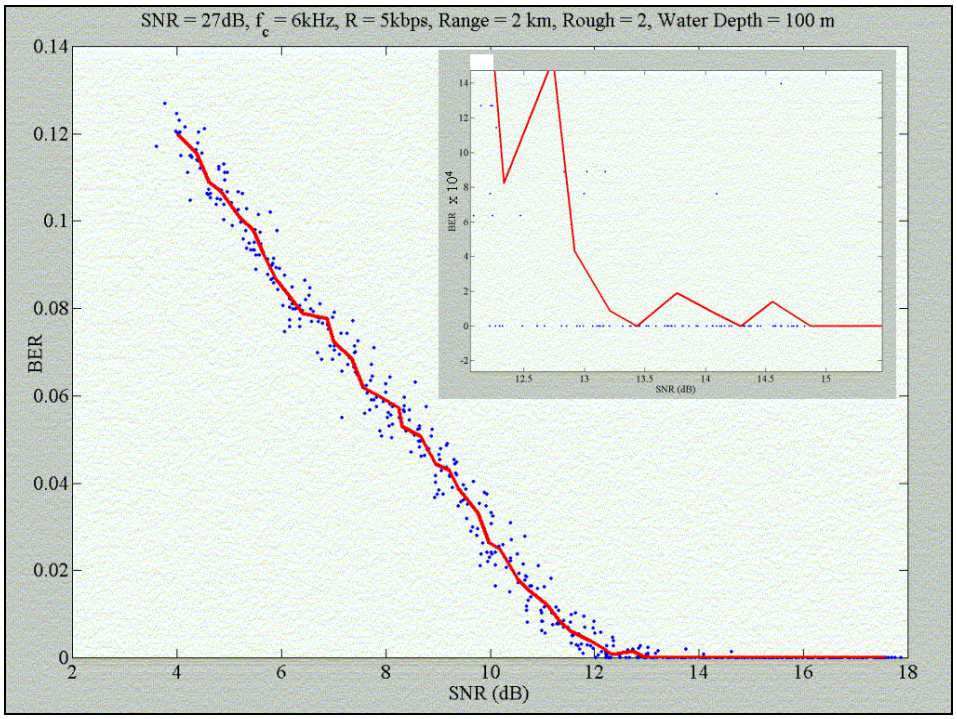


Figure A.40. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.

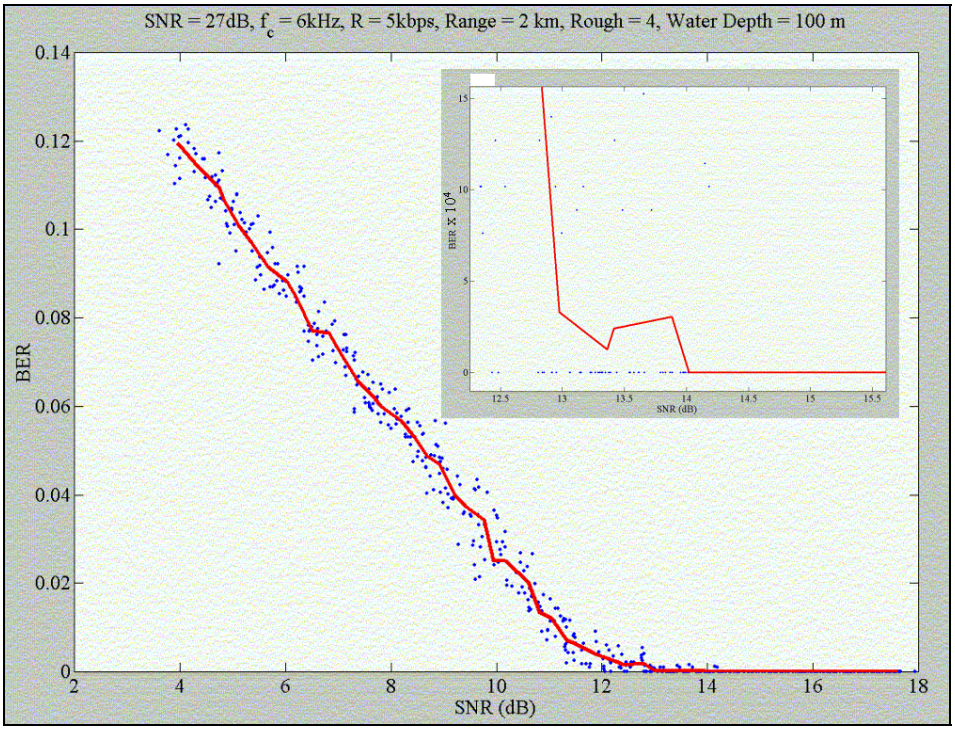


Figure A.41. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 4m, Water Depth = 100m.

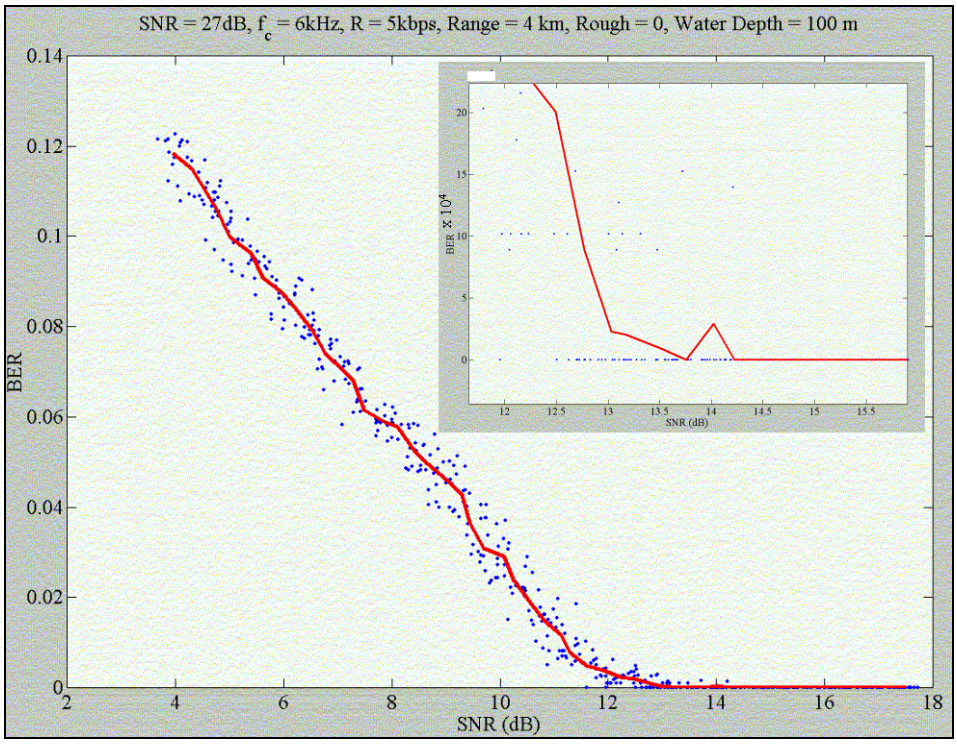


Figure A.42. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.

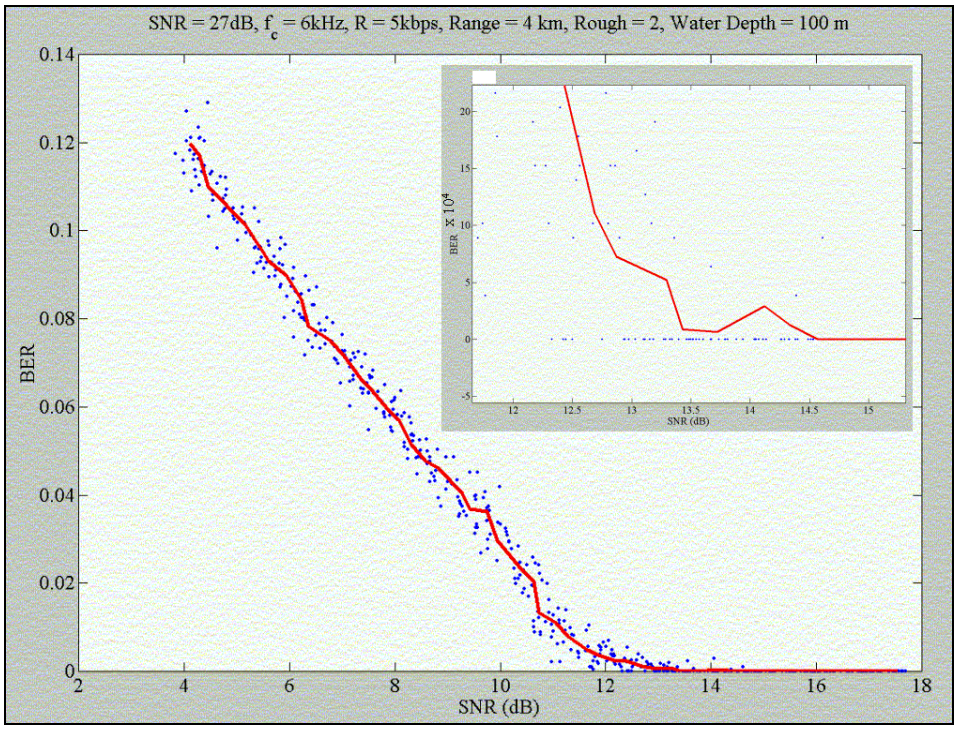


Figure A.43. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 2m, Water Depth = 100m.

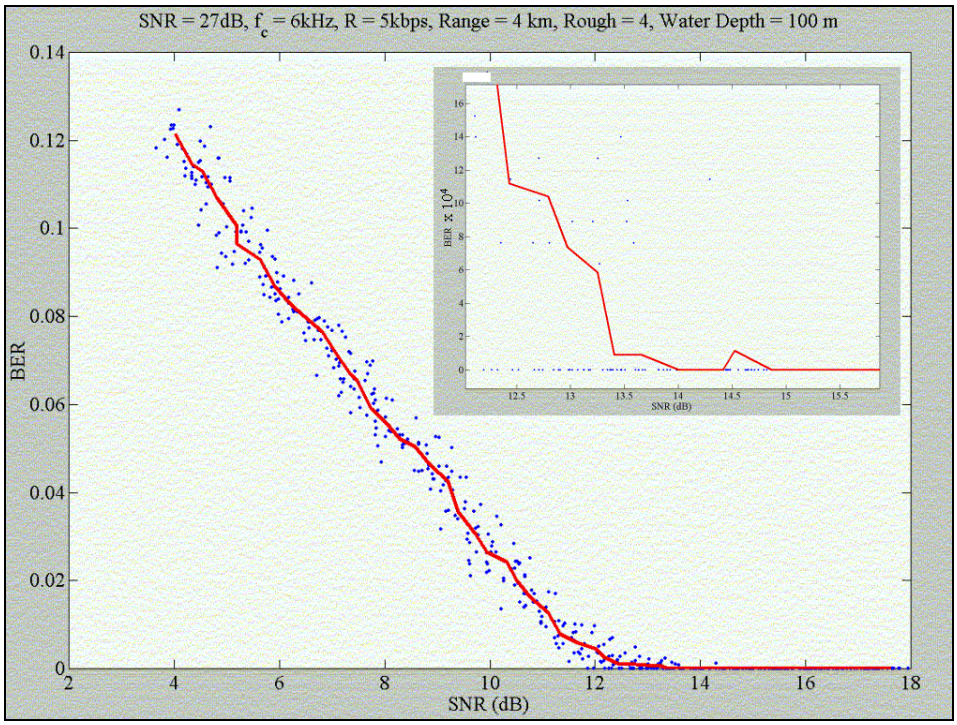


Figure A.44. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 4m, Water Depth = 100m.

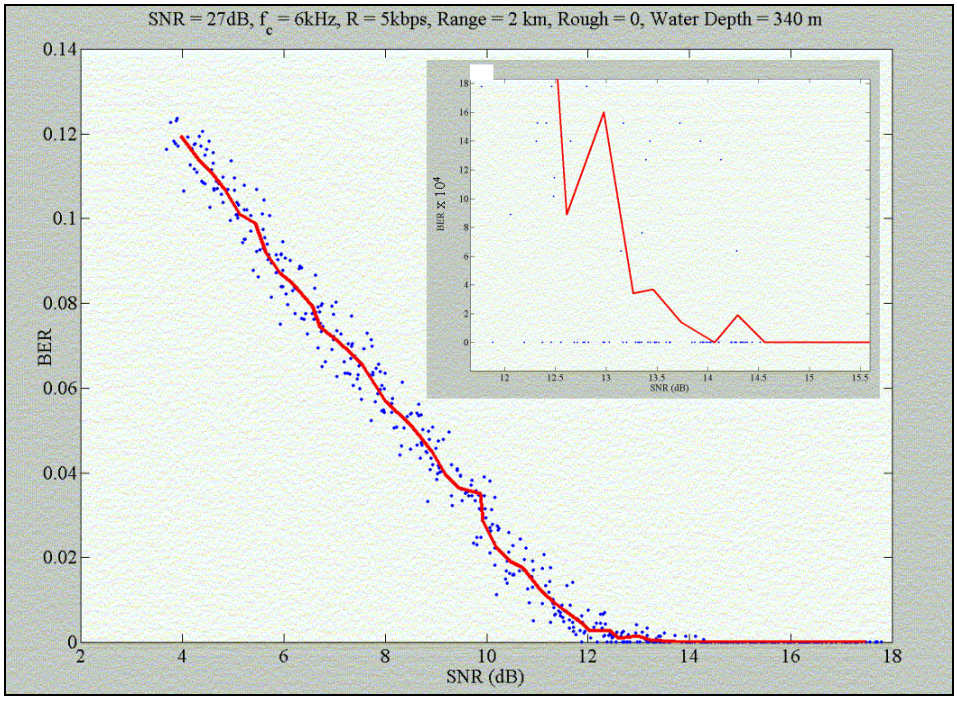


Figure A.45. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 0m, Water Depth = 100m.

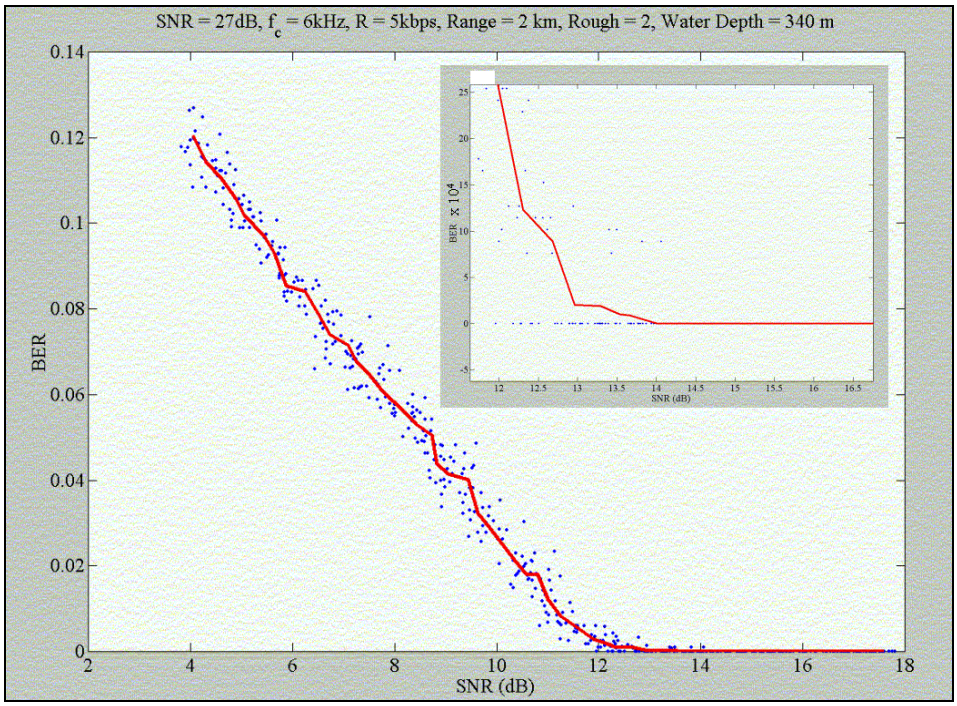


Figure A.46. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 2 km, Rough = 2m, Water Depth = 100m.

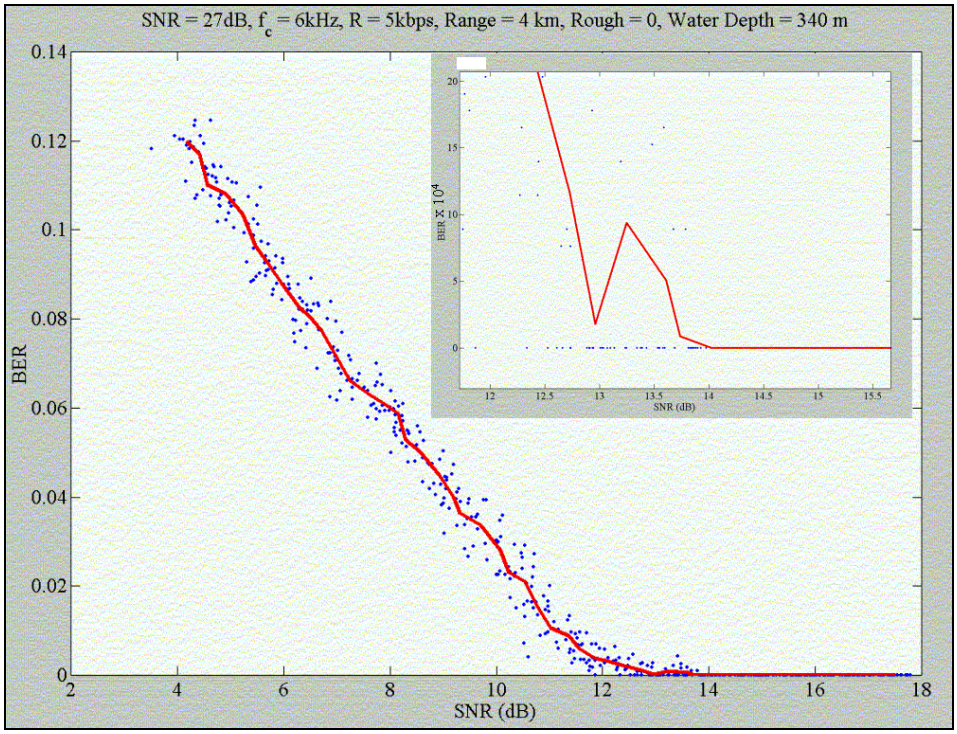


Figure A.47. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 0m, Water Depth = 100m.

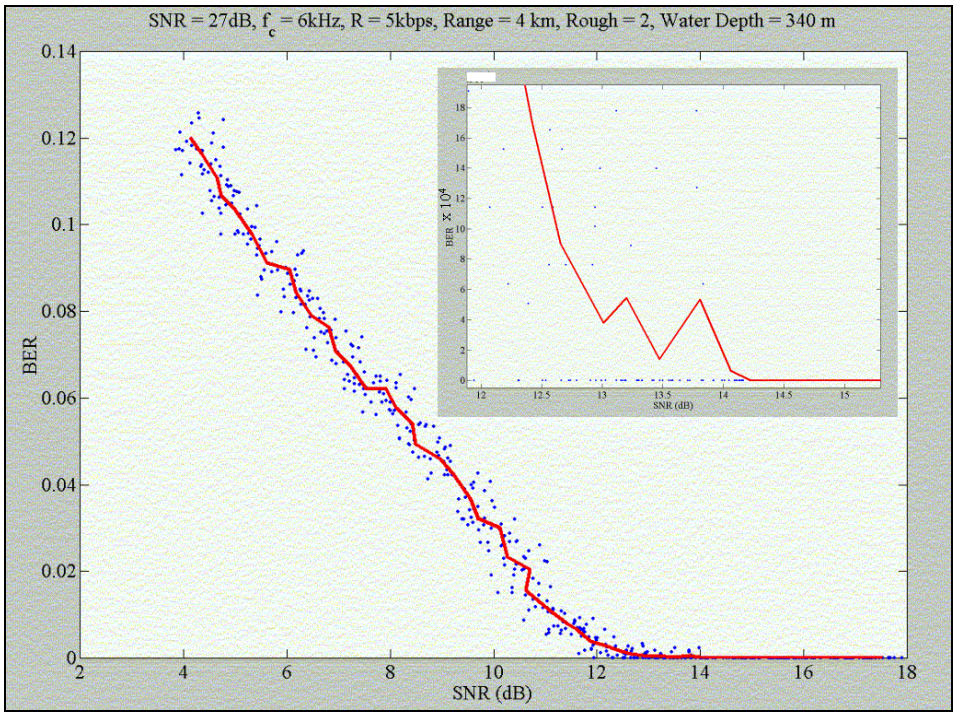


Figure A.48. BER vs SNR for 64 QAM, fc = 6 kHz, R = 5 kbps, Range = 4 km, Rough = 2m, Water Depth = 100m.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. MMPE RESULTS

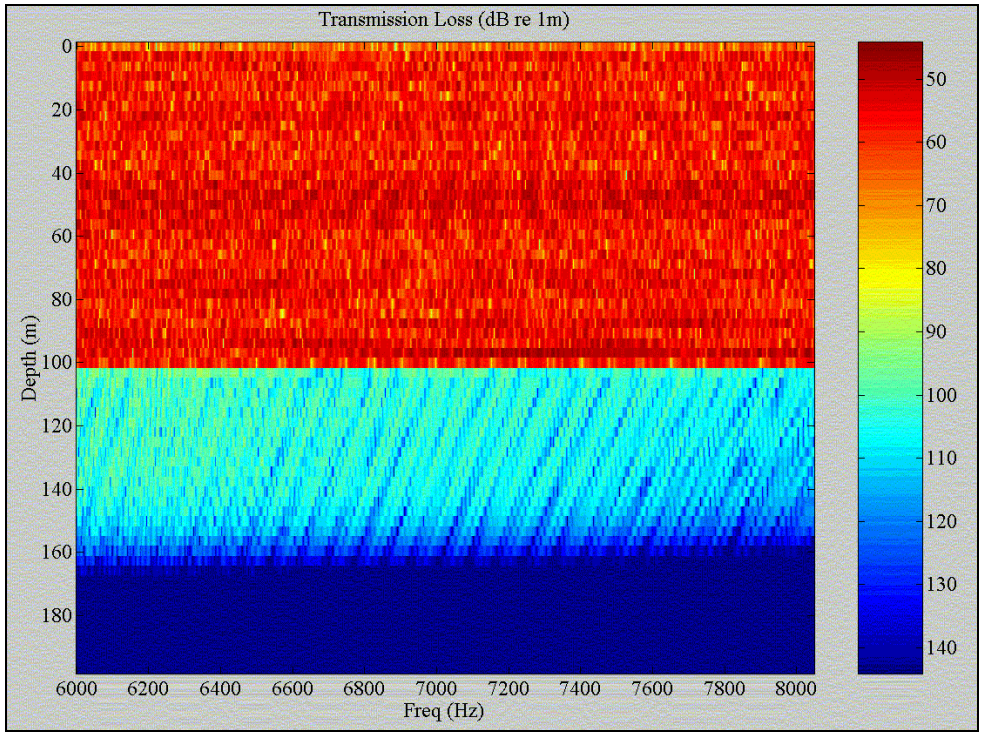


Figure B.1 Transmission Loss, Roughness 0 m, Range 2 km

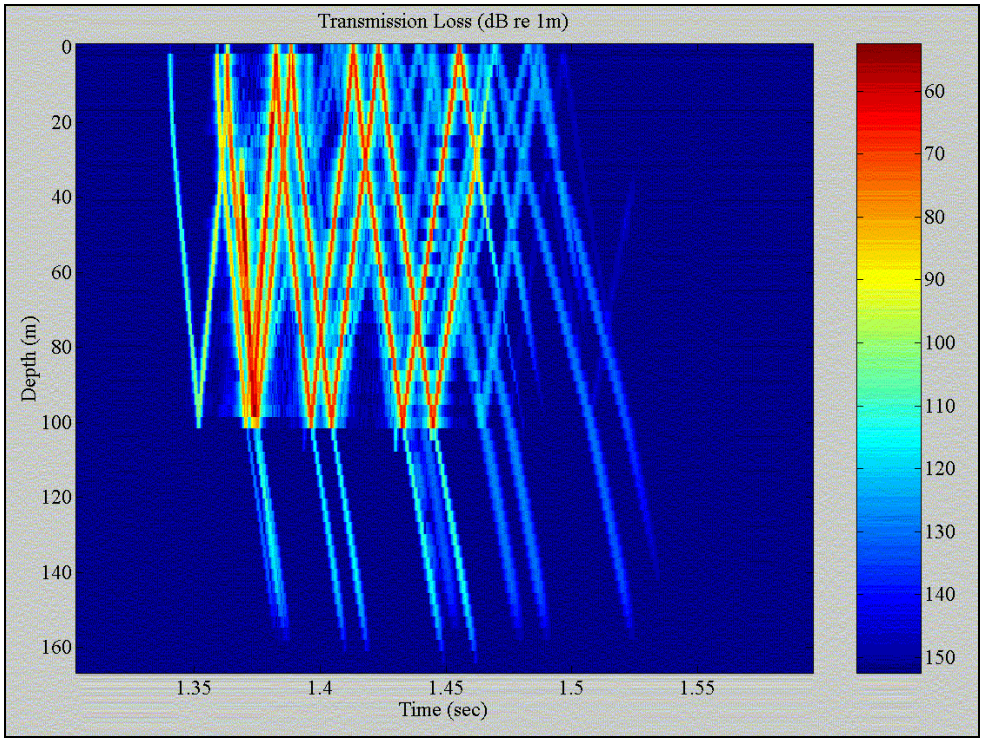


Figure B.2 Acoustic Paths, Roughness 0 m, Range 2 km, f_c 7 kHz

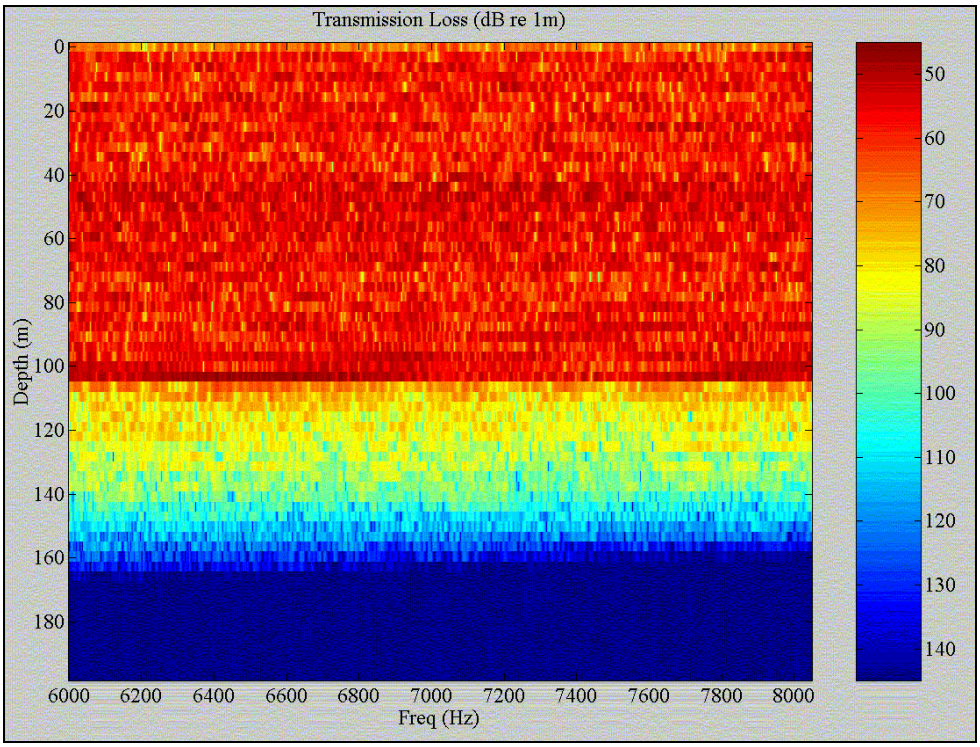


Figure B.3 Transmission Loss, Roughness 2 m, Range 2 km

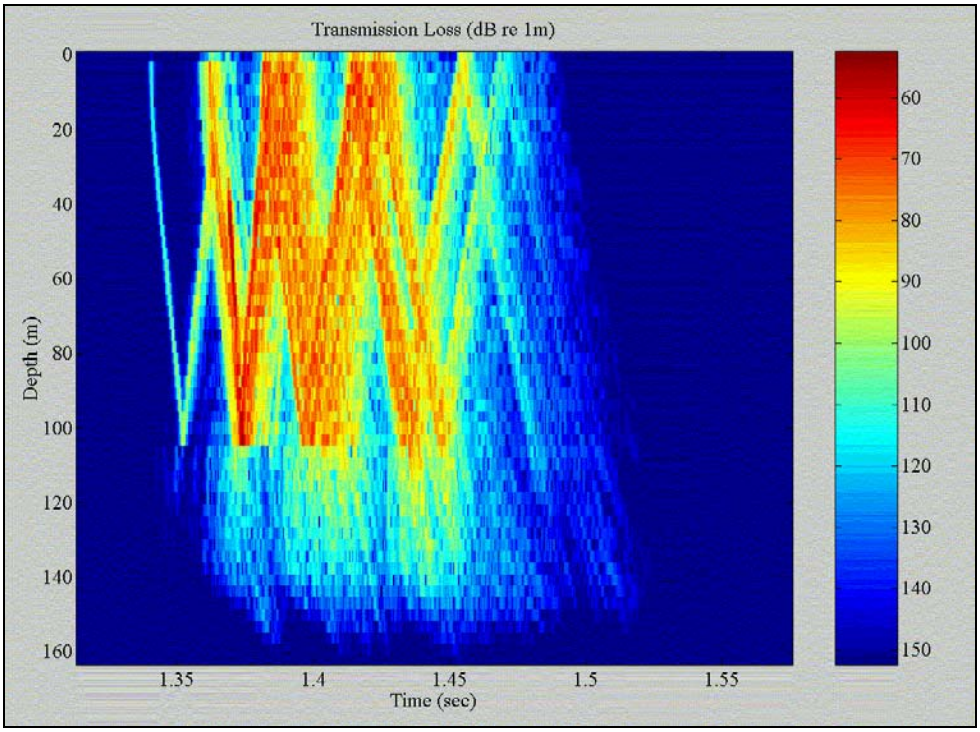


Figure B.4 Acoustic Paths, Roughness 2 m, Range 2 km, f_c 7 kHz

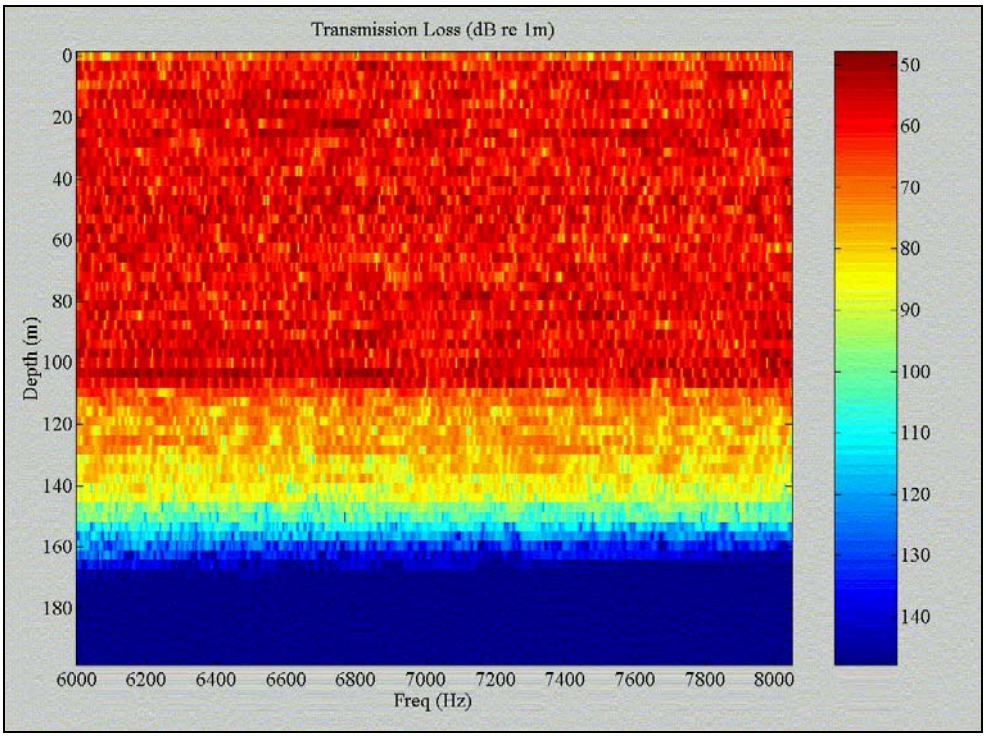


Figure B.5 Transmission Loss, Roughness 4 m, Range 2 km

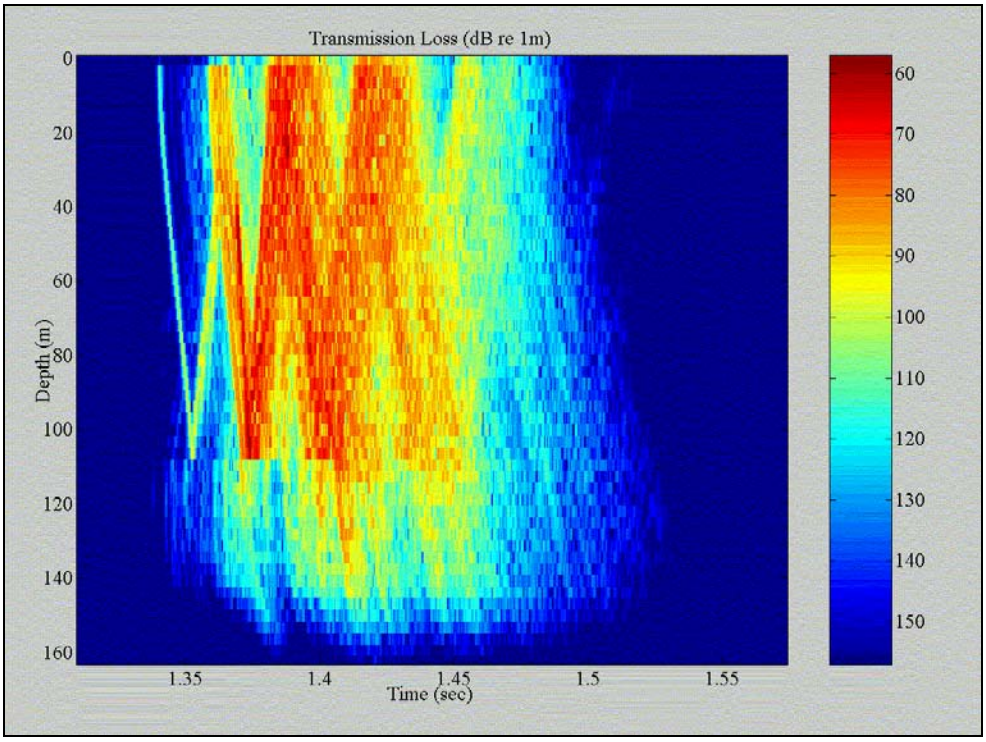


Figure B.6 Acoustic Paths, Roughness 4 m, Range 2 km, f_c 7 kHz

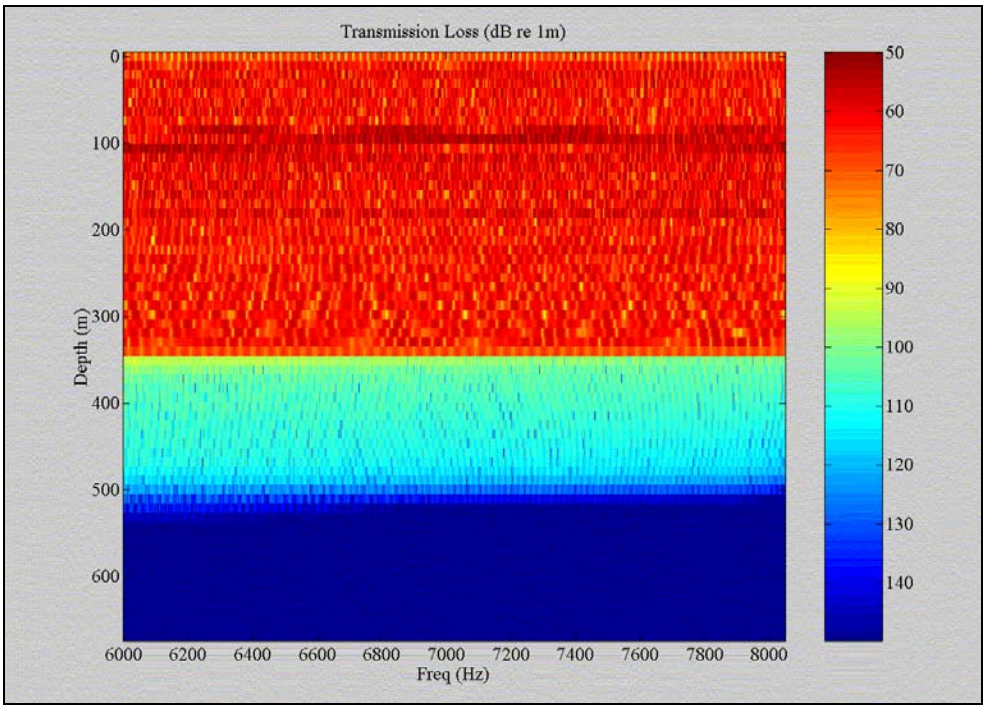


Figure B.7 Transmission Loss, Roughness 0 m, Range 2 km

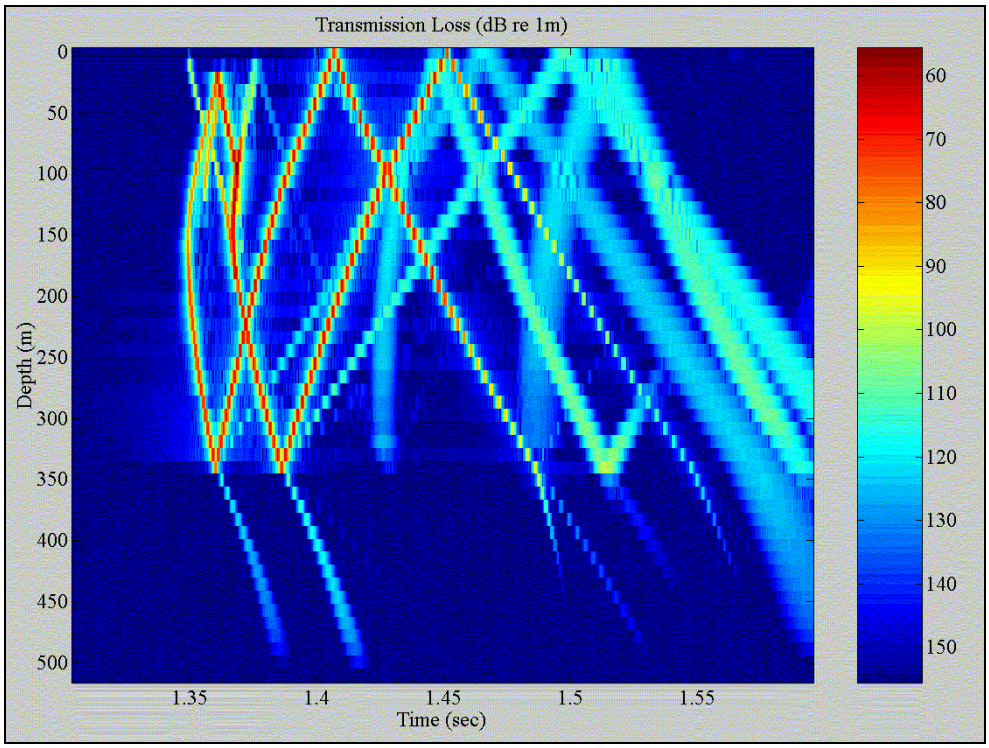


Figure B.8 Acoustic Paths, Roughness 0 m, Range 2 km, f_c 7 kHz

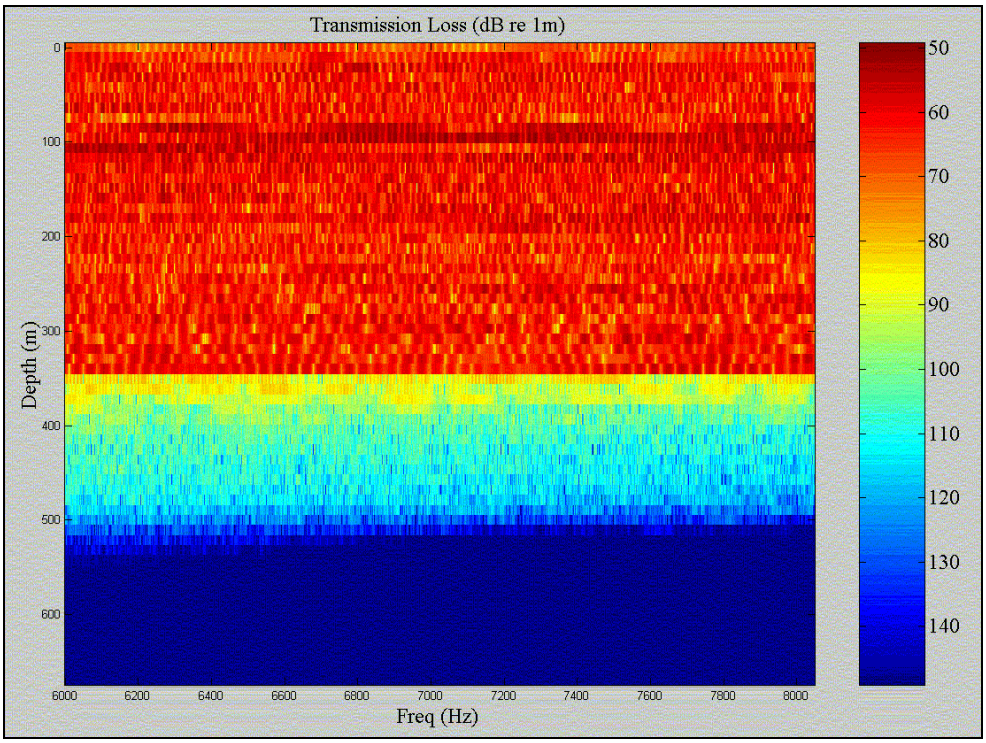


Figure B.9 Transmission Loss, Roughness 2 m, Range 2 km

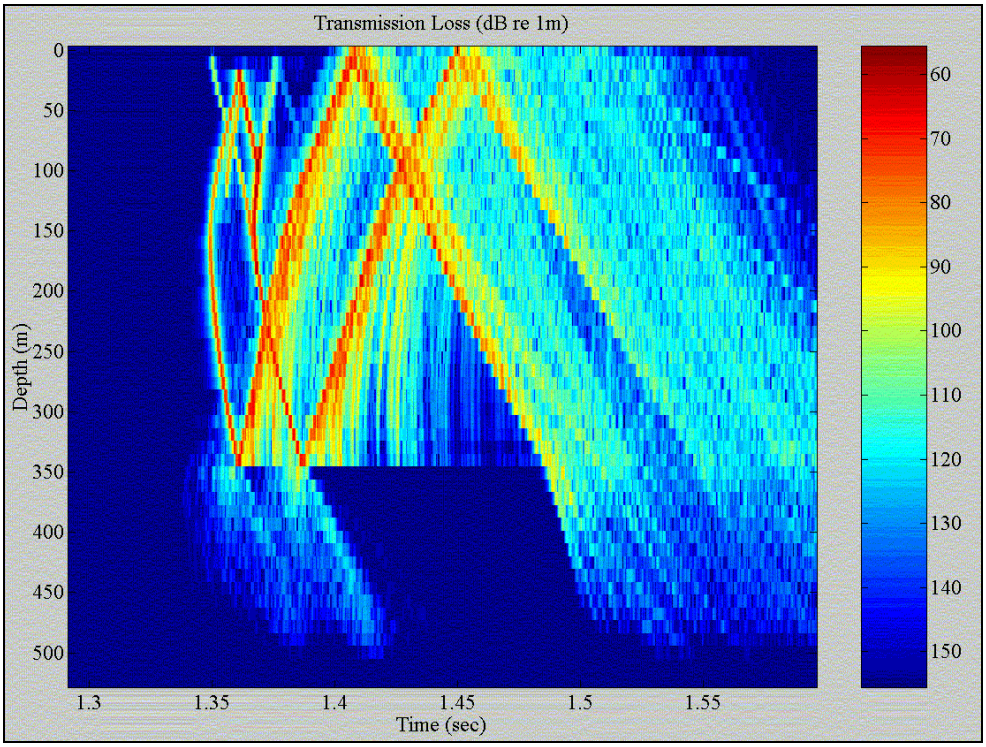


Figure B.10 Acoustic Paths, Roughness 0 m, Range 2 km, f_c 7 kHz

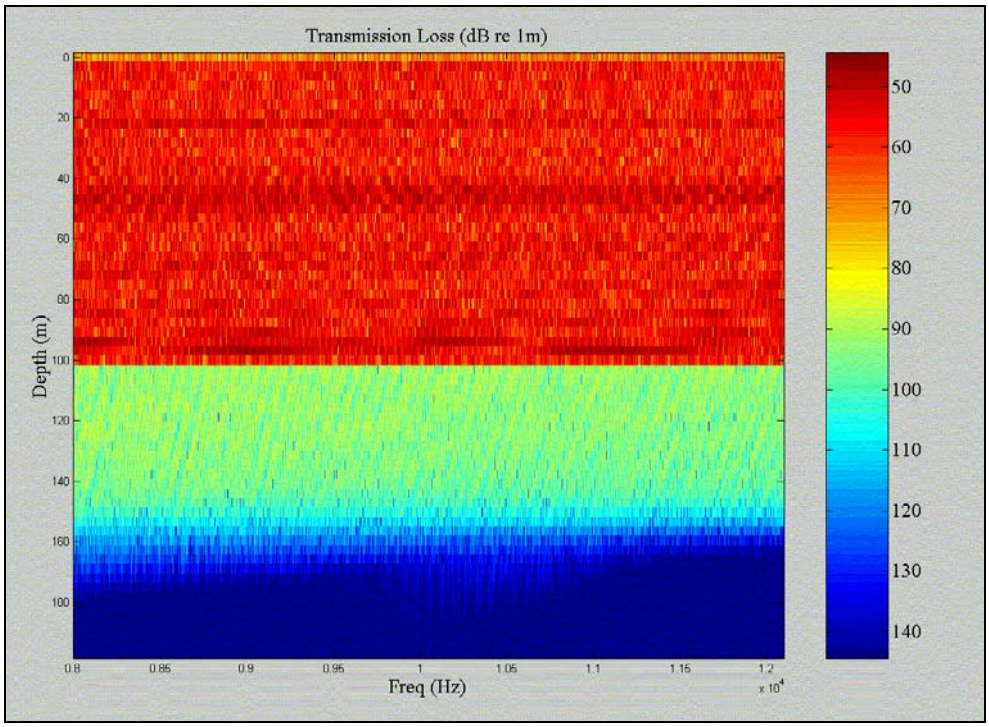


Figure B.11 Transmission Loss, Roughness 0 m, Range 2 km

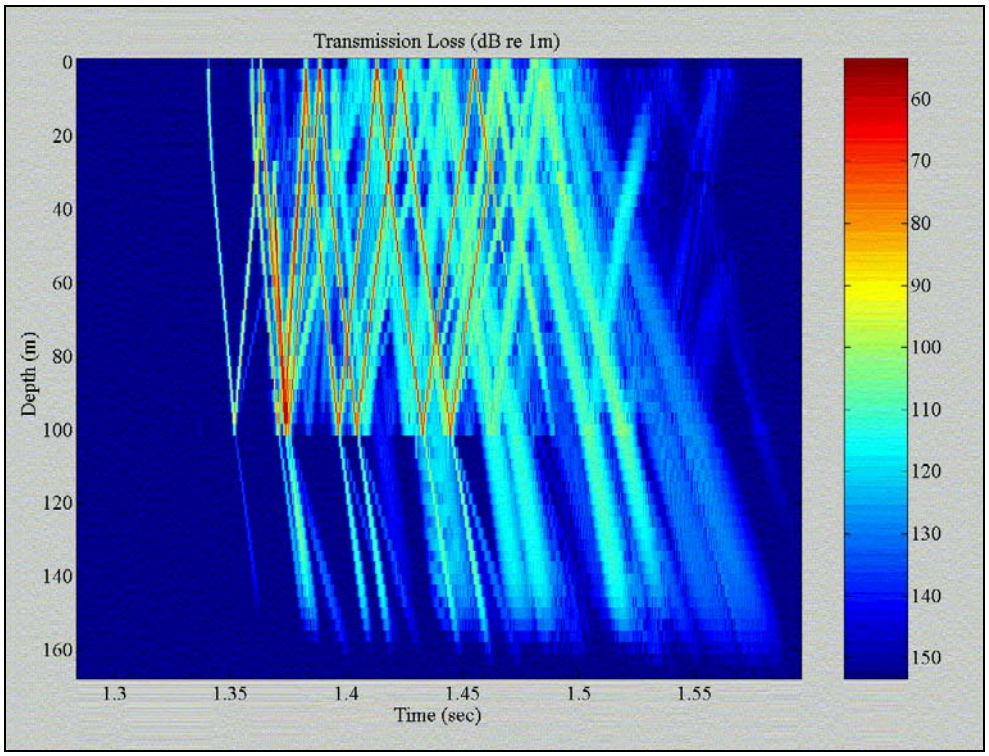


Figure B.12 Acoustic Paths, Roughness 0 m, Range 2 km, f_c 10 kHz

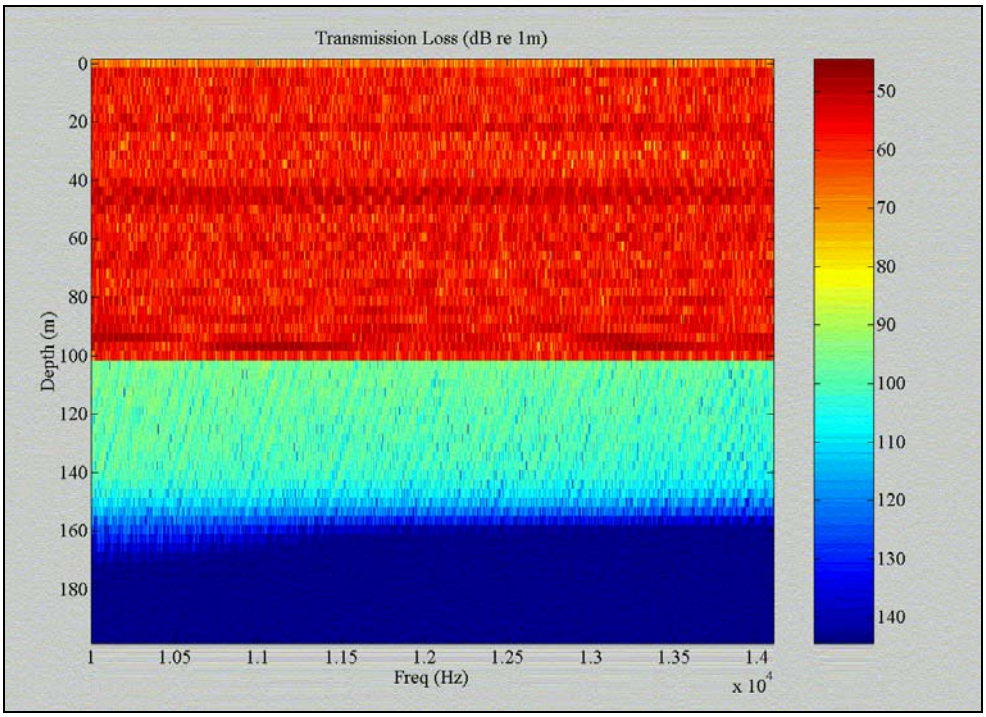


Figure B.13 Transmission Loss, Roughness 0 m, Range 2 km

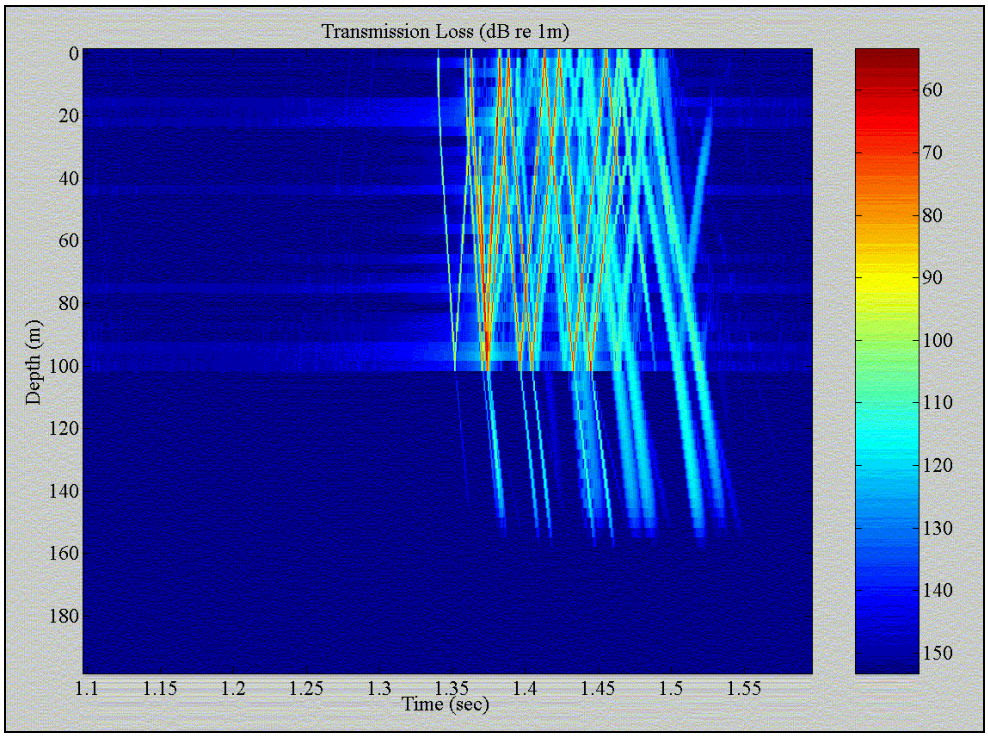


Figure B.14 Acoustic Paths, Roughness 0 m, Range 2 km, f_c 12 kHz

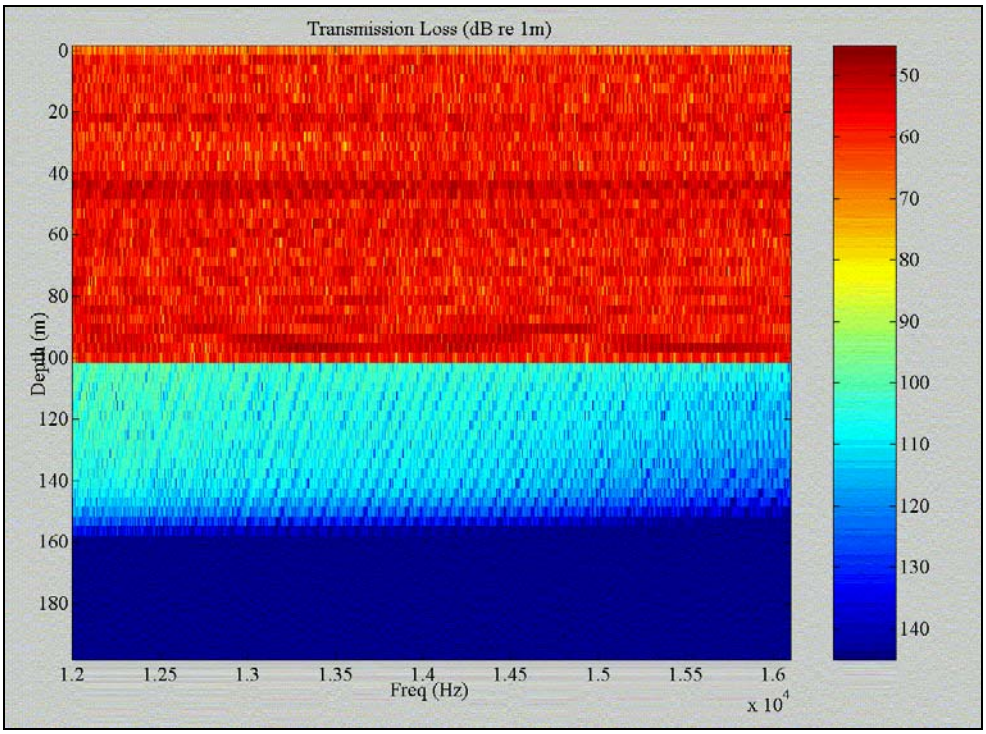


Figure B.15 Transmission Loss, Roughness 0 m, Range 2 km

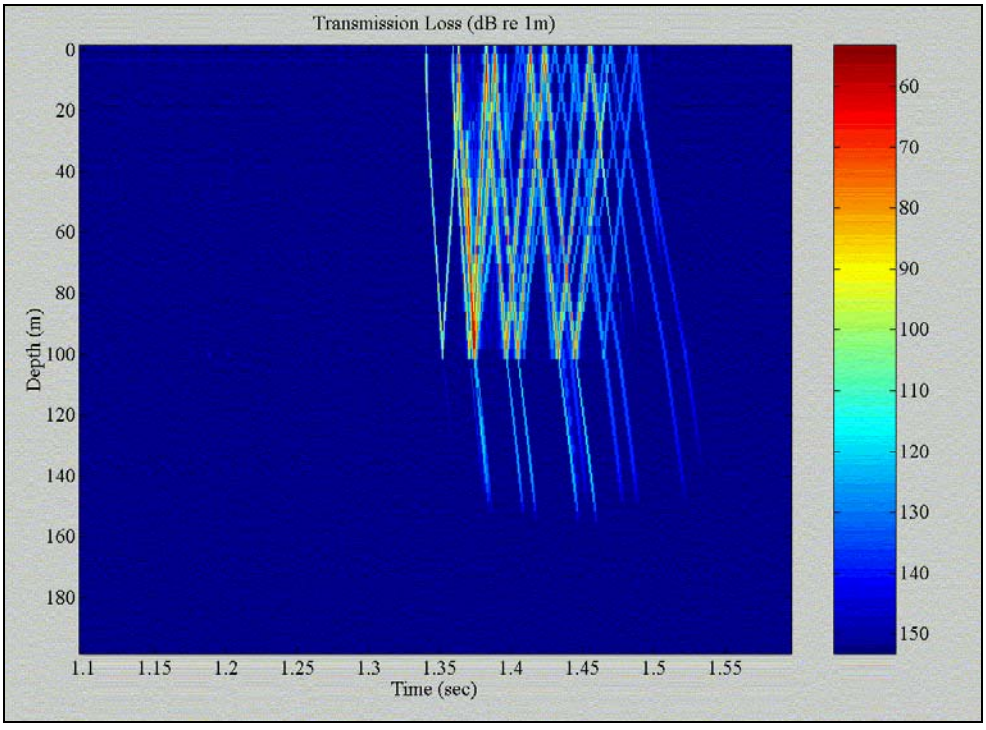


Figure B.16 Acoustic Paths, Roughness 0 m, Range 2 km, f_c 14 kHz

APPENDIX C. MATLAB CODE

A. TRANSMITTER CODE

1. ofdm_sim_xmitter.m

```
%LT Tiger Pittman
%UWA OFDM Thesis m-file
%Last modified 4/16/01
clear all, close all
fn = 0;            %running figure number index
synchdec = 0;
%logic variable for synchronization, 0 == perfect synchronization,
%1 == synchronization using cyclic prefix correlator
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
zero_fraction = .0250;          %fraction of subcarriers that
%carry zeroes, near Fs/2 to avoid pseudo-aliasing effects
[N_bar, N_subc_act, N_zerc, guard_time, T, T_fft, Fs_fft, zero_fraction, W, fc, m, SNR, delay_spread] = parameters(zero_fraction);
%takes input from user to determine key parameters in the OFDM system
%including the number of active subcarriers, guard time(correlates to cyclic prefix length)
%OFDM symbol period(T, includes the FFT/IFFT interval and the guard time),
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
noise = 1;
%logic variable for noise addition
noise_factor = 0.35;
%if noise == 1 then AWGN is added with following SNR to received signal
%otherwise no noise is added
SNR_awgn = SNR*noise_factor;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
modulate = 2;
%if modulate = 2 then double side band modulation is performed, otherwise
%no modulation is performed
%%%
```

```
%Conversion (quantized to binary)
bit_matrix = binary_conversion(quant_sig, bits);
%pts by bits matrix of binary data
%%%
```

```
qam_coded_dec = qam_buffer(bit_stream, q);
%buffers bit_stream to q column matrix and converts
%to decimal format vector with values 0 to Q-1 of length
%ceil(length(bit_stream)/q)
%%%
```

```
%%%
```

```
end
```

2. parameters.m

```
function [N_bar, N_subc_act, N_zerc, guard_time, T, T_fft, Fs_fft, zero_fraction, W, fc, m, SNR, delay_spread] = parameters(zero_fraction)
%This function uses input data to determine key parameters in the OFDM signal generation.
%Last modified 7/3/01.
R = 1000*(5.0+0/3);%input('What is the desired data rate (kbps)?  ');
W_max = 1000*7;
fc = 1000*6;
delay_spread = 1e-3*25;
Pe = .0001;
%one dimensional symbol error probability, 2 D symbol error probability = 2*Pe
SNR_gap3 = 20*log10(sqrt(2)*erfinv(1-Pe))
SNR_gap = SNR_gap3 - 10*log10(3)
SNR = 21;
b_Pe = log2(1 + 10^(SNR/10)/10^(SNR_gap/10));
%bits per subsymbol that can be carried at given Pe, SNR_gap, coding gain = margin dB and SNR
guard_time = 4*delay_spread;
%guard time required to prevent ISI (sec)
T  = 6*guard_time;
%OFDM symbol period including the cyclic prefix
Rs = 1/T;
%OFDM symbol rate (symbols/sec)
b_ofdm = R*T;
%bits per OFDM symbol required to get desired rate
N_subc_act = ceil(round(b_ofdm)/floor(b_Pe));
%number of subcarriers required for desired data rate at achievable b_Pe
%these are data carrying subcarriers
power2 = 2^ceil(log2(N_subc_act));
%determines the next power of 2 above N_subc_act
```

```
if (((power2-N_subc_act)/power2) >= zero_fraction)
  N_bar = power2;
else
  N_bar = 2^ceil(log2(N_subc_act) + 1);
end
N_zerc = N_bar - N_subc_act;
%number of zero subcarriers
zero_fraction = N_zerc/N_bar;
%fraction of subcarriers that carry zeroes
T_fft = T - guard_time;
%FFT/IFFT period
Fs_fft = N_bar/(T_fft);
%first guess at FFT sampling rate in FFT interval
%based on guard time = 4*expected delay time and an OFDM symbol period of
%6*guard time
%this is half the sampling rate required at the DAC and LPF since the
%IFFT doubles the number of samples thereby effectively interpolating by 2
%and doubling the sampling rate
n_samp = Fs_fft*T;
%number of samples in the OFDM symbol period based on Fs_fft
%must also be an integer to ensure that the subcarriers are orthogonal
if (isposint(n_samp)~=1)
  while (isposint(n_samp)~=1)
    Fs_fft = Fs_fft + 2;
    n_samp = Fs_fft*T;
  end
end
%revises the sampling rate slightly in the FFT interval such that there are an
integer
%number of samples in the OFDM symbol period
T_fft = N_bar/Fs_fft;
%revises the FFT/IFFT interval length based on the sampling rate
%such that there will be an integer number of samples in the interval
```

```
guard_time = T - T_fft;
%revises the guard time based on the new T_fft and the known T
format long e
SC_del_f = (T-guard_time)^-1
%subcarrier spacing (Hz)
format short e
W_mmpe = SC_del_f*N_bar
W = SC_del_f*N_subc_act;
%bandwidth in Hz
if (W > W_max)
  disp('Unable to meet requirements with given bandwidth limitation')
end
%system parameters output
m = floor(b_Pe);
if (m==5|m==3)
  disp('QAM constellation is not Gray coded')
end
disp('!!!!!!!!!!!          RESULTS          !!!!!!!!!!!!!!!')
disp(['System bandwidth for DSBSC transmission is ',num2str(2*W/1000),' (kHz)'])
fmax = fc + W;
fmin = fc-W;
if (fmax>7e3|fmin<200)
  disp('Frequency spectrum of system is outside the bandwidth of the known channel transfer function')
end
disp(['System Transmission Band for USBSC is ', num2str(fmin/1000),' (kHz) to ', num2str(fmax/1000),' (kHz).'])
disp(['Maximum Delay Spread is ', num2str(delay_spread*1000),' (msec)'])
disp(['System uses ', num2str(N_subc_act),' active subcarriers with ',num2str(N_bar),' total subcarriers.'])
disp(['The subcarrier bandwidth is ', num2str(SC_del_f), ' (Hz).'])
disp(['OFDM symbol duration is ', num2str(T*1000),' (msec).'])
disp(['OFDM Symbol rate is ', num2str(Rs),' (symbols/sec).'])
```

```
disp(['Number of bits per subcarrier is ',num2str(m),'.'])
disp(['System bit rate is ', num2str(R/1000), ' (kbps).'])
disp(['System Bandwidth Efficiency is ',num2str(R/(W*2)),'.'])
disp(['System carries ', num2str(floor(b_Pe)*N_subc_act),' bits per OFDM symbol.'])
```

3. signal_generator.m

```
function [x, fn] = signal_generator(pts,Fs,f1,f2,a1,a2,decision, factor, fn)
%This function generates a discrete time analog signal.
%The signal is a vector of length pts, from the superposition of
%2 unit amplitude sinusoidal signals of frequency f1 and f2, sampled
%at Fs Hz.
%Last modified 5/15/01
%global fn
if decision == 1
n=0:pts-1;
x = a1*sin(2*pi*f1*n/Fs)+ a2*cos(2*pi*f2*n/Fs);    %information signal
spec_info_sig = fft(x,2048);
fn = fn+1; figure(fn), plot(abs(spec_info_sig))
title('FFT of sampled information signal (info sig)')
elseif decision ==2
s = wavread('numnuts');
l = round(length(s)/factor);
x = s(1:l);
elseif decision == 3
  x = [zeros(1,100), 0:.2:1, ones(1,100), 1:-.2:0, zeros(1,100)];
  spec_info_sig = fft(x,2048);
        fn = fn+1; figure(fn), plot(abs(spec_info_sig))
        title('FFT of sampled information signal (info sig)')
end
```

4. quantization.m

```
function [symbol, codebook] = quantization(info_sig, bits)
%Last modified 3/19/01
M = 2^bits;      %number of uniform quantization levels
%must determine the method for partitioning to use in general
max_sample = max(abs(info_sig));
%absolute maximum value of input signal
part=-(M-1)/M*max_sample:2/M*max_sample:(M-1)/M*.9*max_sample;
          %partitions
codebook = -((M-1)*2 + 1)/(2*M)*max_sample:2/M*max_sample:((M-
1)*2+1)/(2*M)*max_sample;
symbol = quantiz(info_sig, part);
```

5. dec_2_bin.m

```
function bit_stream = dec_2_bin(qam_sym, bits, q)
%Last modified 3/19/01
bit_matrix = de2bi(qam_sym',q);
%binary matrix length(qam_sym) by q
bit_stream_long = reshape(bit_matrix, length(qam_sym)*q, 1);
num_zero_bits = length(qam_sym)*q - floor((length(qam_sym)*q)/bits)*bits;
%number of zero bits inserted in xmitter
bit_stream = bit_stream_long(num_zero_bits+1:length(bit_stream_long));
```

6. qam_buffer.m

```
function qam_dec = qam_buffer(bit_stream, q)
%This function buffers the bit stream into a matrix form for QAM encoding.
%Last modified 3/19/01
add_length = q*(ceil(length(bit_stream)/q))-length(bit_stream);
%determines the number of zeroes needed to be added to bit_stream
%such that buffered matrix will be rectangular and include all of the data.
bit_stream_long = [zeros(1,add_length),bit_stream];
%bit_stream with zeroes added to left end i.e. at 1,2,3...
buffer_mat = reshape(bit_stream_long,length(bit_stream_long)/q,q);
```

```
%buffers bit_stream into q column matrix for QAM modulation
%data is reshaped columnwise
qam_dec = bi2de(buffer_mat);        %converts buffered binary data to corresponding
%decimal integer where the integers are from 0 to Q-1(2^q - 1)
%this reduces the sampling frequency since there are now less samples
%least sig bit assumed in column 1
```

7. reedsolomon.m

```
function [code, zero_symbols] = reedsolomon(qam_coded_dec, N, k)
%This function performs Reed Solomon encoding of the decimal format
%Gray coded Qary QAM symbols in qam_coded_dec
%k is the message length in symbols, N is the code word length in symbols
%Last modified 3/19/01
full_blocks = floor(length(qam_coded_dec)/k);
%number k length blocks filled with data
zero_symbols = (full_blocks + 1)*k - length(qam_coded_dec);
%number of zero blocks needed to fill final block
qam_coded_dec_long = [zeros(zero_symbols,1);qam_coded_dec];
%QAM code with zeroes at top of column vector
qam_k_mat = reshape(qam_coded_dec_long,full_blocks+1,k);
%full_blocks + 1 row by k column QAM symbol matrix for block coding
code = encode(qam_k_mat, N, k, 'rs/decimal');
%full_blocks + 1 row by N column matrix of Qary-QAM encoded symbols
%RS encoded
```

8. qam_modulator.m

```
function [X, data_carriers, zero_fill] = qam_modulator(code_stream, N_bar, N_subc_act, N_zerc, Q)
%Last modified 5/15/01
[inphase, quad] = qaskenco(code_stream,Q);
%encodes integer decimals into quadriture and inphase components of Q-ary QAM
```

```
[phase_qam, mag_qam] = cart2pol(inphase,quad);
%converts QAM subsymbols to magnitude and phase(radians) for IFFT
zero_carriers = N_zerc;          %ceil(zero_fraction*N_bar);
%number of zero subcarriers near Fs/2
data_carriers = N_bar - zero_carriers;
%number of subcarriers with encoded data
data_blocks = ceil(length(phase_qam)/data_carriers);
%number of blocks( i.e. symbols) neeeded for given N_bar and zero fraction
zero_fill = data_blocks*data_carriers - length(phase_qam);
col_z = floor(zero_fill/data_blocks);
%number of zeroes to be added into QAM symbol matrix for columns 2 :
data_blocks
col_1z = zero_fill - (data_blocks-1) * col_z;
%number of zeroes to be added to the first column in the QAM symbol matrix
zero_fill = [col_1z, ones(1, data_blocks+2-1)*col_z];
phase_c1 = phase_qam(1:(data_carriers - col_1z));
phase_col = phase_qam(1+(data_carriers - col_1z):length(phase_qam));
col_phase_mat = reshape(phase_col, data_carriers - col_z, data_blocks-1);
mag_c1 = mag_qam(1:(data_carriers - col_1z));
mag_col = mag_qam(1+(data_carriers - col_1z):length(mag_qam));
col_mag_mat = reshape(mag_col, data_carriers - col_z, data_blocks-1);
tpc1 = [phase_c1(1:col_1z).';zeros(1,col_1z)];
tpc2 = reshape(tpc1, 2*col_1z, 1);
phasec10 = [tpc2; phase_c1(col_1z+1:length(phase_c1))];
tmc1 = [mag_c1(1:col_1z).';zeros(1,col_1z)];
tmc2 = reshape(tmc1, 2*col_1z, 1);
magc10 = [tmc2; mag_c1(col_1z+1:length(mag_c1))];
for col = 1 : data_blocks - 1
  tpcola = [col_phase_mat(1:col_z, col).'; zeros(1, col_z)];
  tpcolb = reshape(tpcola, 2*col_z, 1);
  phasecol0(:,col) = [tpcolb;col_phase_mat(col_z+1:size(col_phase_mat,1), col)];
  tmcola = [col_mag_mat(1:col_z, col).'; zeros(1, col_z)];
  tmcolb = reshape(tmcola, 2*col_z, 1);
```

```
  magcol0(:,col) = [tmcolb;col_mag_mat(col_z+1:size(col_mag_mat,1), col)];
end
phase_qam_mat = [phasec10, phasecol0];
mag_qam_mat = [magc10, magcol0];
for k = 1:data_blocks
   X_half(:,k) = zeros(N_bar+1,1);
   for i=1:data_carriers;
    X_half(i+1,k) = mag_qam_mat(i,k).*exp(j.*phase_qam_mat(i,k));
        end
        X_half(1,k) = real(X_half(data_carriers+1,k));
        X_half(data_carriers+1,k) = imag(X_half(data_carriers+1,k));
        X(:,k) = [X_half(:,k); conj(flipud(X_half(2:N_bar,k)))];
end
count = 0;
for r = data_carriers+1:-1:1
  for c = size(X_half,2):-1:1
    count = count + 1;
    X_pilot_half(count,1) = X_half(r,c);
  end
end
X_half(:,data_blocks+1) = zeros(N_bar+1,1);
X_half(:,data_blocks+2) = zeros(N_bar+1,1);
X_half(2:data_carriers+1,data_blocks+1) = X_pilot_half(1 : data_carriers);
X_half(1,data_blocks+1) = real(X_half(data_carriers+1,data_blocks+1));
X_half(data_carriers+1,data_blocks+1)                                    =
imag(X_half(data_carriers+1,data_blocks+1));
X_half(1:data_carriers+1,data_blocks+2)                                  =
flipud(X_half(1:data_carriers+1,data_blocks+1));
for c = 1:2
  for r = 1:data_carriers+1
    if X_half(r,2) == 0
      X_half(r,data_blocks+c) = 0;
    end
```

```
  end
end
for k = data_blocks+1:data_blocks+2
  X(:,k) = [X_half(:,k); conj(flipud(X_half(2:N_bar,k)))];
end
X_pilots = X(:, data_blocks+1:size(X,2));
X = [X_pilots, X(:,1:data_blocks)];
for k = 1:data_blocks+2
  for i=1:data_carriers;
    if (k <= data_blocks)
      X_half(i+1,k) = mag_qam_mat(i,k).*exp(j.*phase_qam_mat(i,k));
                    %creates first half of complex symbol, other half is the
conjugate
      %symmetric counterpart
    else
     X_half(i+1,k) = 1;
    end
  end
  X_half(1,k) = real(X_half(data_carriers+1,k));
        %first channel is the real part of the last symbol
  if k <= data_blocks
  X_half(data_carriers+1,k) = imag(X_half(data_carriers+1,k));
        end
        X(:,k) = [X_half(:,k); conj(flipud(X_half(2:N_bar,k)))];
end
```

9. inverse_fft.m

```
function x_n = inverse_fft(X)
%Last modified 4/13/01
x_n = ifft(X);
%N time domain samples, real valued from symmetric freq spectrum, X
%x_n is a matrix the same size as X, representing the time domain samples
```

```
%to be transmitted for each block
max_imag = max(max(imag(x_n)));
if (max_imag>=1e-14)
  error('Time domain output from IFFT not real to machine accuracy')
end
```

10. cyc_extension.m

```
function x_cycex = cyc_extension(x, N_bar,L)
%Last modified 3/19/01
for i = 1:size(x,2)
x_cycex(:,i) = [x(2*N_bar-L+1:2*N_bar,i);x(:,i)];
end
```

11. parlel_2_serial.m

```
function x_ser = parlel_2_serial(x_cyclic)
%This function converts the parallel output of the IFFT to serial form.
%x_ser(1) represents the last sample to be transmitted
%Last modified 3/19/01
x = reshape(x_cyclic, 1, size(x_cyclic,1)*size(x_cyclic,2));
x_ser = real(x);
%real command necessary to correct for small imaginary
%components in x from machine error
%converts symbol from parallel to serial w/ x(N,data_blocks)leading the signal
%and x(1,1) trailing
```

12. dsb_modulator.m

```
function [mod_sig, Fs_mod, f_m, n_lpf2, len_xint, fn] = dsb_modulator(xmit_sig,
fc, Fs_prime, W, zero_fraction, fn);
%This function modulates the baseband signal, xmit_sig to the carrier frequency,
fc, using
%double sideband suppressed carrier (DSBSC) modulation.
```

```
%The baseband signal is interpolated as necessary to meet Nyquist frequency requirements
%Last modified 4/5/01
%plots
N=8192;                                                    %fft length
f_p=(0:N-1)*Fs_prime/N;              %index for plotting fft's
%%%
```

```
digf_pass2 = 2*pi*fracpass2*W/(Fs_mod);  % relative pass frequency
digf_stop2 = 2*pi*fracstop2*W/(Fs_mod);  % relative stop frequency
xtion2 = digf_stop2-digf_pass2;          %transition region width
fpass2 = fracpass2*W/(.5*Fs_mod);
%fraction of Fs/2 for cutoff frequency in FIR filter from fir1.m
n_lpf2 = round( 8*pi/xtion2 );           %hamming filter order
if (isposint(n_lpf2/2) ~= 1)             %ensures filter has odd number of coefficients
  n_lpf2 = n_lpf2+1;
end
[Nh2,Dh2] = fir1( n_lpf2 , fpass2 );     %Hamming LP filter numerator and denominator coefficients
x_int_lpf = conv(Nh2, x_int);
%%%
```

```
%can be 0,2 or 4
%0 for all allowed fc, 2 for deep water and fc = 7 only,
%2 and 4 for shallow water at fc = 7 only
recvd_sig = mod_xmit_sig;
%%%
```

```
  [sync_sig, synch, fn] = synchronizer(recvd_sig, block_length, num_blocks, N_bar, L, SNR_awgn, fn);
else
  sync_sig = recvd_sig;
end
%%%
```

```
%%%
```

2. rcvr_dsb_demodulator.m

```
function [dec_f_rsig, fn] = rcvr_dsb_demod(recvd_sig, Fs_xmit, fc, mod_xmit_sig, W, Fs_prime, n_lpf2, len_xint, synchdec, fn);
%This function demodulates the DSBSC signal from the channel transfer function
%Last modified 5/9/01
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%DEMODULATION
demodsig = cos(2*pi*fc*(1:length(recvd_sig))/Fs_xmit);
%demodulating signal to combine upper side band spectrums to recreate
%baseband signal
demod_rec_sig = recvd_sig.*demodsig;
%demodulated time domain signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N = 2*8192; %length(demod_rec_sig);
f = (0:N-1)*Fs_xmit/N;                  %index for plotting fft's
%frequency index of received signal spectrum
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%LOW PASS FILTERING
fracpass = 1.075;
% fraction of W for pass band
fracstop = 1.2;
% fraction of W for stop band
digf_pass=2*pi*fracpass*W/(Fs_xmit);        % relative pass frequency
digf_stop=2*pi*fracstop*W/(Fs_xmit);        % relative stop frequency
if (digf_stop>=.99*pi)
  error('Stop band exceeds .99 of the Sampling frequency')
end
xtion=digf_stop-digf_pass;                  %transition region width
fpass = fracpass*W/(.5*Fs_xmit);
%fraction of Fs/2 for cutoff frequency in FIR filter from fir1.m
```

```
n_lpf = round( 8*pi/xtion );          %hamming filter order
if (isposint(n_lpf/2) ~= 1)           %ensures filter has odd number of coefficients
  n_lpf = n_lpf+1;
end
[Nh,Dh] = fir1( n_lpf, fpass);        %Hamming LP filter numerator and denominator coefficients
filt_rsig = conv(Nh, demod_rec_sig);
if synchdec == 0
time_delay_mod = n_lpf2;
time_delay_demod = n_lpf;
time_delay_tot = floor(time_delay_mod/2) + floor(time_delay_demod/2);
filt_rsig = filt_rsig(1, time_delay_tot+1:len_xint+time_delay_tot);
        %THIS COMMAND PERFORMS PERFECT SYNCHRONIZATION
end
%%%
```

```
    else
      filt_rsig = [filt_rsig, 0 0 0];
    end
  end
end
if D>=5
  error('Interpolation/Decimation factors greater than 4 requires code adjustment in rcvr_dsb_demod.m')
end
dec_mat = reshape(filt_rsig, D, length(filt_rsig)/D);
dec_f_rsig = dec_mat(1,:);
%decimated received signal, returns the sampling frequency to Fs_prime = Fs_xmit/D
f_p = (0:N-1)*Fs_prime/N;              %index for plotting fft's
%frequency index of received signal spectrum
```

3. synchronizer.m

```
function [sync_sig, synch, fn] = synchronizer(recvd_sig, block_length, num_blocks, N_bar, L, SNR_awgn, fn)
%Correlator synchronizer
%last modified 6/21/2001
%%%
```

```
display(SNR_est_synch)
leng_noise1 = 1000;
leng_noise2 = 1500;
r = [sqrt(Pn)*randn(1, leng_noise1), recvd_sig, sqrt(Pn)*randn(1, leng_noise2) ];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for k = 1:length(r)-2*N_bar-2*L+1              %loops over entire length of data
  r_sig = r(1, k:k+L-1);
  r_del_conj_sig = conj(r(1, k+2*N_bar:k+2*N_bar+L-1));
  corr_sig(k) = max(xcorr(r_sig, r_del_conj_sig));
end
[synch, fn] = peakfinder(corr_sig, fn);
%last block identifies the ofdm block length
%previous blocks identify the discrete time prior to the beginning of the data blocks
%i.e. synch(1) is the beginning of the first data block and synch(end) is the ofdm block length
%performance check
block_length_error = abs(block_length - synch(length(synch)));
peak_count_error = num_blocks - (length(synch) - 1);
block_start_error = abs(leng_noise1 + 1 - synch(1));
disp(['Block length error = ', num2str(block_length_error),' discrete time units.'])
disp(['Peaks missed = ', num2str(peak_count_error),'.'])
sync_sig = r(synch(1)+1 : synch(1) + 1 + num_blocks*block_length - 1);
```

a. peakfinder.m

```
function [synch, fn] = peakfinder(corr_sig, fn)
%last modified 5/16/01
peak_index = 0;
%peaks = find(corr_sig>mn + 2.5*sd);
fracnum = 0;
t1 = .4; t2 = .6; dt = .05;                %threshholds and differential
for fraction = t1:dt:t2
```

```
  fracnum = fracnum + 1;
  %fraction = 0.7;
peaks = find(corr_sig > fraction*max(corr_sig));
count = 0;
peaknum = 0;
while count < length(peaks)
  same_peak = 1;
  peaknum = peaknum + 1;    %index for number of peaks found
  lookstart = count + 1;
  while same_peak == 1;
    count = count + 1;
    if (count >= length(peaks))
      break
    end
    if (peaks(1+count) - peaks(count) == 1)
        same_peak = 1;
          else
          same_peak = 0;
          end
        end
                %while and if loop determine the indices of corr_sig corresponding to the 1st peak which is where
                %the first OFDM symbol begins
                %count-1 after while loop is the last index of the 1st peak,
    nextpeakstart = count + 1;
                [mag_peak(peaknum), peak(peaknum)] = max(corr_sig(peaks(lookstart):peaks(count)));
        peak_index(fracnum, peaknum) = peak(peaknum) + peaks(lookstart) - 1;
end
end
%disp(fraction)
%disp(peak_index)
```

```
numtruepeaks = 0;
for l = 1:size(peak_index,2)
  if peak_index(1,l) ~= 0
    matches = find(peak_index(1,l) == peak_index(2:size(peak_index,1),:));
        if length(matches) == size(peak_index,1) - 1
    numtruepeaks = numtruepeaks + 1;
    truepeakindex(numtruepeaks) = peak_index(1,l);
    end
  end
end
%break
block_length = truepeakindex(2) - truepeakindex(1) - 1;
synch = [truepeakindex, block_length];
fn=fn+1; figure(fn), plot(corr_sig), title('Correlator Synchronizer Output'), hold on
plot(truepeakindex, corr_sig(truepeakindex), 'r*')
for k = t1:dt:t2
  line([1, length(corr_sig)], [k*max(corr_sig), k*max(corr_sig)])
end
hold off
```

4. mmpe_channel.m

```
function [X_ch, rcvr_depth, fn] = mmpe_channel(X_r_eb, N_bar, N_subc_act, fc, depth, range, rough, fn)
%This function applies the transfer function from the MMPE model to the received
%QAM symbols. X_r_ebch(:,i) = X_r_eb(:,i).*H(z), where i is the OFDM block index
%
%Last modified 6/4/01
if N_bar == 1024
  if fc ~= 6e3
    error('Carrier frequency does not correspond to model carrier frequency')
```

```
  end
  if (depth == 0 & range == 0 & rough == 0)
    load f7r02k.mat;
  elseif (depth == 0 & range == 0 & rough == 2)
    load f7r22k.mat;
  elseif (depth == 0 & range == 0 & rough == 4)
    load f7r42k.mat;

  elseif (depth == 0 & range == 1 & rough == 0)
      load f7r04k.mat;
  elseif (depth == 0 & range == 1 & rough == 2)
    load f7r24k.mat;
  elseif (depth == 0 & range == 1 & rough == 4)
    load f7r44k.mat;

  elseif (depth == 1 & range == 0 & rough == 0)
    load f7r02kd.mat;
  elseif (depth == 1 & range == 0 & rough == 2)
    load f7r22kd.mat;

  elseif (depth == 1 & range == 1 & rough == 0)
      load f7r04kd.mat;
  elseif (depth == 1 & range == 1 & rough == 2)
    load f7r24kd.mat;
  end

elseif N_bar == 2048

  if (fc == 8e3 & range == 0)
    load f10r02k.mat;
  elseif (fc == 8e3 & range == 1)
    load f10r04k.mat;
  elseif (fc == 10e3 & range == 0)
```

```
    load f12r02k.mat;
  elseif (fc == 10e3 & range == 1)
    load f12r04k.mat;
  elseif (fc == 12e3 & range == 0)
    load f14r02k.mat;
  elseif (fc == 12e3 & range == 1)
    load f14r04k.mat;
  end
end
r = size(press,1);              %number of depths in model, first half are in acoustic channel, i.e. water
[v dep_index] = max(sum(abs(press(1:r/2,:)),2));              %dep_index indetifies the depth with the greatest
%return over all frequencies
H = press(dep_index,:);              %places the receiver at the optimal depth
%H = press(10,:);
H_half = zeros(length(H)+1,1);
for k = 1:length(H)
  H_half(k+1) = H(k);
end
H_half(1) = real(H_half(length(H_half)));
H_half(length(H_half)) = imag(H_half(length(H_half)));
H_full = [H_half; conj(flipud(H_half(2:N_bar)))];
%transfer function to be applied to received QAM symbols
h = flipud(ifft(H_full));
rcvr_depth = (1-depth)*(dep_index/(r/2))*100 + depth*(dep_index/(r/2))*340;
%receiver depth in meters
for k = 1:size(X_r_eb,2)
  X_ch(:,k) = X_r_eb(:,k).*H_full;
End
```

5. equalization.m

```
function X_r = equalization(X_r_eb, X, N_subc_act, N_bar, fn)
```

```
%performs channel equalization using pilot blocks
%last modified 5/15/01
H_est_mat_real = X_r_eb(1:N_subc_act+1, 1:2)./ ...
  X(1:N_subc_act+1, 1:2);
H_est_mat_imag = X_r_eb(size(X_r_eb,1) - N_subc_act + 1: size(X_r_eb,1), 1:2)./ ...
  X(size(X,1) - N_subc_act + 1: size(X,1), 1:2);
%estimate of channel freqency response from 2 unit magnitude OFDM blocks
%estimate is the average of the 2 columns and is used for channel equalization
%real is estimate for first N_subc_act subcarriers, and imag is for last N_subc_act subcarriers
%equalization is only performed over the active subcarriers
H_est_real = mean(H_est_mat_real.').';
H_est_imag = mean(H_est_mat_imag.').';
%2*N_subc_act column vector representing the channel estimate over the actice subcarriers
X_r = X_r_eb(:, 3:size(X_r_eb,2));
%removes channel estimation blocks assuming the first 2 blocks are the pilot blocks and no others
for k1 = 1:size(X_r,2);
  X_r_real(:,k1) = X_r(1:N_subc_act+1,k1)./H_est_real;
  X_r_imag(:,k1) = X_r(size(X_r,1) - N_subc_act + 1: size(X_r,1),k1)./H_est_imag;
  X_r(:,k1) = [X_r_real(:,k1); zeros(2*N_bar - 2*N_subc_act - 1,1); X_r_imag(:,k1)];
end
%equalizes the data for the channel
```

6. qam_demodulator.m

```
function [code, inphase_r, quad_r, fn] = qam_demodulator(X_r, N_bar, zero_fraction, Q, data_carriers, N,  zero_fill, fn)
%Last modified 5/11/01
X_half_r = X_r(1:data_carriers+1,:);
%strips conjugate symmetric and zero subcarriers from data
```

```
for k = 1: size(X_half_r,2)
  for i = 1:data_carriers
    mag_r(i,k) = abs (X_half_r(i+1,k));
    %determines the magnitude of the QAM symbol
    phase_r(i,k) = angle(X_half_r(i+1,k));
    %determines the phase of the QAM symbol
  end
  mag_r(data_carriers,k) = sqrt(X_half_r(1,k)^2 + real(X_half_r(data_carriers+1,k))^2);
    %combines the Re(N_bar+1) and the Imag(N_bar+1) that were separated
    %in the qam modulator to get the magnitude if the final symbol
  phase_r(data_carriers,k) = atan2(real(X_half_r(data_carriers+1,k)), X_half_r(1,k));
        %combines the Re and Im at N_bar + 1 to get the phase for the final symbol
        %real command necessary to account for small imaginary component resulting from
        %machine error
end
%result of these 2 loops is a data_carrier row matrix representing the QAM modulated
%symbols where each column is the kth symbol and each row is the ith subsymbol
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
tmag1(:,1) = mag_r(1:2:zero_fill(1)*2,1);
tphase1(:,1) = phase_r(1:2:zero_fill(1)*2,1);
for col = 2:size(X_r,2)
  tmagcol(:,col-1) = mag_r(1:2:zero_fill(col)*2,col);
  tphasecol(:,col-1) = phase_r(1:2:zero_fill(col)*2,col);
end
mag_r_vec1(:,1) =  [tmag1(:,1); mag_r(zero_fill(1)*2+1:length(mag_r),1)];
phase_r_vec1(:,1) = [tphase1(:,1); phase_r(zero_fill(1)*2+1:length(phase_r),1)];
for col = 2:size(X_r,2)
```

```
  mag_matcol(:,col-1) = [tmagcol(:,col-1); mag_r(zero_fill(col)*2+1:length(mag_r),col)];
  phase_matcol(:,col-1) = [tphasecol(:,col-1); phase_r(zero_fill(col)*2+1:length(phase_r),col)];
end
mag_r_veccol = reshape(mag_matcol, size(mag_matcol,1)*size(mag_matcol,2), 1);
phase_r_veccol = reshape(phase_matcol, size(phase_matcol,1)*size(phase_matcol,2), 1);
mag_r_vec_short = [mag_r_vec1; mag_r_veccol];
phase_r_vec_short = [phase_r_vec1; phase_r_veccol];
%%%
```

```
%       by the form:
%                   | X_min    X_max |
%          MINMAX = |                 |
%                   | Y_min    Y_max |
%
%       See also QASKENCO.
%       Last modified 5/11/01
scale = 0;        %logic statement to scale received symbols, 1 = yes, o/w = no
%if scale == 0
%     disp('NO SCALING OF RECEIVED QAM CONSTELLATION PERFORMED IN QAM DECODER')
%end
error(nargchk(1,4,nargin));
if (nargin <= 2) | (nargin > 4)
   error('Wrong number of input arguments.')
end;
M = m;
if isstr(m)
  error(['QASK constellation has changed to use Gray code. ', ...
        sprintf('\n'),...
        'Use COMMUPDT("FILENAME") to update your SIMULINK model'...
        sprintf('\n'),...
        'before the simulation.']);
end;
K = log(m) ./ log(2);
if m ~= 2^K
   error('M must equal to 2^K where K is an integer.')
end;
xx = constlay(K, 1);
[leny, lenx] = size(xx);
if (nargin == 4)
```

```
    [xmat,ymat] = qaskenco(M);
    %xmat and ymat are the inphase and quad components, respectively, of
the Mary QAM
    %constellation
    MAT = [xmat.';ymat.'];
    %2 by M matrix of vectors in 2D representing the M Qam symbols in
the constelation
    for k=1:length(x)
      E = [x(k); y(k)]*ones(1,length(xmat)) - MAT;
      [v, index] = min([1,1]*abs(E));
      msg(k) = index-1;
    end
    end;
[x_m, x_n] = size(x);
[y_m, y_n] = size(y);
if min(x_m, x_n) * min(y_m, y_n) ~= 1
  error('Input X and Y must be vectors.');
end;
if x_m > x_n
  msg = msg';
end;
```

7. deinterleaver.m

```
function deint_mat = deinterleaver(code, N)
%Last modified 3/19/01
rows = length(code)/N;
%number of rows in matrix for RS decoding
deint_mat = reshape(code, rows, N);
%matrix of symbols deinterleaved, ready for RS decoder
```

8. rsdecoder.m

```
function decode = rsdecoder(code_deint, N, k, zero_symbols)
%Last modified 3/19/01
deco = rsdeco(code_deint, N, k, 'rs/decimal');
%message length k columns matrix
deco2 = reshape(deco, 1, size(deco,1)*size(deco,2));
%converts to row vector
decode = deco2(zero_symbols+1:length(deco2));
%removes zeroes added in transmitter
%zero symbols passed from xmitter function reedsolomon.m
```

9. error_check.m

```
function ber = error_check(bs, bsr)
%This function determines the Bit Error Rate (BER) for the system.
%last modified 7/2/01
be = 0;
for l = 1:length(bs)
  be = be + (bs(l)~=bsr(l));
end
ber = be/l;
```

C. PERFORMANCE ANALYSIS CODE

1. ofdm_sim_control_func.m

```
function [BER, SNR_est] = ofdm_sim_control_func(SNR, fc, R, rough, range, depth)
%Control m-file for ofdm simulation
fn = 0;
modulate = 2;            %DSBSC modulation logic control value
noise = 1;
```

```
for avn = 1:10
count = 0;
for noise_factor  = .15:.01:.65
  count = count + 1;
  synchdec = 0;                    %synchronization logic control value, 0 == NO, 1 == YES
  [zero_fill, Q, N, data_carriers, N_bar, N_subc_act, zero_symbols, k, bits, info_sig, code_stream, bit_stream, X, mod_xmit_sig, ...
      Fs_prime, Fs_xmit, fn, block_length, zero_fraction, SNR_awgn, chan_app, W, n_lpf2, len_xint, L, q, codebook] = ...
        ofdm_sim_xmitter_func(synchdec, noise, noise_factor, fn, R, fc, SNR);
        [SNR_est(count,avn), BER(count,avn)] = ofdm_sim_receiver_func(W, zero_fill, Q, N, data_carriers, N_bar, N_subc_act, zero_symbols, k, bits, info_sig, ...
    code_stream, bit_stream, X, mod_xmit_sig, Fs_prime, Fs_xmit, fn, block_length, zero_fraction, SNR_awgn, ...
    depth, range, rough, chan_app, modulate, noise, fc, synchdec, n_lpf2, len_xint, L, q, codebook);
end
end
```

2. ofdm_performance.m

```
%OFDM performance analysis m-file
%runs ofdm sim control func m-file over all mmpe channels
%last modified 6/28/01
warning off
clear all
%%%
```

```
for range = 0:1          %0 == 2 km, 1 == 4 km
  count = count + 1;
  rough = 0;
  depth = 0;
  [BERw, SNR_estw] = ofdm_sim_control_func(SNR, fc, R, rough, range, depth);
  BER(:,:,count) = BERw;
  SNR_est(:,:,count) = SNR_estw;
end
clear BERw SNR_estw
fc = 10e3
R = 6e3
for range = 0:1          %0 == 2 km, 1 == 4 km
  count = count + 1;
  rough = 0;
  depth = 0;
  [BERw, SNR_estw] = ofdm_sim_control_func(SNR, fc, R, rough, range, depth);
  BER(:,:,count) = BERw;
  SNR_est(:,:,count) = SNR_estw;
end
clear BERw SNR_estw
fc = 12e3
R = 7.5e3
for range = 0:1          %0 == 2 km, 1 == 4 km
  count = count + 1;
  rough = 0;
  depth = 0;
  [BERw, SNR_estw] = ofdm_sim_control_func(SNR, fc, R, rough, range, depth);
  BER(:,:,count) = BERw;
  SNR_est(:,:,count) = SNR_estw;
end
```

```
clear BERw SNR_estw
%%%
```

```
  rough = 0;
  depth = 0;
  [BERw, SNR_estw] = ofdm_sim_control_func(SNR, fc, R, rough, range, depth);
  BER(:,:,count) = BERw;
  SNR_est(:,:,count) = SNR_estw;
end
clear BERw SNR_estw
fc = 8e3
R = (6+2/3)*1e3
for range = 0:1          %0 == 2 km, 1 == 4 km
  count = count + 1;
  rough = 0;
  depth = 0;
  [BERw, SNR_estw] = ofdm_sim_control_func(SNR, fc, R, rough, range, depth);
  BER(:,:,count) = BERw;
  SNR_est(:,:,count) = SNR_estw;
end
clear BERw SNR_estw
fc = 10e3
R = (8+0/3)*1e3
for range = 0:1          %0 == 2 km, 1 == 4 km
  count = count + 1;
  rough = 0;
  depth = 0;
  [BERw, SNR_estw] = ofdm_sim_control_func(SNR, fc, R, rough, range, depth);
  BER(:,:,count) = BERw;
  SNR_est(:,:,count) = SNR_estw;
end
clear BERw SNR_estw
```

```
%%%
```

```
  count = count + 1;
  [BERw, SNR_estw] = ofdm_sim_control_func(SNR, fc, R, rough, range, depth);
  BER(:,:,count) = BERw;
  SNR_est(:,:,count) = SNR_estw;
end     %rough
end     %range
clear BERw SNR_estw
fc = 6e3
R = (5+0/3)*1e3
depth = 1;
for range = 0:1          %0 == 2 km, 1 == 4 km
  for rough = 0:2:2
  count = count + 1;
  [BERw, SNR_estw] = ofdm_sim_control_func(SNR, fc, R, rough, range, depth);
  BER(:,:,count) = BERw;
  SNR_est(:,:,count) = SNR_estw;
end     %rough
end     %range
clear BERw SNR_estw
%%%
```

```
  BER(:,:,count) = BERw;
  SNR_est(:,:,count) = SNR_estw;
end
clear BERw SNR_estw
fc = 6e3
R = (5+0/3)*1e3
depth = 0;
for range = 0:1          %0 == 2 km, 1 == 4 km
  for rough = 0:2:4
  count = count + 1;
  [BERw, SNR_estw] = ofdm_sim_control_func(SNR, fc, R, rough, range,
depth);
  BER(:,:,count) = BERw;
  SNR_est(:,:,count) = SNR_estw;
end     %rough
end     %range
clear BERw SNR_estw
fc = 6e3
R = (5+0/3)*1e3
depth = 1;
for range = 0:1          %0 == 2 km, 1 == 4 km
  for rough = 0:2:2
  count = count + 1;
  [BERw, SNR_estw] = ofdm_sim_control_func(SNR, fc, R, rough, range,
depth);
  BER(:,:,count) = BERw;
  SNR_est(:,:,count) = SNR_estw;
end     %rough
end     %range
clear BERw SNR_estw
warning on
save perform BER SNR_est
```

3. perform_plotter.m

```
%m-file to plot BER vs SNR_est for performance results
clear all
load perform
fn = 0;
close all
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%8 QAM
for l = 1:6
  mn_ber8(:,l) = mean(BER(:,:,l),2);
  mn_snr8(:,l) = mean(SNR_est(:,:,l),2);
end
fnc = 0
fnc = fnc+1
figure(fnc)
plot(mn_snr8,mn_ber8)
title('8 QAM Composite Results')
xlabel('SNR (dB)'), ylabel('BER')
legend('1', '2', '3', '4', '5', '6')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%16 QAM
for l = 7:22
  mn_ber16(:,l) = mean(BER(:,:,l),2);
  mn_snr16(:,l) = mean(SNR_est(:,:,l),2);
end
fnc = fnc+1
figure(fnc)
plot(mn_snr16,mn_ber16)
title('16 QAM Composite Results')
xlabel('SNR (dB)'), ylabel('BER')
legend('1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%32 QAM
for l = 23:36
  mn_ber32(:,l) = mean(BER(:,:,l),2);
  mn_snr32(:,l) = mean(SNR_est(:,:,l),2);
end
fnc = fnc+1
figure(fnc)
plot(mn_snr32,mn_ber32)
title('32 QAM Composite Results')
xlabel('SNR (dB)'), ylabel('BER')
legend('1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14')
%%%%%%%%%%%%%%%%%%%%%%%%%%%
%64 QAM
for l = 37:48
  mn_ber64(:,l) = mean(BER(:,:,l),2);
  mn_snr64(:,l) = mean(SNR_est(:,:,l),2);
end
fnc = fnc+1
figure(fnc)
plot(mn_snr64,mn_ber64)
title('64 QAM Composite Results')
xlabel('SNR (dB)'), ylabel('BER')
legend('1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12')
%break
%%%%%%%%%%%%%%%%%%%
```

```
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fc = 10e3
R = 6e3
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
```

```
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fc = 12e3
R = 7.5e3
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
%%%
```

```
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 2, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 4, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 0, Water Depth = 100 m'])
```

```
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 2, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 4, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
%depth = 0;
%for range = 0:1                %0 == 2 km, 1 == 4 km
%   for rough = 0:2:4
%   count = count + 1;
%    [BERw, SNR_estw] = ofdm_sim_control_func(SNR, fc, R, rough, range, depth);
%   BER(:,:,count) = BERw;
%   SNR_est(:,:,count) = SNR_estw;
%end   %rough
%end   %range
%clear BERw SNR_estw
%%%
fn = fn + 1; figure(fn), orient tall
```

```
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 0, Water Depth = 340 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 2, Water Depth = 340 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 0, Water Depth = 340 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 2, Water Depth = 340 m'])
xlabel('SNR (dB)'), ylabel('BER')
```

```
%%%
fc = 12e3
R = 10e3
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fc = 8e3
R = (6+2/3)*1e3
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
```

```
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fc = 10e3
R = (8+0/3)*1e3
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
%%%
```

```
%32-ary QAM PERFORMANCE
SNR = 25
fc = 10e3
R = (10+0/3)*1e3
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fc = 8e3
R = (8+1/3)*1e3
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 0, Water Depth = 100 m'])
```

```
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fc = 6e3
R = (5+0/3)*1e3
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 2, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
```

```
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 4, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 2, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 4, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
```

```
%%%
fc = 6e3
R = (5+0/3)*1e3
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 0, Water Depth = 340 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 2, Water Depth = 340 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 0, Water Depth = 340 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
```

```
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 2, Water Depth = 340 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
%%%
```

```
R = (5+0/3)*1e3
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 2, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 4, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
```

```
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 0, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 2, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 4, Water Depth = 100 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fc = 6e3
R = (5+0/3)*1e3
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 0, Water Depth = 340 m'])
xlabel('SNR (dB)'), ylabel('BER')
```

```
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 2 km, Rough = 2, Water Depth = 340 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 0, Water Depth = 340 m'])
xlabel('SNR (dB)'), ylabel('BER')
%%%
fn = fn + 1; figure(fn), orient tall
plot(SNR_est(:,:,fn), BER(:,:,fn), 'b.')
hold on
plot(mean(SNR_est(:,:,fn),2), mean(BER(:,:,fn),2), 'r-')
hold off
title(['SNR = ',num2str(SNR), 'dB, f_c = ',num2str(fc/1000),'kHz, R = ',num2str(R/1000),'kbps, Range = 4 km, Rough = 2, Water Depth = 340 m'])
xlabel('SNR (dB)'), ylabel('BER')
```

LIST OF REFERENCES

1. S. Coatelan and A. Glavieux, " Design and Test of a Multi-carrier Transmission System on the Shallow Water Acoustic Channel," *OCEANS '94 'Oceans Engineering for Today's Technology and Tomorrow's Preservation' Proceedings*, Vol. 3, pp. III-472 – III-477, 1994.

2. Kim, Byung-Chul and Lu, I-Tai, "Parameter Study of Underwater Communication," *OCEANS 2000 MTS/IEEE Conference and Exhibition*, Vol. 2, pp. 1251-1255, 11-14 September 2000.

3. J. M. Cioffi, "A Multicarrier Primer," Tutorial, Amati Communications Corporation and Stanford University. (unpublished)

4. R. Van Nee and R. Prasad, *OFDM for Wireless Multimedia Communications*, Artech House, Boston, 2000.

5. J.A.C. Bingham, "Multicarrier Modulation for Data Transmission: An Idea Whose Time has come," *IEEE Communications Magazine*, 28(4):5-14, April 1990.

6. J.M. Wozencraft and P.H. Moose, "Modulation and Coding for In-Band Digital Audio Broadcast using Multi-Frequency Modulation," *National Association of Broadcasters 1991 Conference Proceedings, see also Radio 1991 Proceedings (San Francisco, 9/91)*, Las Vegas, April 1991.

7. L. J. Ziomek, *Fundamentals of Acoustic Field Theory and Space Time Signal Processing*, CRC Press, Boca Raton Fl, 1995.

8. J. E. Houdeshell, "Bandwidth Optimization of Underwater Acoustic Communication Systems," Master's Thesis, Naval Postgraduate School, Monterey, California, 2001.

9. Tappert, F. D. “The parabolic approximation method,” in Lecture Notes in Physics, Vol. 70, Wave propogation and Under Water Acoustics, eds. J.B. Keller and J.S. Papadakis, Springer-Verlag, New York, 1977, pp.224-287.

10. Thomson, D. J. and Chapman, N.R. (1983). “A wide-angle split step algorithm for the parabolic equation,” *J. Acoust. Soc. Am.*, Volume 74, pp. 1848-1854.

11. Smith, Kevin B. (2001), “Convergence, Stability, and Variability of Shallow Water Acoustic Predictions Using the Split-Step Fourier Parabolic Equation Model,” *J. Comp. Acoust.*, Vol.9, No. 1, pp. 243-285.

12. J. van de Beek and others, “Low-Complex Frame Synchronization in OFDM Systems,” *Proceedings of Fourth IEEE International Conference on Universal Personal Communications*, pp. 982-986, 1995.

13. J. G. Proakis, *Digital Communications*, 4th Ed., McGraw-Hill, New York, 2001.

14. R. G. Gallager, *Information Theory and Reliable Communication*, Wiley, New York, 1968.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   Ft. Belvoir, Virginia

2. Dudley Knox Library 2
   Naval Postgraduate School
   Monterey, California

3. Chairman, Code EC 1
   Naval Postgraduate School
   Monterey, California
   jknorr@nps.navy.mil

4. Prof. Roberto Cristi 2
   Naval Postgraduate School
   Monterey, California
   rcristi@nps.navy.mil

5. Prof. Kevin B. Smith 2
   Naval Postgraduate School
   Monterey, California
   kbsmith@nps.navy.mil

6. LT Tiger Pittman 3
   Naval Postgraduate School
   Monterey, California
   glpittma@nps.navy.mil