2003-06

Performance of acoustic spread-spectrum signaling in simulated ocean channels

Pelekanos, Georgios N.

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

PERFORMANCE OF ACOUSTIC SPREAD-SPECTRUM SIGNALING IN SIMULATED OCEAN CHANNELS

by

Georgios N. Pelekanos

June 2003

Thesis Advisor:	Roberto Cristi
Co- Advisor:	Joseph Rice

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2003	**3. REPORT TYPE AND DATES COVERED** Master's Thesis

4. TITLE AND SUBTITLE: **Performance of Acoustic Spread-Spectrum Signaling in Simulated Ocean Channels**	**5. FUNDING NUMBERS**
6. AUTHOR **Georgios N.Pelekanos**	
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)*

Direct-Sequence Spread Spectrum (DSSS) modulation is being advanced as the physical-layer basis for Seaweb undersea acoustic networking. DSSS meets the need for channel tolerance, transmission security, and multi-user access. This thesis investigates the performance of subspace-decomposition blind-equalization algorithms as alternatives to RAKE processing of DSSS signals. This approach is tailored for superior performance in time-dispersive and frequency-dispersive channels characteristic of ocean acoustic propagation. Transmitter and receiver structures are implemented in Matlab and evaluated with a statistics-based model of a doubly spread channel with additive noise. Receiver performance is examined using Monte Carlo simulation. Bit-error rates versus signal-to–noise ratio are presented for various multipath assumptions, noise assumptions, and receiver synchronization assumptions.

14. SUBJECT TERMS Acoustic Communications, Underwater Communications, Underwater Networks, Undersea Warfare, Statistics-Based Channel Modeling, Direct-Sequence Spread-Spectrum, Blind Equalization, Subspace-Decomposition, DSSS, DS-CDMA, Telesonar, Seaweb.	**15. NUMBER OF PAGES** 125
	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

PERFORMANCE OF ACOUSTIC SPREAD-SPECTRUM SIGNALING IN SIMULATED OCEAN CHANNELS

Georgios N. Pelekanos
Lieutenant, Hellenic Navy
B.S. Hellenic Naval Academy, 1992

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING
and
MASTER OF SCIENCE IN ENGINEERING ACOUSTICS

from the

NAVAL POSTGRADUATE SCHOOL
June 2003

Author:	Georgios Pelekanos
Approved by:	Roberto Cristi Thesis Advisor
	Joseph Rice Co-Advisor
	John Powers Chairman, Department of Electrical and Computer Engineering
	Kevin Smith Chairman, Engineering Acoustics Academic Committee

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Direct-Sequence Spread Spectrum (DSSS) modulation is being advanced as the physical-layer basis for Seaweb undersea acoustic networking. DSSS meets the need for channel tolerance, transmission security, and multi-user access. This thesis investigates the performance of subspace-decomposition blind-equalization algorithms as alternatives to RAKE processing of DSSS signals. This approach is tailored for superior performance in time-dispersive and frequency-dispersive channels characteristic of ocean acoustic propagation. Transmitter and receiver structures are implemented in Matlab and evaluated with a statistics-based model of a doubly spread channel with additive noise. Receiver performance is examined using Monte Carlo simulation. Bit-error rates versus signal-to–noise ratio are presented for various multipath assumptions, noise assumptions, and receiver synchronization assumptions.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to dedicate this work to the loving memory of my parents Nikolaos and Maria for the principles and the quest for higher education they taught me.

I am very grateful to my affectionate wife, Konstantina, for her loving support, for the many sacrifices she made throughout my studies, and for the greatest gift of all, the baby she will deliver seven months from now.

I would also like to thank my advisors, Dr. Roberto Cristi and Joseph Rice for their patience and for their guidance, as well as others who provided valuable data and feedback for this study. These include Dale Green of Benthos and Dr. Paul Hursky of SAIC.

I must also express my gratitude to the Hellenic Navy for providing me with the opportunity to pursue a postgraduate degree at the Naval Postgraduate School and to the host U.S. Navy for making my stay a memorable experience.

Most importantly I thank God for his faithful provision throughout my life.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Underwater acoustic signalling is of increasing importance to the Navy since it extends network-centric warfare under the sea. In such applications small-scale channel effects like time variability and time spreading are significant when higher rate and coherent communications are desirable. Time spreading is caused by multipath propagation whereas time variability is the result of movement in the channel. Direct-Sequence Spread-Spectrum (DSSS) modulation is being advanced as the basis for undersea acoustic networking, because it meets the need for channel tolerance, transmission security, and multi-user access in the underwater environment.

This thesis investigates the performance of subspace-decomposition blind-equalization algorithms as alternatives to RAKE processing of DSSS signals. These algorithms assume that the receiver has no a priori knowledge of the channel interference, but, since this interference spans a limited subspace in the eigenvalues of the autocorrelation matrix of the received signal, it can successfully be removed. This approach is tailored for superior performance in time-dispersive and time-spread channels. Transmitter and receiver structures are implemented in MATLAB and evaluated with a simulated statistics-based model of a doubly-spread channel with additive noise. Receiver performance is examined using Monte Carlo simulation. Bit-Error Rates (BER) versus Signal-to–Noise Ratio (SNR) are presented for various multipath assumptions, noise assumptions and receiver synchronization assumptions.

The results suggest that the subspace decomposition method is suitable for underwater acoustic communications. A BER on the order of 10^{-5} is achievable within the desired SNR range. For a very severe channel, communications may be maintained at a lower bit-rate.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

The primary focus of this research is to improve a previously proposed Direct-Sequence Spread-Spectrum (DSSS) signaling scheme intended for use in Seaweb [1]. Seaweb is an experimental underwater acoustic network being developed by the Space and Naval Warfare Systems Center (SPAWARSYSCEN), San Diego which will extend network-centric warfare to the undersea environment [2]. Seaweb is being developed to possess Low Probability of Intercept (LPI) and Low Probability of Detection (LPD) characteristics, high reliability, fault tolerance capability, and relative immunity to slow time-varying multipath. The previously proposed DSSS scheme combined a Differentially Encoded, spread In-Phase and Quadrature (IQ) components, Binary Phase Shift Keying (DS-IQ-BPSK) modulation which was proven very robust in the adverse underwater acoustic environment. The underwater acoustic channel effect has been simulated with an actual impulse response derived from the Signalex 2000 experiment (which means loss of generality) and by using the *Bellhop* propagation model. Bellhop is a static 2-D Gaussian ray model and is described in detail in [3]. Like most of the existing models for underwater acoustic channels, Bellhop is static in the sense that the impulse response only accounts for the multipath effect between transmitter and receiver, considering the surface to be an ideal flat reflecting boundary. However, the ocean surface is seldom stationary or smooth mainly because of the wind-induced surface waves. Part of the research in this thesis will focus on developing a more representative model that captures the effects of time variability. Based on this time-variable channel model, a better acquisition and equalization process will be implemented to further improve the DSSS scheme.

B. SEAWEB REQUIREMENTS

Under the current developments, Seaweb functions with node-to-node communication distances of three to five kilometers, using omni-directional transducers. The nodes are mounted on the bottom of the sea and no special geometry of the node distribution can be assumed because this varies according to the application. Moreover, the network operates in shallow waters (50 to 200 m depth). The available operating frequency bandwidth is 9 to 14 kHz with provisions for a higher 15 to 22 kHz band. The acceptable bit-error rate (BER) is similar to the standards used in RF communication networks and is on the order of 10^{-5}. Of interest to this thesis, Seaweb utility packets are fixed length sequences of 72 bits. The desired information transmission rate is 100 bits per second (bps).

C. GOALS AND METHODOLOGY

The first step in improving the signaling scheme is to use a more representative underwater acoustic model to account for the time variability and Doppler spread of the actual underwater channel. To introduce time variability, well known results and methodology, such as link-budget analysis and small-scale fading, are borrowed from the cellular communication community and adapted for the underwater environment. A static impulse response derived from a physics-based model (Bellhop) initiates the model. Subsequently, the static impulses are introduced with random τ (or lag time) variability and, following a statistics-based approach, a new time-variable model is developed through independent Monte Carlo simulations and time averaging. Finally, the model allows the new time variable impulses to have a Gaussian width in order to simulate time spreading induced by scattering. This approach is **not** intended to produce a high-fidelity underwater acoustic model but rather an acceptable engineering model incorporating properties of the real underwater channel.

The combination of the DS-IQ-BPSK and DSSS transmitting scheme was proven very robust as part of a previous thesis [1] as well as [4]. However, the RAKE receiver group based on the *integrate-and-dump* method was proven inadequate for coherent underwater communications. Therefore, a RAKE replacement is needed for the existing communication scheme in order to operate in the adverse underwater medium. This thesis focuses on developing an improved receiver structure based on advanced signal processing algorithms. The algorithms are implemented in MATLAB for various combinations of multipath, white and colored noise. Two blind-equalization algorithms are implemented based on whether synchronization between the transmitter and the receiver is achieved or not. The final step in the simulation is to test the developed algorithms against the doubly-spread channel, using Monte Carlo simulation. All of the analysis takes place in the discrete time domain and the signals involved are considered to be impulses. Incorporating the algorithms developed here in the actual Seaweb modem will be the task of follow-on work.

D. BENEFITS OF STUDY

Underwater networks that employ wireless acoustic communication have a wide variety of applications in undersea warfare. Therefore, implementing reliable wireless underwater communications is of great importance to the Navy as it extends network-centric warfare under the sea. These wireless underwater acoustic applications include networked telemetry between sensors and base stations in littoral waters, submarine communications, control of minefields, and control of Unmanned Underwater Vehicles (UUVs). DSSS has been proven a durable and reliable scheme for underwater communication. However, the modem design is based on static models, which do not introduce time variability and Doppler spread like the real channel does. A time-variable model permits higher fidelity receiver design, making the existing scheme more robust.

E. THESIS ORGANIZATION

This thesis is organized into five remaining chapters. Chapter II discusses the large-scale sources of signal loss and attenuation in underwater communications and develops the associated link-budget analysis. Chapter III develops the small-scale fading effects of the underwater medium and analyzes the proposed statistics-based channel model. Chapter IV develops the theory for the DS-IQ-BPSK transmitter, and the simplified transmitter and receiver structures, emphasizing the signal processing algorithms implemented in the current communication scheme. Chapter V concerns the MATLAB implementation and performance measurements of the new communication scheme. Finally, Chapter VI reviews and summarizes the results and recommends follow-on work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. LARGE-SCALE FADING CHARACTERISTICS OF THE UNDERWATER ACOUSTIC CHANNEL

This chapter discusses the large-scale fading effect, borrowing terminology from the digital communication community, of the underwater environment on the current communication scheme and uses link-budget analysis to predict the channel SNR and describe its performance. The underwater channel losses obey the one-way sonar equation and are formulated as an acoustic link-budget analysis. The overall losses and attenuation imposed on the transmitted signal by the medium are regarded to be the large-scale fading component. The severe fading of the pressure signal due to multipath (small-scale fading) in shallow water is addressed in the next chapter.

A. ACOUSTIC LINK-BUDGETING

The acoustic link-budget derives from the one way sonar equation:

$$SNR = SL - TL - AN + DI_{TRANSMITTER} + DI_{RECEIVER} \tag{2.1}$$

where all the quantities are in dB. The Signal-to-Noise Ratio (SNR) available at the receiver is defined as the ratio:

$$SNR\ [\text{dB}] = 10\log_{10}\left(\frac{S}{N}\right) \tag{2.2}$$

where S is the time average signal power and N is the time average noise power. Each of the remaining terms in Equation (2.1) is identified in the following paragraphs with a short description from [5].

1. Parameters Determined by the Medium

a. Transmission Loss (TL)

Transmission Loss (*TL*) is a metric used to describe the sum in dB (reference 1 μPa) of the signal loss from the transmitter to the receiver due to spreading and attenuation. Analytically this can be expressed as:

$$TL = TL_g + TL_a \tag{2.3}$$

where TL_g is the component due to geometric spreading and TL_a is the component due to attenuation.

Spreading is a geometrical effect that represents the weakening of the intensity of the sound field as the energy propagates into a larger volume. The signal is assumed to undergo a spherical spreading loss (inversely proportional to r^2) to a range equal to the depth of the channel, gradually giving way to cylindrical spreading loss (inversely proportional to *r*). Spherical spreading loss is expressed as:

$$TL_{g(spherical)} = 10 \cdot \log_{10} \frac{I_1}{I_2} = 20 \cdot \log_{10}(r) \tag{2.4}$$

and the cylindrical spreading loss as

$$TL_{g(cylindrical)} = 10 \cdot \log_{10} \frac{I_1}{I_2} = 10 \cdot \log_{10}(r) \tag{2.5}$$

where I_1 and I_2 are the time-average sound intensities in $\left[W/m^2\right]$ at the reference point (usually at 1 m from the source) and the receiver, respectively, and *r* is the range in [m].

Attenuation occurs as a result of absorption and scattering in the underwater channel. When sound propagates in the underwater medium part of the energy is transformed into heat and part is scattered in the ocean. Attenuation is used to describe the overall effect of leakage out of the sound channel by absorption and scattering as shown in Figure 1 on the next page. We observe that leakage dominates in the first region, ionic relaxation of two constituents of seawater dominates in the second and third, and the viscosity of seawater dominates the last region.

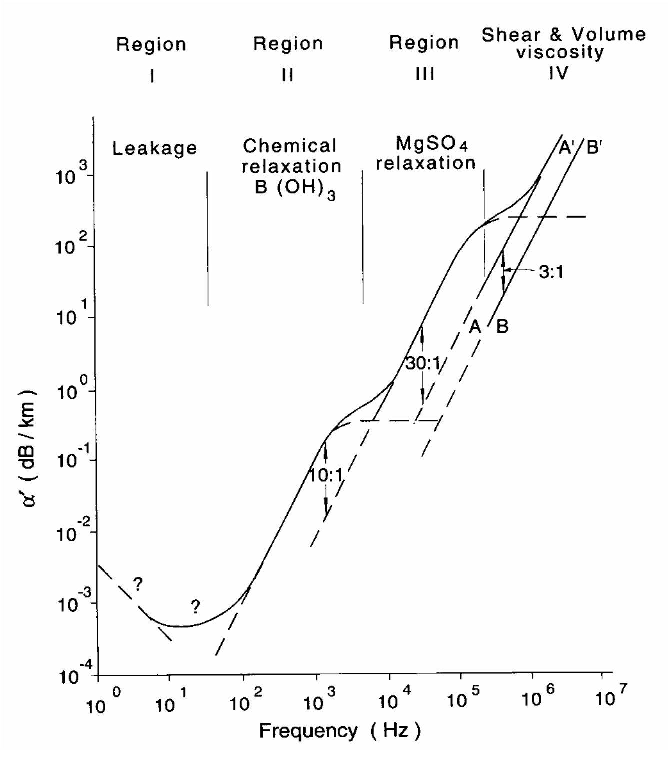


Figure 1. Attenuation Coefficient α (f) [From Ref. 6.]

The attenuation coefficient can be effectively calculated using Thorp's expression [7] and each of the terms is respectively associated with the four regions described previously.

$$\alpha(f) = 3.3\times10^{-3} + 0.11\frac{f^2}{1+f^2} + 44\frac{f^2}{4100+f^2} + 2.75\times10^{-4} f^2 \qquad (2.6)$$

where f is the frequency in [kHz] and $a(f)$ is the frequency-dependent absorption coefficient in [dB/km].

The TL_a component of TL can be calculated by:

$$TL_a = \alpha(f)\cdot r\cdot 10^{-3} \qquad (2.7)$$

where r is the distance in [km] and $\alpha(f)$ is the frequency-dependent absorption coefficient in [dB/km].

b. Ambient Noise (AN)

Ambient Noise (*AN*) is the term describing the combined effect of oceanic turbulence noise, shipping noise, surface agitation noise and thermal noise at the receiver. *AN* is frequency dependent and different noise sources dominate the different frequency bands (in decades) [8]. A graphical representation can be seen in Figure 2 below.

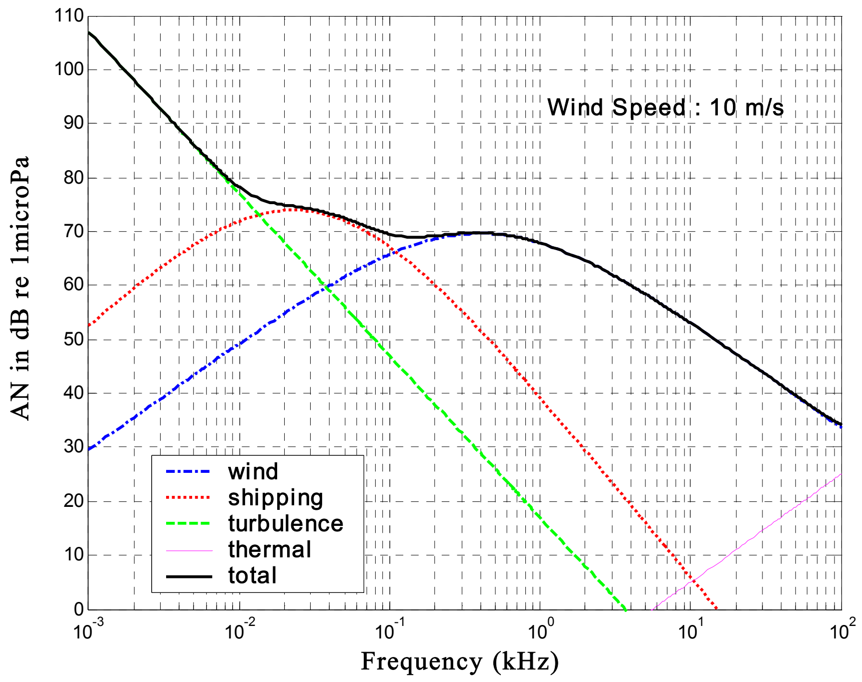


Figure 2. Ambient Noise (dB re 1μPa) vs. Frequency (kHz) [After Ref. 8.]

The first decade (1 Hz to 10 Hz) is dominated by oceanic turbulence whereas the second (10 Hz to 100 Hz) is dominated by shipping. For the next three decades (100 Hz to 100 kHz) wind-induced surface motion is the dominant feature and, for frequencies in excess of 100 kHz, thermal noise is dominant. In the frequency range used in Seaweb (9 to 14 kHz), we observe that the surface agitation noise is the dominant feature.

Ambient noise has several kinds of variations: for example, diurnal, seasonal and geographical. Moreover, in shallow waters two intermittent noise factors must be considered: industrial noise (e.g., boats) and biological noise (e.g., snapping shrimp). To incorporate these factors accurately in an analytical expression would require significant amount of in situ collected data for the deployment area of intent.

The plot of the total ambient noise for different wind conditions in Figure 3 below indicates that in the frequency range 9 to 14 kHz currently used in Seaweb, varying the wind speed from 0 to 20 m/s increases by 30 dB on the total ambient noise.

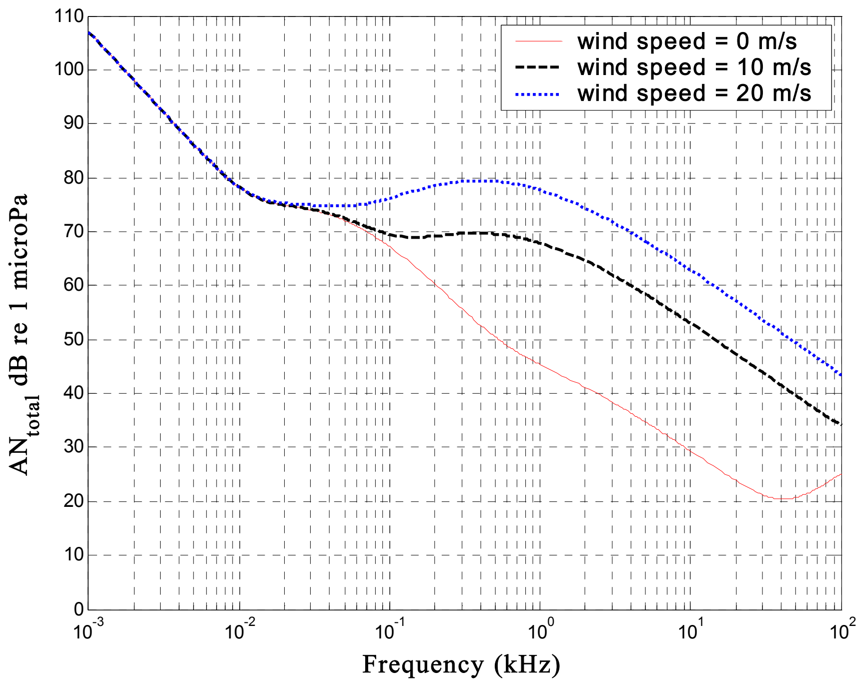


Figure 3. Total Ambient Noise (dB re 1μPa) vs. Frequency (kHz) for Various Wind Speeds [After Ref. 8.]

c. Large-Scale Fading Component ($LSFC_{channel}$)

The sum of all the above factors compose the overall signal degradation caused by the medium to the underwater acoustic channel. This sum is the large-scale fading component in wireless communication nomenclature and is measured in dB re 1 μPa. Analytically:

$$LSFC_{channel} = TL + AN . \qquad (2.8)$$

This intermediate result represents the total channel degradation that the signal must overcome for a given range. Figure 4 below is a plot of the channel SNR as a function of frequency for various ranges.

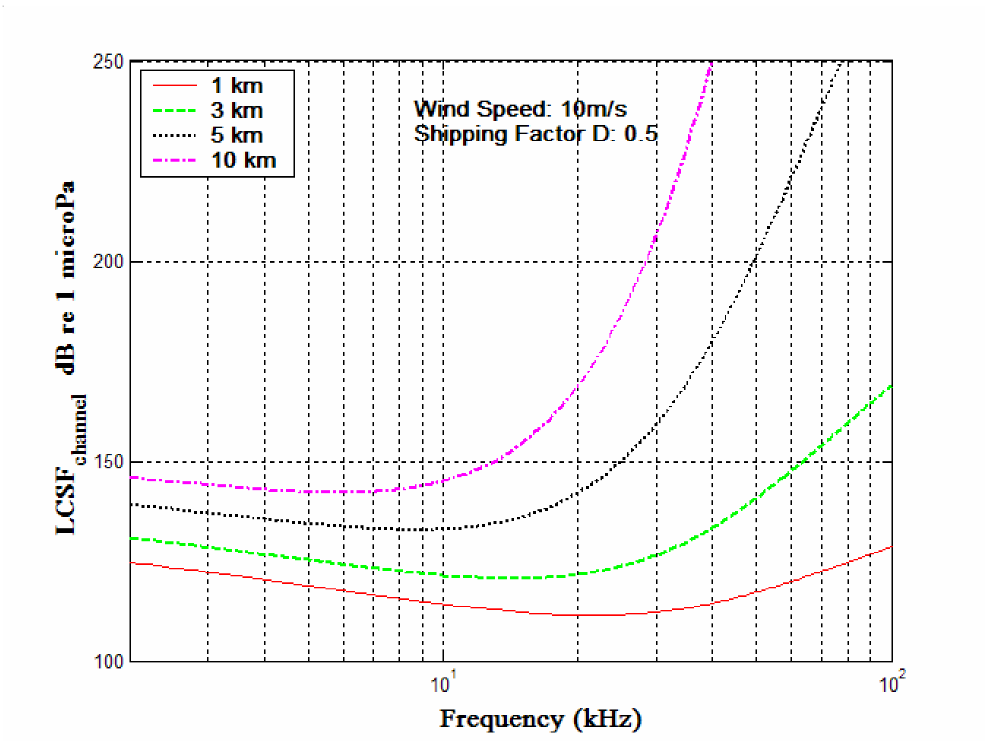


Figure 4. Total Loss due to Medium for Various Ranges [After Ref. 8.]

The large-scale fading component increases with distance, which in turn means that the transmitted signal must compensate for more as the range increases. Furthermore, higher acoustic frequencies are impaired more significantly, implying that the available spectral bandwidth diminishes as a function of source-to-receiver range.

2. Parameters Determined by the Equipment

a. Directivity Index (*DI*)

The directivity index (*DI*) is defined as the ratio of the intensity of a source at a specified direction divided by the intensity at the same reference point by an omni-directional source. In the case of Seaweb, the effect of *DI* is omitted since the transducers now in use are omni-directional.

b. Source Level (SL)

The source level (*SL*) is defined [9] by the following expression:

$$SL \text{ [dB re } P_{ref}] = 20 \cdot \log_{10}\left(\frac{\sqrt{2}/2 \cdot P_0}{P_{ref}}\right) \tag{2.9}$$

where P_0 is the peak acoustic pressure amplitude measured at a distance of 1 m from the source along its acoustic axis. In terms of power, the *SL* can be expressed as [5]:

$$SL\ [dB] = 171.5 + 10 \cdot \log_{10} P_{transmitted} \tag{2.10}$$

where *P* is the power in [W] measured 1 m from the source.

B. SUMMARY AND CONCLUSIONS

The scope of the current chapter was to provide a general insight into the underwater acoustic environment for system design purposes. During the analysis, we observed that despite the similarities to the wireless networks, underwater communications are different from RF communications for several reasons. Firstly, underwater communications use acoustic pressure waves to propagate through the medium instead of electromagnetic waves. Secondly, the underwater medium strongly attenuates frequencies in excess of 30 kHz and therefore the bandwidth available for communications is very small compared to that of RF channels. Thirdly, the ocean noise is non-Gaussian. It is composed of discrete components, which can only be described analytically by "empirical" formulas like shipping noise, turbulence noise, surface noise and thermal noise, and other factors (e.g., industrial, biological, etc.) which are intermittent or can only be taken into account statistically if a large amount of data is available for a specific region of interest.

Under the current link-budget analysis, a practical way of describing the large-scale fading effect in analogy to the cellular community was attempted. Several parameters have been unaccounted for resulting in a rough estimate of the link margin or SNR needed to communicate successfully underwater. If a more accurate estimate of the parameters in the sonar equation is necessary for a specific environment, one should apply an underwater acoustic propagation model to derive the losses. However, the link-budget will prove a handy tool for a general engineering approach to environmental losses and to calculate the required E_b/N_0 to establish communication underwater. The interaction with the medium boundaries and the associated time variability and time spreading are considered in the next chapter.

III. SMALL-SCALE FADING CHARACTERISTICS OF THE UNDERWATER ACOUSTIC CHANNEL

This chapter discusses the small-scale fading effects of the underwater environment. Two different characteristics of the underwater acoustic medium are responsible for the small-scale fading: time spreading and time variability. Time spreading is caused by multipath propagation whereas time variability is the result of movement in the channel. As we see later in the chapter time spreading introduces frequency selectivity of the channel while time variability introduces Doppler spread. It is not uncommon for a real underwater channel to be simultaneously subject to time and Doppler spreading, a property that makes it a most unforgiving medium for communication purposes. Figure 5 below depicts the time varying and spreading processes affecting the signal.

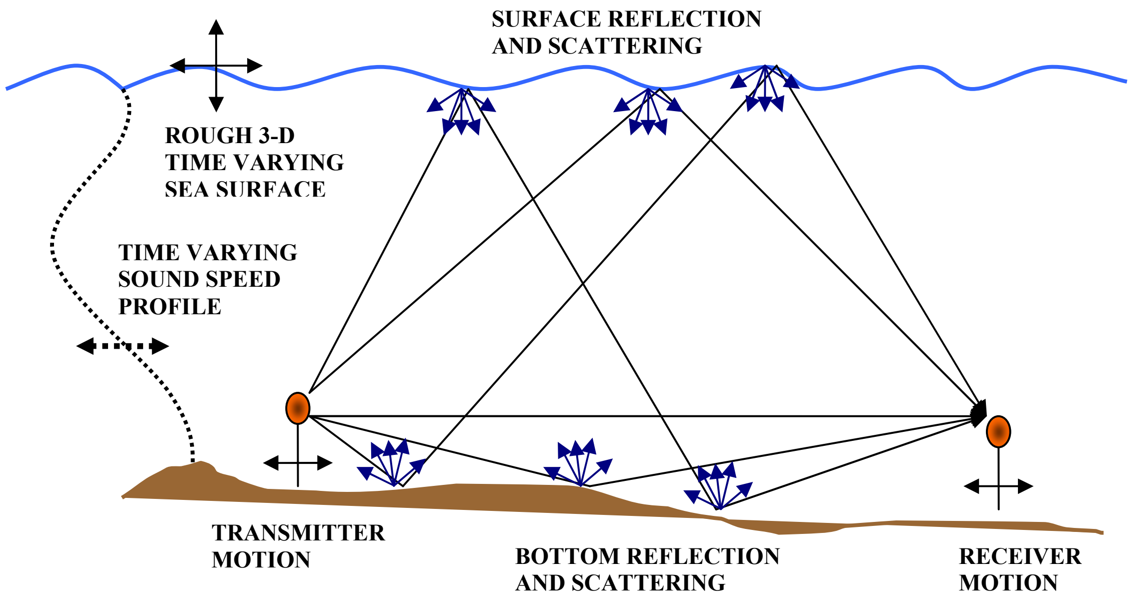

SURFACE REFLECTION AND SCATTERING
ROUGH 3-D TIME VARYING SEA SURFACE
TIME VARYING SOUND SPEED PROFILE
TRANSMITTER MOTION
BOTTOM REFLECTION AND SCATTERING
RECEIVER MOTION

Figure 5. Processes Causing Small-Scale Fading Underwater

An engineering model accounting for these effects is required as an instrument for developing advanced DSSS signaling algorithms.

A. SMALL-SCALE FADING CHARACTERISTICS

1. Multipath

The effect of multipath is that time-delayed echoes of the original signal arrive at the receiver at different times. The delay is caused by reflections from the sea surface and sea bottom. Multipath arrivals cause fluctuations in the received signal strength and the corresponding impulse response will have the general form:

$$h(t) = \sum_{i=1}^{n} w_i(\tau) \cdot \delta(t - \tau_i) \tag{3.1}$$

where w_i are the amplitudes associated with the i-th interaction and δ is the Kronecker delta function used to denote the corresponding time-delayed version of the original signal.

The effect of multipath in the transmitted signal is shown in Figure 6 below. The gross multipath delay is called T_m and represents the duration of the overall impulse response. Values of T_m in excess of ten milliseconds are not uncommon. For a transmitted signal with duration T_s, the received signal is no longer of duration T_s but of duration T_s+ T_m .

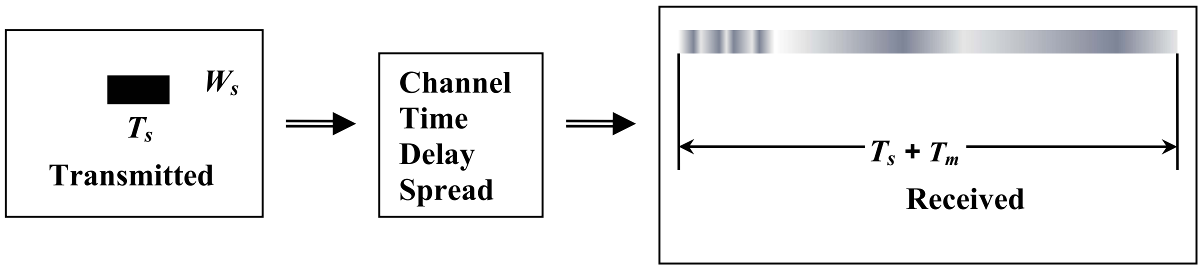


Figure 6. Effect of Time-Delay Spread on the Transmitted Signal

Micropaths exist as a result of the various scatterers in the medium and on the reflective boundaries. The result of the reflection on a momentarily aligned boundary facet is that the unequal micropath lengths induce time spreading, randomize the phase and cause Rayleigh fading on the incident sound signal. Micropaths associated with sea-surface reflection are represented in Figure 7 on the next page.

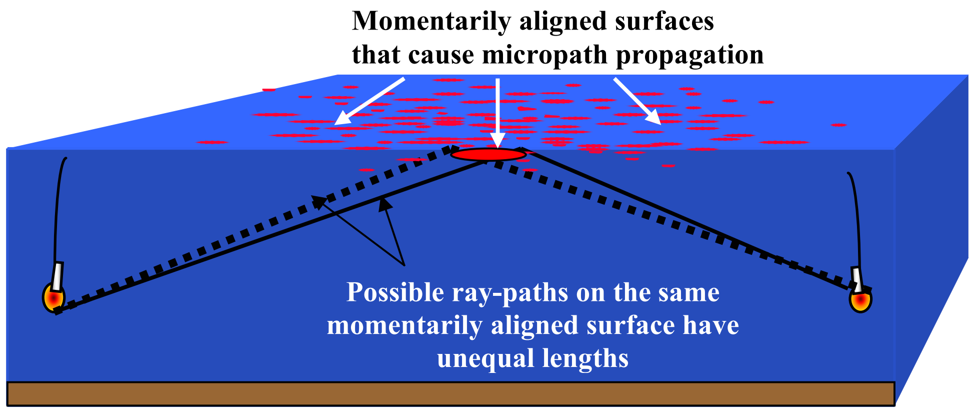

Momentarily aligned surfaces that cause micropath propagation
Possible ray-paths on the same momentarily aligned surface have unequal lengths

Figure 7. Effect of Micropaths on Incident Signal

The phenomenon as seen from the frequency-domain perspective is that the time delay spread introduces a bandwidth $B_c \approx 1/T_m$ (called coherence bandwidth) for which the channel passes all frequency components with equal gain. Within the coherence bandwidth, frequency components are well correlated and the channel transfer function is relatively flat.

Based on the definition of the coherence bandwidth a channel may be characterized as either frequency selective or frequency non-selective based on the comparison of the signal bandwidth W_s to the coherence bandwidth of the channel B_c. Whenever $B_c < W_s$ the channel is called *frequency selective* and significant distortion occurs because the spectral components of the signal are attenuated in a frequency-dependent way. Since the above condition also implies that $T_m > T_s$, successive pulses overlap at the receiver causing inter-symbol interference (ISI). The spectral effect of a frequency selective channel on the signal is shown in Figure 8 on the next page.

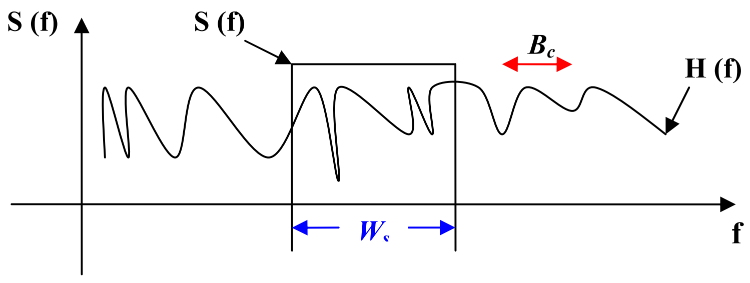


Figure 8. Typical Frequency-Selective Fading When $B_c < W_s$

The channel is characterized as *frequency non-selective* or *flat* when $B_c > W$. In this case the various spectral components are equally affected by the channel. Furthermore, since this also implies that $T_m < T_s$ the channel will not induce ISI in the received signal. The spectral effect of a frequency non-selective channel is shown in Figure 9 below.

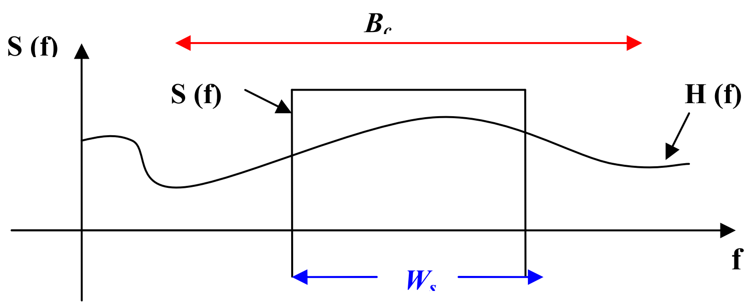


Figure 9. Typical Flat Fading When $B_c > W_s$

2. Time Variability

The time variability in the channel is introduced by the movement of the transmitter or the receiver or the channel itself. The sea-surface boundary is not static. It is the interaction with this boundary, combined with the motion of the source and receiver that accounts for most of the time variability of the real underwater channel. As a result, the impulse response of such a channel is not time invariant. In order to incorporate the time variability with time spreading, Equation (3.1) is rewritten as:

$$h(t,\tau) = \sum_{i=1}^{n} w_i(\tau) \cdot \delta(t - \tau_i(t)) \qquad (3.2)$$

where $w_i(\tau)$ are the amplitudes associated with the *i*-th interaction and $\tau_i(t)$ is the associated time delay τ , which is a function of time.

The gross statistical measures describing the time variability of the underwater channel is its coherence time and Doppler spread. By coherence time T_C, we mean the time duration during which the impulse response can be approximated as time invariant. In the frequency domain, the Doppler spread B_D is the spectral broadening of the signal due to the channel variability. The two phenomena are related by $B_D \approx 1/T_C$. When both the transmitter and the receiver are fixed, the only time variability is caused by the surface wind induced waves and the corresponding surface motion. For telemetry systems the fluctuation of bandwidths is given by [10]:

$$B_D = 2 \cdot f_w \cdot \left[1 + \frac{4\pi f_0 \cos\theta_0}{c} \cdot h_w \right] \tag{3.3}$$

where w is the wind speed in [m/sec], $f_w = 2/w$ is the wave frequency in [Hz], $h_w = 0.005 \cdot w^{5/2}$ is the wave height in [m], f_0 is the carrier (or center) frequency in [Hz], θ_0 is the incident grazing angle in degrees and c the speed of sound in [m/sec].

A Doppler spread channel is characterized according to the relation between the coherence time T_C and the signal duration T_S. A channel is said to be *fast fading* when $T_S > T_C$ or equivalently $B_D > W$ where W is the bandwidth of the signal. Since $W \approx 1/T_S = R_S$, where R_S is the symbol rate, the channel impulse response changes rapidly during the time each symbol is transmitted. On the contrary, a channel is characterized as *slow fading* when $T_S < T_C$ or, equivalently, $B_D < W$ in which case the coherence time of the channel is greater than the transmitted symbol period and, therefore, less distortion is introduced as a result of the propagation in the medium. The effect of Doppler spread on a transmitted signal is shown in Figure 10 on the next page.

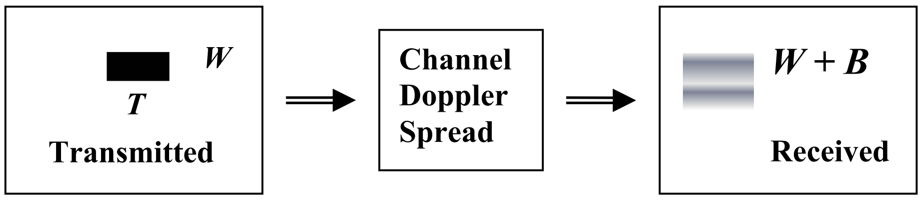


Figure 10. Effect of Doppler Spread on the Transmitted Signal

3. Doubly Spread Channels

A channel can display both time spreading and time variability simultaneously. Based on the relation of the product $B_D \cdot T_m$, called the "spread factor", a channel may be *underspread* if $B_D \cdot T_m << 1$ or *overspread* if $B_D \cdot T_m >> 1$.The most desired case is instinctively the under-spread condition for which a relatively high data rate can be achieved through the use of a coherent modulation method with coherent or differentially coherent detection. The definition of $B_D \cdot T_m << 1$ is always a point of discussion. In the case of Seaweb and the current communication scheme, a spread factor of less than 10^{-3} is desirable. Given the multipath nature of the channel it is sometimes possible to resolve the multipaths effectively by reducing the spread factor.

B. PROPOSED MODEL

Existing underwater acoustic models are static and do not account for the small-scale factors. In such a model, the channel is idealized to impulses placed at lags τ_i with associated amplitudes w_i and is always of the form of Equation (3.1). However, small-scale factors influence high-data-rate underwater communication schemes especially when a coherent scheme is required. The statistics-based model that follows is not intended to produce a high-fidelity underwater acoustics model but is aimed at providing a better simulation tool for analyzing communications performance.

A static impulse response derived by Bellhop is shown in Figure 11 on the next page. The data were supplied courtesy of Dr. Paul Hursky of SAIC and are based on the exact bathythermograph conditions at which the Signalex 2000 sea trial was conducted off the coast of San Diego. More details for the Signalex series experiments can be found in [2]. Figure 11 below displays the impulse response as a function of the source-receiver range.

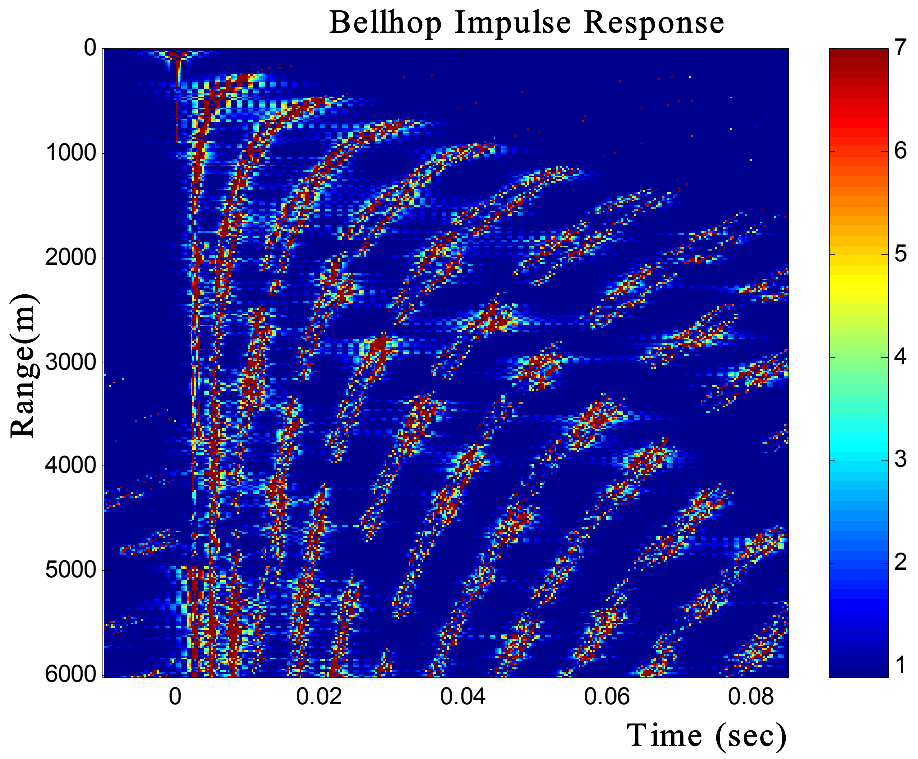


Figure 11. Bellhop-Derived Series 0 to 6 km (Time vs. Range)

The impulse response derived from Bellhop contained some very small amplitudes near the strong direct-path and multipath components. In order to have minimum overlapping when time variability is introduced, amplitudes smaller than 0.03 were filtered out and only the dominant peaks were kept. The impulse response for range $r = 5$ km in magnitude vs. time axes is shown in Figure 12 on the next page:

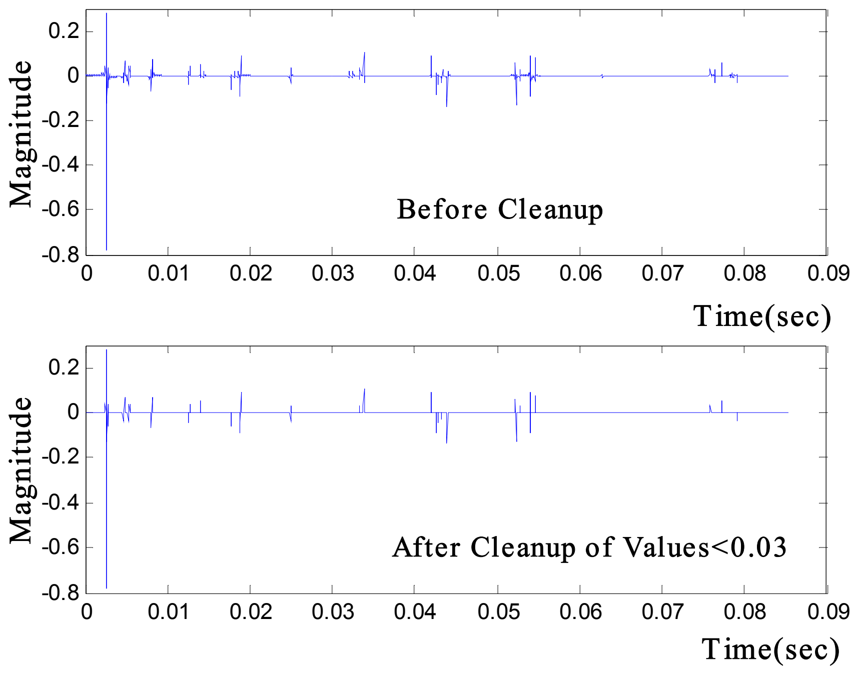


Figure 12. Bellhop-Derived Impulse Response for r = 5 km

We observe that the above impulse responses have exactly the form of Equation (3.1) and the lags τ_i are time invariable. In a time-varying channel, like the one presented in Figure 13 below, the lags τ_i at which the multipath components occur are a function of time t. Therefore, if we snapshot the channel impulse response at different times t we observe different multipath components w_i occurring at lags $\tau_i(t)$.

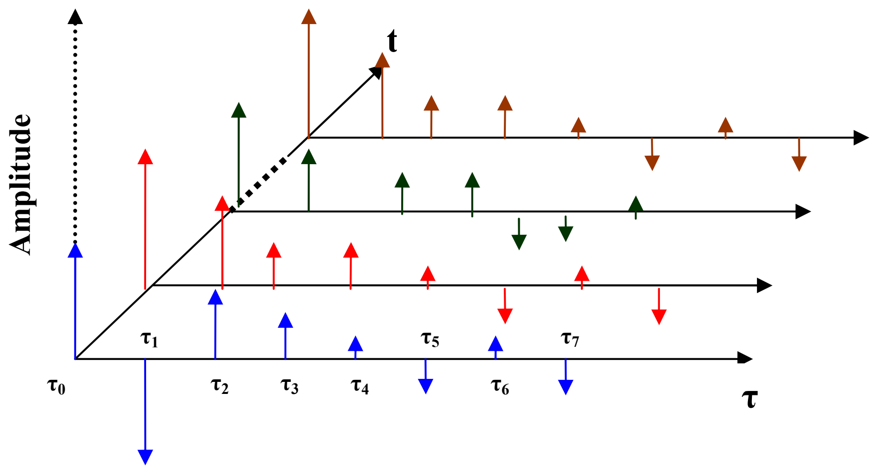


Figure 13. Time Variability of a Cellular Communication Channel [After Ref. 11.]

In order to introduce time variability to an otherwise static impulse response, as the one received from Bellhop, we allowed each of the multipath components to vary randomly within an arbitrary range $\pm$ 3 lags in τ. The amplitudes associated with each lag were not altered at this stage and are the original amplitudes derived by Bellhop. The random phase shift caused by irresolvable micropaths was approximated by adding a uniform random phase factor from [0,2π] to each of the amplitudes. The process was simulated a thousand times in order to get a good statistical sample and the position of each lag was ensemble averaged in time *t*. This process is similar to collapsing the time t axes in Figure 13. The resulting impulse response is time varying, as each time lag τ is now a function of time *t* and possesses a random phase shift. The new impulse response has the form:

$$h(t,\tau)=\sum_{i=1}^{n} w_i(\tau)\cdot\delta(t-\tau_t(t))\cdot\exp(j\phi_i) \tag{3.4}$$

where $w_i(\tau)$ are the amplitudes associated with the *i*-th interaction, $\tau_t(t)$ is the associated time delay τ, which is a function of time, and ϕ_i the uniform phase shift. Furthermore, to better regulate the amount of time spread imposed by the real channel, the delta function was replaced by a Gaussian, except for the direct path component. To avoid rescaling the amplitudes, the factor $1/\sigma\sqrt{2\pi}$ was omitted. The final form of the impulse response is:

$$h(t,\tau)=\sum_{i=1}^{n} w_i(t,\tau)\cdot\exp(j\phi_i)\cdot g_i(t-\tau_i(t)) \tag{3.5}$$

where $g_i(t)=\exp(-x^2/2\sigma_i^2)$ is a Gaussian shaped function with zero mean and σ_i^2 the desired time spread for multipath *i*. Figure 14 on the next page represents the time varying and time spread impulse response for two different values of σ_i^2.

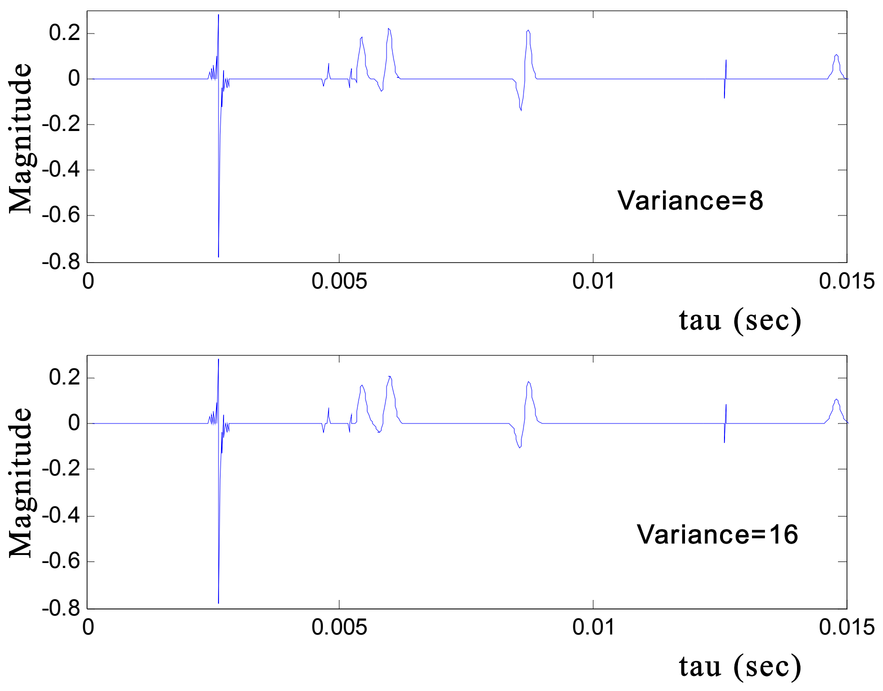


Figure 14. Time-Domain Representation of Doubly Spread Impulse Responses

Based on the previous analysis such a channel is doubly spread and possesses both the time-variability and time-spread features. The effects of such a channel in the frequency domain are omitted because the analysis of the signal processing algorithms described in the next chapter is conducted only in the discrete time domain.

C. LIMITATIONS AND CONCLUSIONS

In this chapter, we have examined the time varying and spreading nature of the underwater medium. It was shown that the small-scale effects are significant when higher rate and coherent underwater communications are desirable. The static impulse response derived by a physics-based model like Bellhop was proven inadequate as an underwater channel simulation tool when the small-scale effects are important. Therefore, a new statistics-based simulation model was developed in this thesis in order to incorporate some of the small-scale effects of the real underwater medium. The new model is intended as a better simulation tool in order to test the receiver structures proposed in the chapters that follow.

The advantages of the proposed model are summarized below:

- The impulse response is time variable.
- Time spreading due to micropaths can easily be manipulated by varying the σ^2 on the Gaussian.
- Random phase shift was introduced in order to simulate micropath propagation
- Spectral components are attenuated differently as a function of time-frequency.

The proposed model has the following limitations:

- It adds statistics on top of the statistics that were used in the first place to derive the Bellhop static impulse response.
- The transmitted signal was an impulse and not a real waveform, which does not fully develop the spectral distortion imposed by a real channel.
- The degree of spread and micropath phase shift may be exaggerated compared to the real channel impulse response.

In the next chapter we discuss the DSSS communication scheme and the proposed blind equalization algorithms to improve the receiver.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. ACOUSTIC SPREAD-SPECTRUM SIGNALING

The previously proposed Seaweb signaling scheme in [1] is based on DSSS. It is DS-IQ-BPSK modulated at the transmitter and processed with a RAKE receiver. Although the transmitter proved to be adequate, the RAKE receiver did not prove to be as robust as anticipated. Therefore, a new receiver structure is needed for efficient equalization. This chapter briefly reviews the necessary theory for spread-spectrum and multi-rate signal processing and then analyzes the existing DSSS signaling scheme. Finally, this chapter examines and evaluates a new receiver structure based on multi-rate signal processing. The analysis emphasizes the signal processing aspect of the receiver and is confined to the discrete time domain. Several cases are evaluated for the blind equalizer based on whether receiver synchronization has been achieved. The actual implementation and integration in the existing DSSS scheme will be the task of future work.

A. SPREAD-SPECTRUM AND MULTI-RATE SIGNAL PROCESSING

1. Direct-Sequence Spread-Spectrum

The choice of a DSSS transmission comes as no surprise, given its resistance to small-scale fading imposed on the received signal by the channel. This is achieved by spreading the signal energy over the widest available bandwidth using a Pseudo-Noise (PN) spreading sequence. This sequence possesses cross-correlation characteristics resembling white noise as much as possible.

A common way to produce a PN sequence is by using an *n*-shift register, with the outputs of each stage combined by a network of exclusive-or (XOR) gates. The output of every XOR gate is fed back to the input of the shift register in an appropriate manner so that the PN generator will cycle through each of the possible states at a random order before any state is repeated. The resulting sequence is called a maximal-length sequence (or “m-sequence”). A serious disadvantage of m-sequences is that their cross-correlations can be quite large compared to the cross-correlation of true random binary sequences. However, a smaller subset of these m-sequences that possesses small cross-correlation characteristics exists, called "preferred m-sequences." A modulo-2 summation of a pair of these latter sequences results in a new pair of preferred m-sequences that possess the desired cross-correlation characteristics. Finally modulo-2 summing one of a pair of preferred m-sequences with *n*-cyclically shifted versions of the other results in a Gold code. The interested reader can find more information on PN sequences and Gold codes in Ref. [12]. An example on the generation of a Gold code from two preferred m-sequences is illustrated in Figure 15 below:

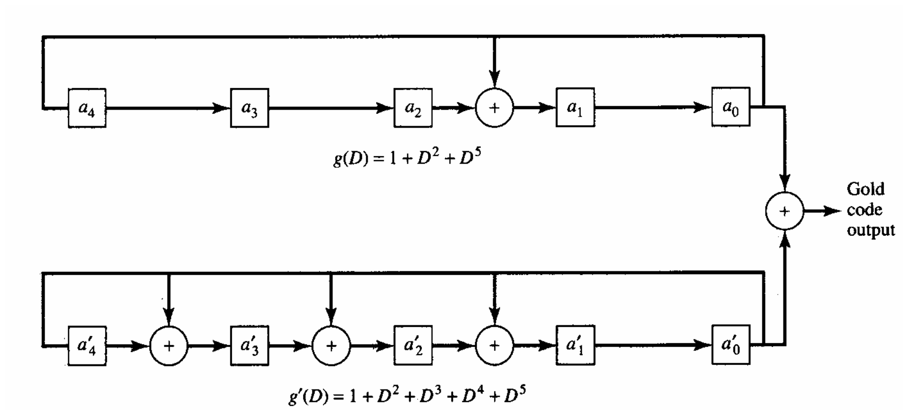


Figure 15. An Example of a Gold-Code Generator [From Ref. 12.]

A PN sequence has a fixed length of P chips. If we let T_{CH} be the pulse duration of the chipping sequence the PN sequence, is periodic with period $P \cdot T_{CH}$. Furthermore, if T_S is the signal pulse duration, the ratio of chips to signal is called the spreading gain k where

$$k = P = \frac{T_S}{T_{CH}}. \tag{4.1}$$

Moreover, if we define the chipping rate as the inverse of T_{CH} and the symbol rate as the inverse of T_S , we have the following relations:

$$R_C = \frac{1}{T_{CH}} \tag{4.2}$$

and

$$R_S = \frac{1}{T_S} . \tag{4.3}$$

Therefore, a chipping sequence of a chipping rate R_S produces $P = R_C / R_S$ chips/symbol. If we combine the three previous relations we can write:

$$k = P = \frac{T_S}{T_{CH}} = \frac{R_C}{R_S}. \tag{4.4}$$

The spreading is done at the baseband by multiplying the baseband data waveform with the PN sequence (also called the chipping sequence). In the receiver, the received signal is despread using a synchronized replica of the transmitter's PN sequence. By using this technique, the channel induced interference is not despread along with the encoded data sequence because it possesses different correlation characteristics than the data sequence. Therefore the different multipath arrivals are rejected and the original signal is reconstructed as it "floats" over the noise. The effects of wideband interference on a DSSS signal are shown in Figure 16 below.

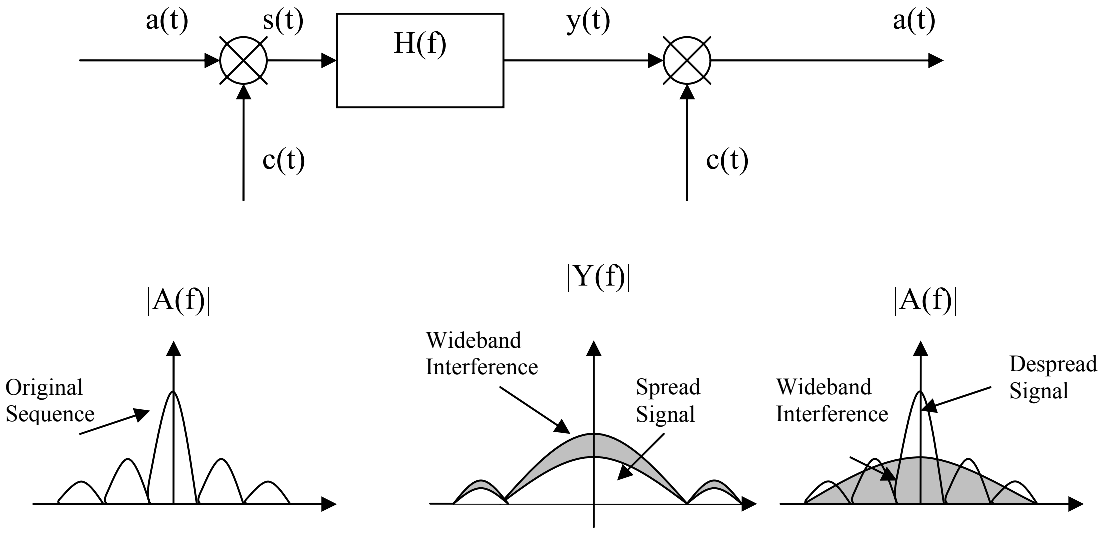


Figure 16. Wideband Interference on the DSSS Signal [After Ref. 1.]

A chipping sequence can be characterized as "short" when the entire sequence is transmitted within every data bit or "long" when only a portion of the sequence is transmitted within each bit. In the original Seaweb modem implementation, a long Gold code was used whereas in the proposed receiver structures a short code of length N proved to give better results.

2. Interpolation by an Integer Factor *I* (Upsampling by *I*)

In many practical applications, changing the sampling frequency of a given signal, either to increase it or decrease it, is desirable. The terms "interpolation" and "upsampling" are used interchangeably in signal processing to denote a desired increase in the sampling frequency f_S, usually by an integer factor *I*. The analysis that follows is based on [13] and will generally retain the same symbols.

If we let F_X represent the initial sampling frequency and F_Y represent the increased sampling frequency, the upsampling operation can be expressed mathematically as:

$$\frac{F_Y}{F_X} = I \Leftrightarrow F_Y = F_X \cdot I\,. \tag{4.4}$$

Moreover, since the sampling interval is equal to the inverse of the sampling frequency or $T_S = 1/f_S$ the operation can be rewritten in terms of the sampling interval as:

$$F_Y = F_X \cdot I \Leftrightarrow \frac{1}{T_Y} = \frac{1}{T_X} \cdot I \Leftrightarrow T_X = T_Y \cdot I \tag{4.5}$$

where T_X and T_Y are the sampling intervals corresponding to the initial and to the increased sampling frequencies F_X and F_Y, respectively. In other words, this means that the sampling interval of the increased sequence should be *I* times smaller than the initial. This operation is usually done by adding $I-1$ zeros between samples. Suppose the signal we desire to upsample is the discrete-time sequence $x[n]$. If we let the desired sequence at a higher sampling rate be $y[n]$, the upsampling process can be represented by the block diagram in Figure 17 below for the specific case of interpolating by three.

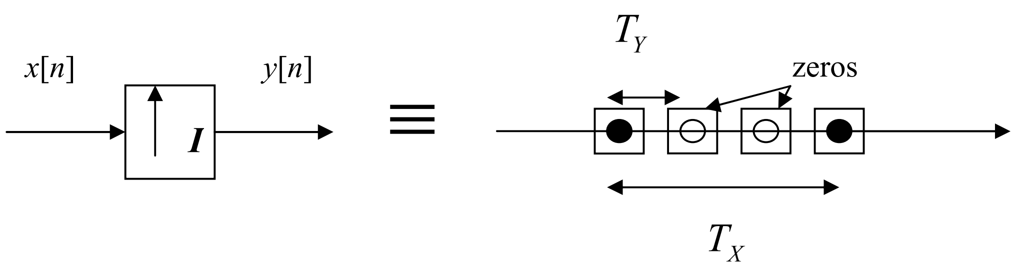


Figure 17. Upsampling by an Integer Factor *I* in the Time Domain

The upsampled sequence in the time domain retains all of the initial information. However, artifacts are generated by the additions of the zeros, which are evident in the frequency domain. Because the zeros were inserted in every $I-1$ sample the sequence $y[n] = y[m \cdot I]$ has non-zero terms only in every m samples. The z-transform of the upsampled sequence is

$$Y(z) = \sum_{m=-\infty}^{\infty} \underbrace{y[m \cdot I]}_{x[m]} z^{-mI} = \sum_{m=-\infty}^{\infty} x[m] z^{-mI} = X(z^{I}) \qquad (4.6)$$

where $x[m]$ is the initial sequence. If evaluated on the unit circle, the spectrum of the upsampled sequence $Y(\omega_{\mathrm{Y}})$ is

$$Y(\omega_{\mathrm{Y}}) = \mathrm{X}(\omega_{\mathrm{Y}} \cdot I) \qquad (4.7)$$

where ω_{Y} is the radian frequency corresponding to the increased sampling frequency. Furthermore, the relation between ω_{Y} and ω_{X} (the initial radian frequency) is

$$\omega_{\mathrm{Y}} = \frac{\omega_{\mathrm{X}}}{I}. \qquad (4.8)$$

The effect of the upsampling operation in the Fourier domain is shown in Figure 18 on the next page for the particular case of upsampling by two:

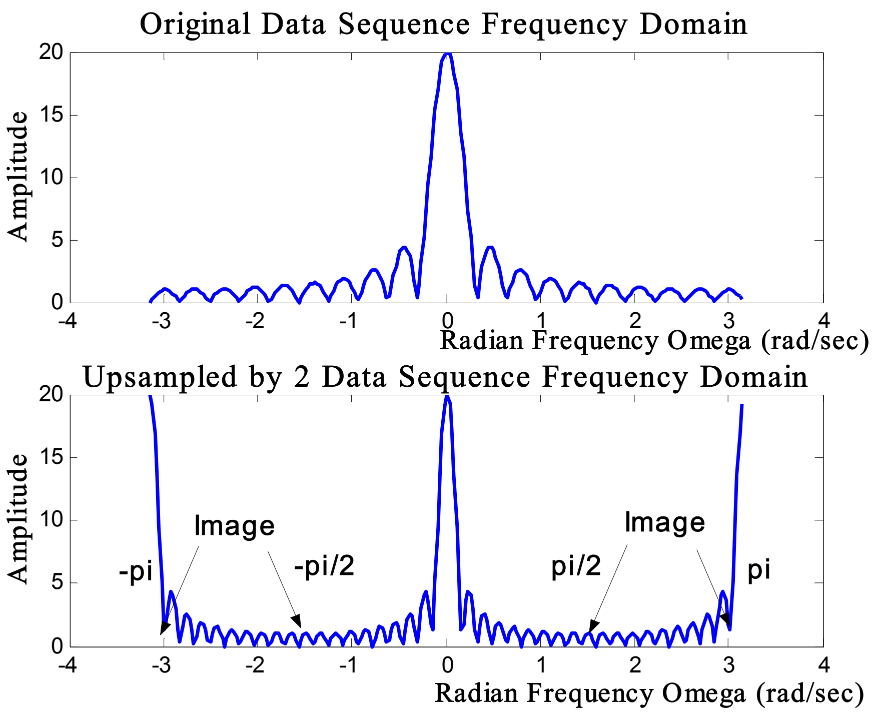


Figure 18. Spectral Effect of Upsampling

We observe that the initial spectrum is squeezed by I ($I = 2$ in the case of Figure 18). Therefore, if the spectrum of the initial sequence lay between $-\pi$ and π, the spectrum of the upsampled sequence is squeezed within the interval $-\pi/I$ and π/I. As a consequence, image frequencies appear in the intervals $[-\pi, -\pi/I]$ and $[\pi/I, \pi]$ which are artifacts created by the process. Fortunately, the image frequencies can be easily eliminated with Low Pass Filtering after upsampling.

3. Decimation by an Integer Factor *D* (Downsampling by *D*)

Decimation is the inverse process of interpolation. In downsampling, we decrease the sampling frequency of a signal by an integer factor *D*.

If we let F_X be the initial sampling frequency and F_Y be the decreased sampling frequency, the downsampling operation can be expressed analytically as:

$$\frac{F_Y}{F_X} = \frac{1}{D} \Leftrightarrow F_Y = F_X \cdot \frac{1}{D}, \tag{4.9}$$

or in terms of the sampling interval:

$$F_Y = F_X \cdot \frac{1}{D} \Leftrightarrow \frac{1}{T_Y} = \frac{1}{T_X} \cdot \frac{1}{D} \Leftrightarrow T_X = T_Y \cdot \frac{1}{D} \tag{4.10}$$

where T_X and T_Y are the sampling intervals corresponding to the initial and the decreased sampling frequencies F_X and F_Y, respectively. The most usual representation is shown in Figure 19 below for the specific case of decimating by three:

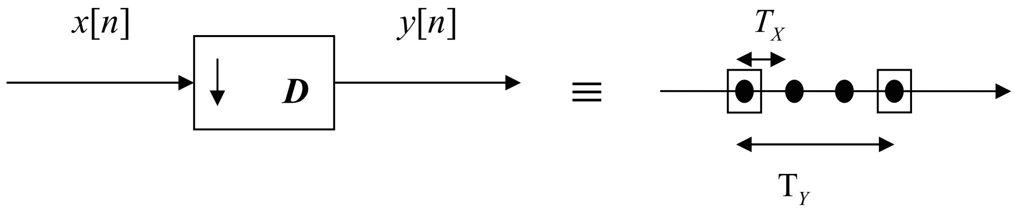


Figure 19. Downsampling by an Integer Factor D in the Time Domain

In the time domain decimation can be thought of as retaining one for every $D-1$ samples and information is irretrievably lost during this process.

As expected, downsampling creates aliasing in the frequency domain. In fact Ref. [13] shows that the downsampled sequence has a corresponding spectrum $Y(\omega_Y)$ as shown in the following equation:

$$Y(\omega_Y) = \frac{1}{D}\sum_{k=0}^{D-1} X\left(\frac{\omega_Y}{D} - k\frac{2\pi}{D}\right) \tag{4.11}$$

where ω_Y is the radian frequency corresponding to the decreased sampling frequency. Moreover, the relation between the increased radian frequency ω_Y and the initial radian frequency ω_X is

$$\omega_Y = D \cdot \omega_X . \tag{4.12}$$

The effect of the downsampling operation in the frequency domain is shown in Figure 20 below for the particular case of downsampling by two:

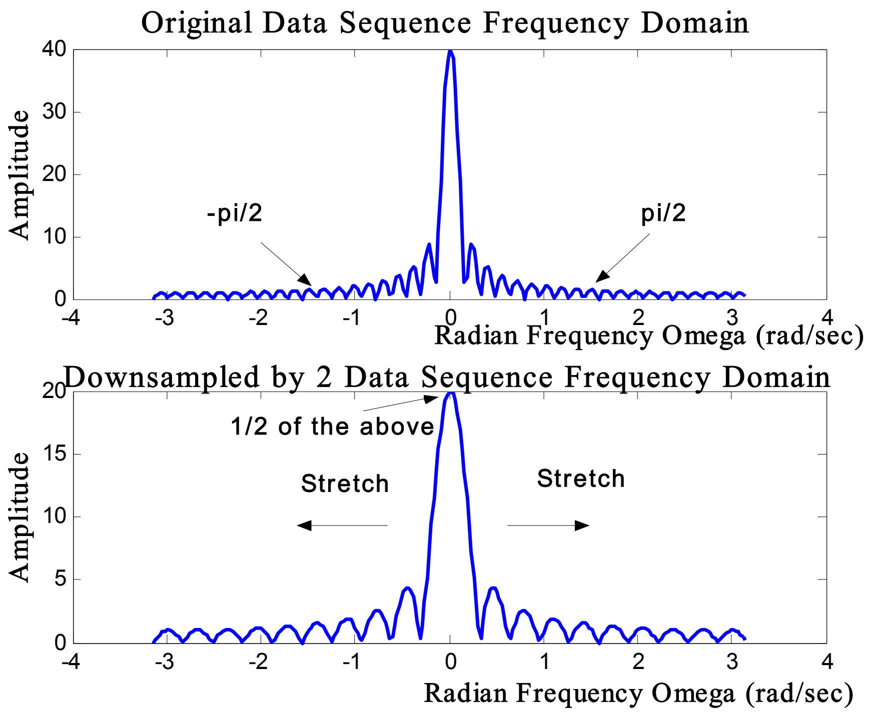


Figure 20. Spectral Effect of Downsampling

We observe that the initial spectrum is stretched by D. Therefore, if the spectrum of the initial sequence lay within the interval $-\pi/D$ and π/D, the spectrum of the downsampled sequence is stretched within the interval $-\pi$ to π. The immediate consequence is that frequencies below $-\pi/D$ or above π/D of the initial sequence are folded back and aliased. In Figure 20, aliases were eliminated by low-pass filtering the original sequence in the interval $[-\pi/2, \pi/2]$ before downsampling.

4. Upsampling and Spreading

As seen previously, the spreading of a signal is done at the baseband by multiplying the baseband data waveform with the PN sequence [12]. Since all of the analyses that follow are done in the discrete time domain and the signals involved are impulses, the effect of spreading the data sequence is equivalent to multiplying each bit of the data sequence with the PN sequence. From a signal processing point of view, this is equivalent to upsampling by P and then convolving with the spreading code. The scope of the following Figure 21 is to demonstrate the equivalence between the two claims for a single bit.

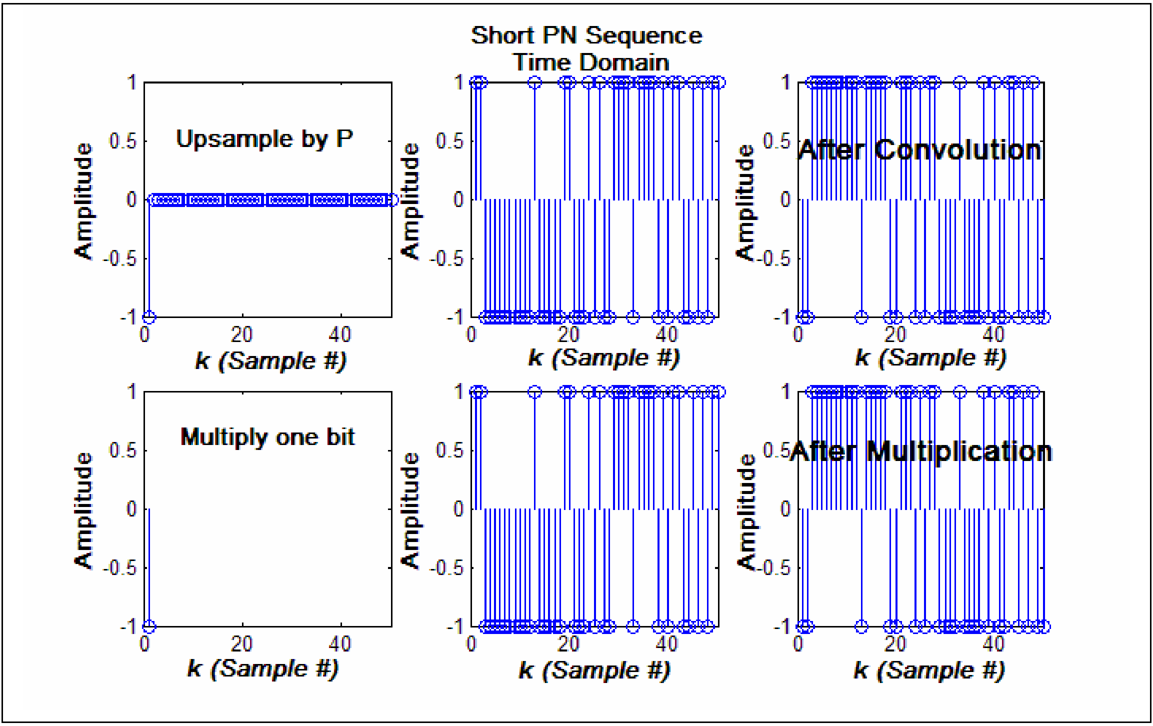


Figure 21. Upsampling and Spreading a Sequence

As we can observe, the result from both methods is identical. Following the upsampling with zeros and convolution approach, each bit is appended with the PN sequence. Equivalently following the multiplication approach, each bit appears to be spread throughout the PN sequence.

B. PREVIOUSLY PROPOSED DSSS SIGNALING SCHEME

In the previously proposed DSSS signaling scheme, a DSSS technique is used to chip and to spread a data bit sequence, which has been modulated by a Differentially-encoded Binary Phase-Shift Keying (DBPSK) carrier. The process is depicted in Figure 22 on the next page.

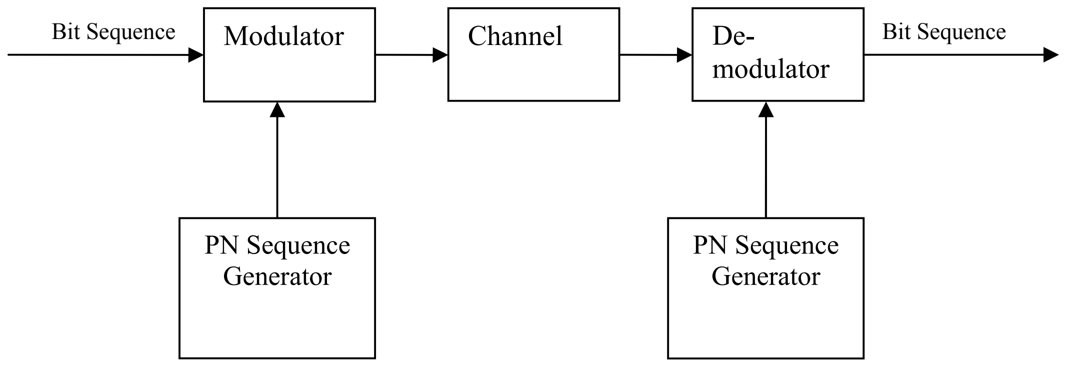


Figure 22. Previously Proposed DSSS Scheme [After Ref. 1.]

The transmitter scheme was implemented as part of a previous thesis [1]. The main problem in the previous implementation was the RAKE receiver. Although this type of receiver is used in cellular communications, it did not perform as anticipated underwater.

The RAKE receiver is based on the principle that multipath arrivals separated by time intervals greater than the coherence time of the channel T_C can be resolved and summed. This technique is also referred to as the "integrate-and-dump" method and the interested reader can find more details in [11]. There are two major problems associated with this technique. First, there is the need to estimate the amplitudes of the equalizer continuously over a time-varying channel which incurs a great computational expense. Moreover, in the case that the channel is fast fading, there is no guarantee that the equalizer will converge before the channel changes again. Secondly, when in any processing leg of the structure there is no signal present, the input is due to the noise only. However, the latter is resolved by selecting an appropriate threshold. The complete structure of the RAKE receiver is shown in Figure 23 on the next page.

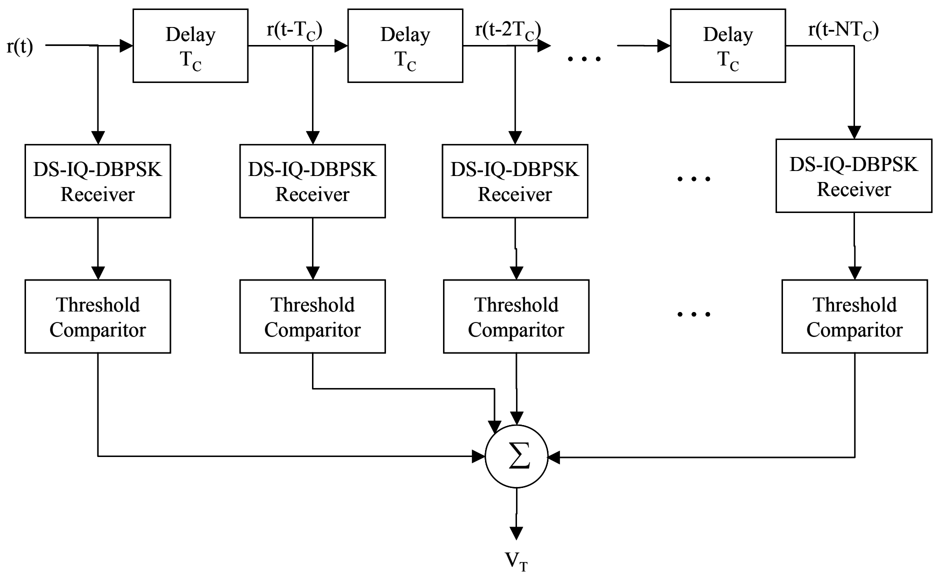


Figure 23. Block Diagram of a RAKE Receiver for DS-IQ-DBPSK [From Ref.1.]

The present thesis follows a different approach in the equalization process, in which equalization is obtained by subspace-decomposition.

C. PROPOSED DSSS SCHEME

1. Simplified Transmitter Structure

Two blind equalization algorithms, based on multi-rate signal processing and subspace-decomposition for DS-CDMA, are evaluated in this thesis for various interference and synchronization assumptions. For the sake of simplicity, the transmitted signal is assumed to be a bit sequence, which is then chipped and transmitted through the channel. All of the analysis that follows is done in the discrete time domain. Channel encoding and modulation are not implemented in this thesis, but they are assumed to improve the signaling scheme. Incorporating the simplified transmitter and receiver structures used in this thesis in the actual Seaweb scheme will be the task of future work. The simplified transmitter used throughout the remainder of the thesis is as shown in the following block diagram in Figure 24 on the next page.

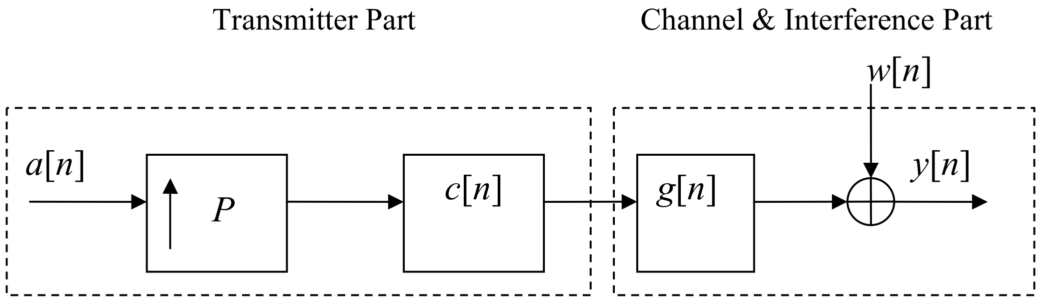


Figure 24. Simplified Multi-Rate DSSS Transmitter

The data sequence $a[n]$ is first upsampled by $P = T_S / T_C$ and then chipped by the chipping sequence $c[n]$. The chipping sequence is a short PN code of length *P*. Essentially, this means filtering the upsampled data sequence $a[n]$ by the *P*-th order filter:

$$C(z) = c[0] + c[1]z^{-1} + ... + c[P-1]z^{-(P-1)}. \tag{4.13}$$

The state space equivalent transmitter structure is represented in Figure 25 below:

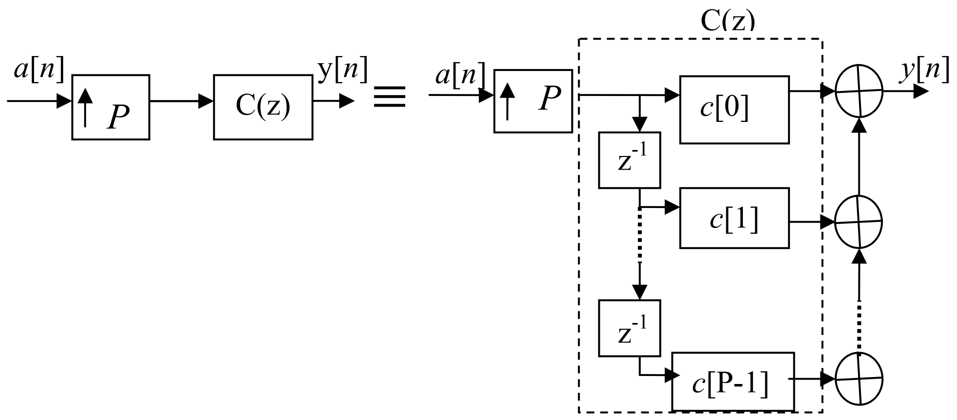


Figure 25. State-Space Representation of CDMA Transmitter

Subsequently, the chipped signal is convolved with the channel impulse response *g*[*n*]. The channel is assumed to be of length *M* < *P* but, as will be shown in the next chapter, the multi-rate technique was proven robust even when the whole impulse response of the underwater channel was used. The corruption by the channel is equivalent to filtering by the *M*-th order linear filter:

$$G(z) = g[0] + g[1]z^{-1} + ... + g[M-1]z^{-(M-1)}. \tag{4.14}$$

Finally, noise is added to account for random interference. As we will see later in the chapter, the noise will be both Gaussian white noise and colored noise, in order to evaluate the receiver's behavior in a simulated adverse environment before using the time-variable impulse response.

2. Blind Equalizer Structure and Matched Filtering

The receiver structures that follow are considered blind equalization methods. By the term "blind equalization," we mean that the receiver has no a priori knowledge of what the transmitted sequence nor the corrupting channel is but attempts to undo the channel effect, as well as adjusts the filter coefficients based on statistical methods applied to the received signal. The receiver structure is based on a DS-CDMA algorithm from Ref. [14] and is adopted for a single user. The matched filter analysis has been adapted from Ref. [15]. Such algorithms have been described for cellular communications and are simulated in the current thesis for the underwater medium. The blind equalization algorithms presented in this thesis are intended for *off-line* use at the present stage since the entire received sequence must be available in order for the algorithms to work. Moreover, the synchronization between the receiver and the transmitter is assumed to be achieved using the current Seaweb synchronization methods.

a. Additive Gaussian White Noise (AWGN) without Multipath

In this case, the $g[n]$ portion of the channel is omitted from the transmitter in Figure 24. The data sequence is upsampled to the chip rate by P and then multiplied by $c[n]$, which has the same effect as appending a code to each bit. The resulting sequence is transmitted and Gaussian noise is added. Therefore, the received sequence $y[n]$ is simply the chipped sequence with added Gaussian white noise and the resulting vector length does not change.

The receiver structure for the white noise case is shown in Figure 26 on the next page. The received sequence $y[n]$ is subdivided into P vectors of length equal to the original data sequence. This is achieved by downsampling the original sequence by P to obtain the first vector, then forward-delaying by one sample and downsampling by P

to obtain the next vector. The process is repeated P times in total. The optimum filter is derived from the resulting matrix $\underline{y}[n]$.

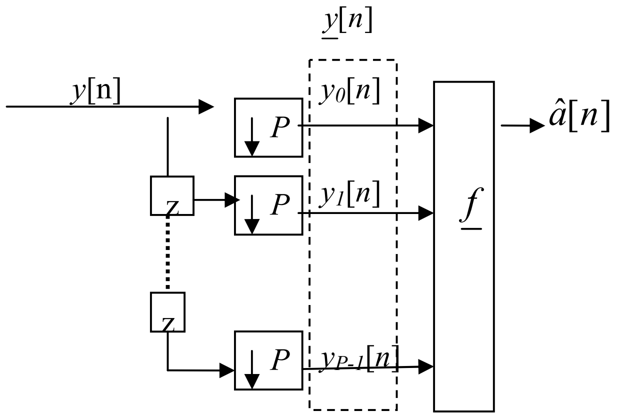


Figure 26. Multi-Rate CDMA Receiver for AWGN

To gain more insight for the algorithm, we combine the transmitter and the receiver in the manner shown in Figure 27 below:

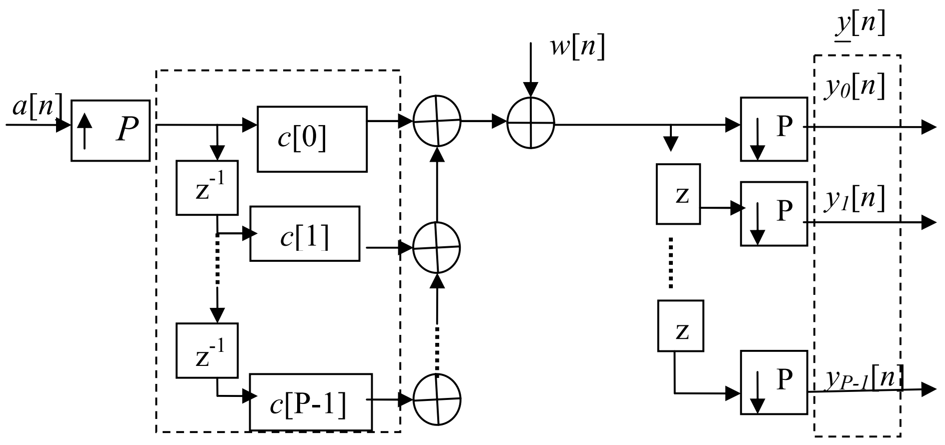


Figure 27. Combined Transmitter-Receiver for AWGN

The receiver structure seems non-causal as it forward-delays and downsamples the received sequence. However, this is not the case. Suppose we select the *k-th* branch from the transmitter part and the q-th branch from the receiver (where k, q= 0, 1, …, $P-1$) as shown in Figure 28 on the next page:

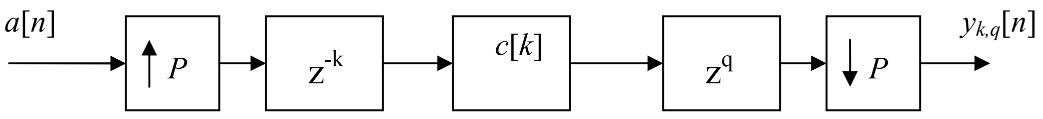


Figure 28. Multi-Rate Representation of a Branch

If we combine the two delay terms into one and observe that there is no cross-talk the branch is simplified to the following Figure 29:

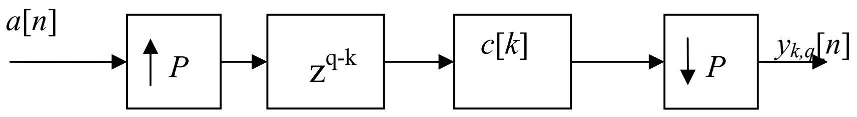


Figure 29. Simplified Multi-Rate Representation of a Branch

Since there is no cross-talk, $y_{k,q}[n] = \begin{cases} 0, & k \neq q \\ c[k]a[n], & k = q \end{cases}$ and the $q-k$ exponent reduces to zero when $k \neq q$. Therefore, the branch is causal and $y_{k,q}[n]$ reduces to $y_k[n] = c[k]a[n] + w_k[n]$. Given that, for every $k = q$ only one branch is non-zero, we can generalize for the whole system:

$$\underline{y}[n] = \underline{c} \cdot a[n] + \underline{w}[n] \tag{4.15}$$

where the vectors $\underline{y}[n]$ and $\underline{c}$ are defined as: $\underline{y}[n] = \begin{bmatrix} y[nP] \\ \vdots \\ y[nP+P-1] \end{bmatrix}$ and $\underline{c} = \begin{bmatrix} c[0] \\ \vdots \\ c[P-1] \end{bmatrix}$.

Using the previous equation and defining $\hat{a}[n]$ to be the estimate of $a[n]$ (the transmitted signal) then

$$\hat{a}[n] = \underline{f}^T y[n] = \underline{f}^T (\underline{c} \cdot a[n] + \underline{w}[n]) \tag{4.16}$$

where $\underline{f}^T$ is the desired filter (estimate of the channel), as shown in Figure 26, which is yet to be computed. The best estimate for $\underline{f}$ is the one that maximizes the SNR in analogy with the matched filter. The matched filter is described by the following equation:

$$SNR = \frac{E\left\{ | \underline{f}^T \underline{c} \cdot a[n] |^2 \right\}}{E\left\{ | \underline{f}^T \underline{w}[n] |^2 \right\}}. \tag{4.17}$$

If we assume $a[n]$ to be stationary, and $a[n]$ and $\underline{w}[n]$ independent, $E\left\{ | a[n] |^2 \right\}$ reduces to $\sigma_\alpha{}^2$. Furthermore, since $E\left\{ | \underline{f}^T \underline{w}[n] |^2 \right\} = \underline{f}^T R_{ww} \underline{f} = \underline{f}^T \sigma_w I \cdot \underline{f} =$ $\sigma_w \underline{f}^T \underline{f}$, the SNR can be rewritten:

$$SNR = \frac{\sigma_\alpha{}^2 | \underline{f}^T \underline{c} \cdot \underline{c}^T \underline{f} |}{\sigma_w{}^2 \underline{f}^T \underline{f}}. \tag{4.18}$$

The SNR is maximized when $\underline{f} = \lambda \cdot \underline{c}$ with λ any scalar constant. Therefore, the desired filter in the white noise case is

$$\underline{f} = \lambda \cdot \underline{c}. \tag{4.19}$$

b. Multipath with AWGN

The transmitter block diagram for the multipath and additive white-noise case is as shown in Figure 24. The data sequence is upsampled to the chip rate by *P*, spread by a short PN sequence and then convolved (corrupted) with the channel impulse response $g[n]$. Finally, white noise is added to compensate for random interference. The receiver structure is slightly different than the previous case. The received sequence is first despread using a synchronized replica at the receiver and then forward delayed and downsampled .The previous analysis on the causality of this process is still valid and will not be repeated. The receiver block diagram is shown in Figure 30 on the next page.

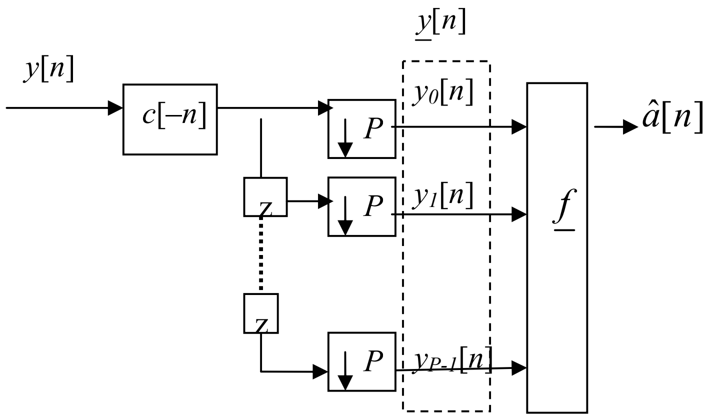


Figure 30. Receiver Structure for Multipath with AWGN

In Figure 30 above the despreading sequence shown as $c[-n]$ is such that $c[n]*c[-n]=\delta[n]$. The combination of transmitter-receiver is shown in Figure 31 below:

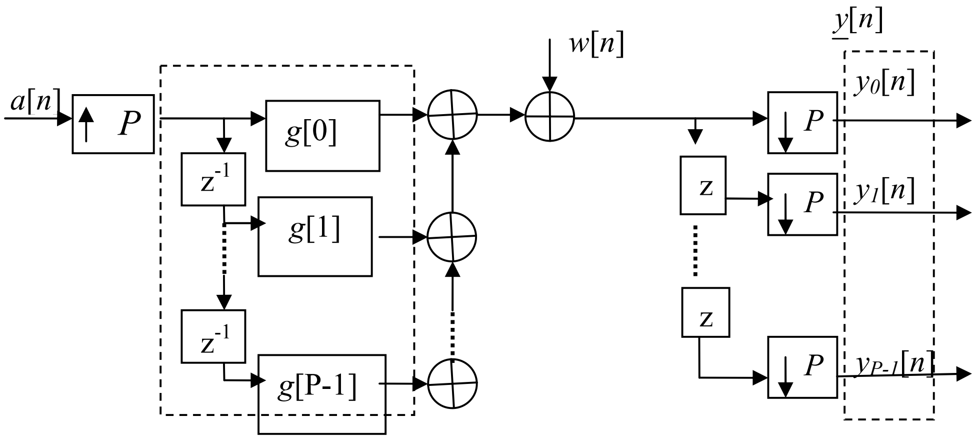


Figure 31. Combined Transmitter-Receiver for Multipath with AWGN

Following the previously laid methodology for the white noise case the matrix $\underline{y}[n]$ is composed by P versions of the original data sequence, which has been corrupted by the channel. This can mathematically be expressed as:

$$\underline{y}[n]=\underline{g}\cdot a[n]+\underline{w}[n] \tag{4.20}$$

where $\underline{g}$ is the unknown channel corruption and $\underline{w}[n]$ the additive white noise. Since the equalization method is blind, the signal sequence is also unknown. However, we can make an estimate of $a[n]$ called $\hat{a}[n]=\underline{f}^T\underline{y}[n]=\underline{f}^T\left(\underline{g}\cdot a[n]+\underline{w}[n]\right)$ provided we can estimate $\underline{f}^T$ (the desired filter). Following the same methodology as previously we will select the $\underline{f}^T$ that maximizes the SNR in analogy with the matched filter. The matched filter, however, presupposes knowledge of the transmitted signal, which in our case is unknown. To overcome that, we use a different metric of SNR called SNR$_{+1}$ [16] in order to make use of the received sequence $y[n]$. The new matched filter is described by the following equation:

$$SNR_{+1} = \frac{E\left\{|\underline{f}^T\underline{g}\cdot a[n]|^2\right\}}{E\left\{|\underline{f}^T\underline{w}[n]|^2\right\}} + 1 = \frac{E\left\{|\underline{f}^T\underline{y}[n]\underline{y}^T[n]\underline{f}|\right\}}{E\left\{|\underline{f}^T\underline{w}[n]\underline{w}^T[n]\underline{f}|\right\}} = \frac{\underline{f}^T R_{yy}\underline{f}}{\underline{f}^T R_{ww}\underline{f}}. \tag{4.21}$$

Since $\underline{f}^T R_{ww}\underline{f} = \underline{f}^T\sigma_w I\cdot\underline{f} = \sigma_w\underline{f}^T\underline{f}$ the SNR_{+1} can be rewritten:

$$SNR_{+1} = \frac{\underline{f}^T R_{yy}\underline{f}}{\sigma_w^{\,2}\underline{f}^T\underline{f}}. \tag{4.22}$$

In order to maximize the SNR_{+1} we set the denominator to one and maximize the numerator. Therefore, we choose $\underline{f}$ so that it maximizes the quantity:

$$\mathfrak{L} = \underline{f}^T R_{yy}\underline{f} + \lambda(1-\sigma_w\underline{f}^T\underline{f}) \tag{4.23}$$

where λ is a real Lagrange multiplier. The maximum for R_{yy} can be obtained by taking the gradient with respect to $\underline{f}$ and setting it to zero:

$$\nabla_f\,\mathfrak{L} = R_{yy}\underline{f} - \lambda\sigma_w\underline{f} \Leftrightarrow R_{yy}\underline{f} = \lambda\sigma_w^{\,2}\underline{f} \Leftrightarrow R_{yy}\underline{f} = \lambda'\underline{f}, \tag{4.24}$$

which is an ordinary eigenvalue problem and $\underline{f}$ is the eigenvector that corresponds to the maximum eigenvalue $\lambda'_{\max}$ of the autocorrelation matrix $R_{yy} = E\left\{\underline{y}[n]\underline{y}^T[n]\right\}$. Applying the matched filter to the received signal yields:

$$\hat{a}[n] = (\underline{f})^T\underline{y}[n]. \tag{4.25}$$

c. *Multipath with Additive Colored Noise*

In the case that the added noise is colored, the receiver would still have the same structure as previously (Figures 30 and 31) but the derivation of the optimum filter $\underline{f}$ would have to account for the fact that the noise is not white but colored. In this case the autocorrelation matrix of the noise is not diagonal and the off-diagonal terms will possess non-zero values. If we let R_{ww} be the new autocorrelation matrix for colored noise, the eigenvalue decomposition of R_{ww} would yield:

$$R_{ww} = E_w \Lambda_w E_w^{\,T} \tag{4.26}$$

where E_w is the eigenvector matrix and Λ_w is the eigenvalue matrix of R_{ww}. Using the Mahalanobis transformation:

$$\underline{y}'[n] = R^{-\frac{1}{2}}{}_{ww} \cdot \underline{y}[n] = (E_w \Lambda_w^{-\frac{1}{2}} E_w^{\,T}) \cdot \underline{y}[n], \tag{4.27}$$

the autocorrelation matrix R_{ww} of the colored noise is transformed into a new coordinate system in which the noise can be considered white. Applying the coordinate transformation (4.23) on the original autocorrelation matrix R_{yy} yields:

$$R'_{yy} = R^{-\frac{1}{2}}{}_{ww} R_{yy} R^{-\frac{1}{2}}{}_{ww}\,. \tag{4.28}$$

Then applying Equation (4.21) to the new coordinate system, we can write:

$$R'_{yy}\underline{f}' = \lambda \underline{f}'\,, \tag{4.29}$$

which essentially means that the problem reduces to a regular eigenvector problem in the new coordinate system and $\underline{f}'$ is the eigenvector that corresponds to the maximum eigenvalue $\lambda_{\max}$ of the autocorrelation matrix R'_{yy} on the new coordinate system. Applying the matched filter in the original coordinate system is done by:

$$\hat{a}[n] = \left(\underline{f}'\right)^T \underline{y}\,[n] = \left(\underline{f}'\right)^T R_{ww}^{-\frac{1}{2}}\, y[n] = \underline{f}\,\underline{y}[n] \tag{4.30}$$

where the optimum filter in the old coordinate system is

$$\underline{f} = \left(\underline{f}'\right)^T R_{ww}^{-\frac{1}{2}}. \tag{4.31}$$

d. Loss of Receiver Synchronization

The foundation for the formerly developed receiver algorithm is that in order to remove the channel effect adequately, synchronization between the transmitter and receiver is imperative. Due to synchronization there is only one dominant eigenvector of the autocorrelation matrix R_{yy}, which belongs to the largest eigenvalue. However, synchronization might not always be possible in the adverse underwater medium. In the present analysis, we will examine a possible remedy for the loss of synchronization.

The basic receiver structure is the same as in Figure 30. As shown previously, upsampling by P (at the symbol rate) in the transmitter extended the bit through the whole length of the code. Therefore, downsampling and forward delaying at the receiver results in the matrix $\underline{y}[n]$, which comprises the P versions of the original data sequence $a[n]$, corrupted by the channel. The current case is diversified from the previous because the loss of synchronization in the PN sequence causes the receiver to despread the received sequence incorrectly as shown in the following Figure 32:

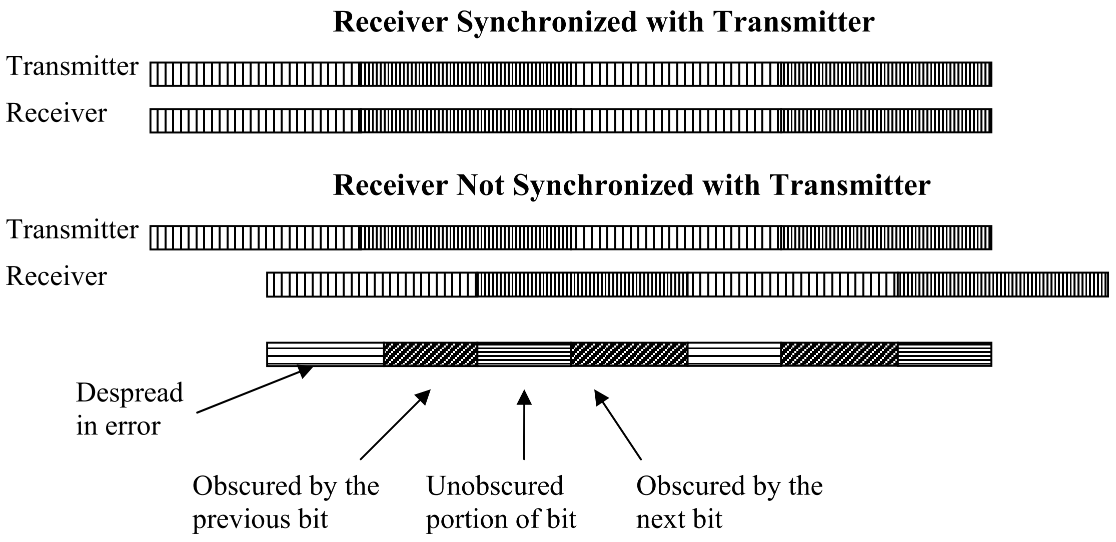

Receiver Synchronized with Transmitter
Transmitter
Receiver
Receiver Not Synchronized with Transmitter
Transmitter
Receiver
Despread in error
Obscured by the previous bit
Unobscured portion of bit
Obscured by the next bit

Figure 32. Loss of Receiver Synchronization

Nevertheless, it is shown in the following analysis that the loss of synchronization is an anomaly that appears in the eigenvalues of the autocorrelation matrix R_{yy} and spans a limited subspace. Let us assume that each portion of $\underline{y}[n]$ contains two symbols at maximum, namely the current and a portion of the previous. This is equivalent in writing to:

$$\underline{y}[n] = \underline{g}_0 a[n] + \underline{g}_1 a[n-1] \tag{4.32}$$

where $\underline{g}_0$ and $\underline{g}_1$ depend on both the channel and the autocorrelation of the code. In matrix form, (4.32) can be expressed as:

$$\underline{y}[n] = \left|\underline{g}_0, \underline{g}_1\right| \cdot \begin{vmatrix} a[n] \\ a[n-1] \end{vmatrix}. \tag{4.33}$$

From Equation (4.33) we can conclude that $\underline{y}[n]$ spans a two-dimensional subspace defined by the vectors $\underline{g}_0$ and $\underline{g}_1$. In order to estimate the new subspace, we will examine the eigenvalues and eigenvectors of the autocorrelation matrix R_{yy}. Figure 33 below illustrates an example of what the eigenvalues of R_{yy} look like when there is no synchronization:

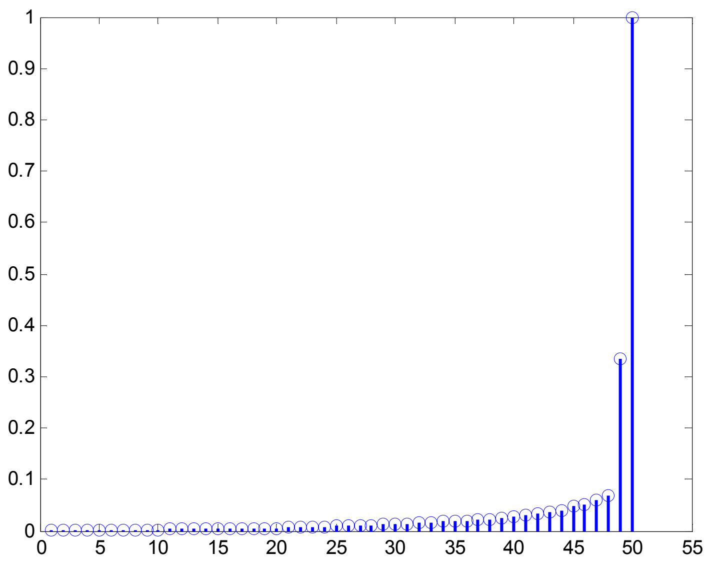


Figure 33. Plot of the Normalized Eigenvalues When No Syncronization Is Achieved

The example uses a particular case with upsampling $P = 50$ and the MATLAB sorting of the eigenvalues in which the largest eigenvalue is placed last. We observe that only two eigenvalues are significant whereas the others are very close to zero and can be neglected.

In the absence of noise, this means that $\underline{y}[n]$ can be analyzed by these two nonzero eigenvalues and corresponding eigenvectors $\underline{v}_1$ and $\underline{v}_2$ only:

$$\underline{y}[n] = \underline{v}_1 e_1[n] + \underline{v}_2 e_1[n] . \tag{4.34}$$

Since the eigenvectors are orthonormal:

$$e_1[n] = \underline{v}_1^T \underline{y}[n] \text{ and } e_2[n] = \underline{v}_2^T \underline{y}[n] \Rightarrow \underline{e}[n] = \begin{vmatrix} \underline{v}_1^T \\ \underline{v}_2^T \end{vmatrix} \cdot \underline{y}[n] . \tag{4.35}$$

Substituting (4.29) into (4.31) yields:

$$\underline{e}[n] = \begin{vmatrix} \underline{v}_1^T \\ \underline{v}_2^T \end{vmatrix} \cdot \begin{vmatrix} \underline{g}_0, \underline{g}_1 \end{vmatrix} \cdot \begin{vmatrix} a[n] \\ a[n-1] \end{vmatrix} . \tag{4.36}$$

Assuming $\underline{g}_0$ and $\underline{g}_1$ are linearly independent and if we let $\begin{vmatrix} \underline{m}_1^T \\ \underline{m}_2^T \end{vmatrix} = \left(\begin{vmatrix} \underline{v}_1^T \\ \underline{v}_2^T \end{vmatrix} \cdot \begin{vmatrix} \underline{g}_0, \underline{g}_1 \end{vmatrix} \right)^{-1}$ we can write:

$$\begin{vmatrix} a[n] \\ a[n-1] \end{vmatrix} = \begin{vmatrix} \underline{m}_1^T \\ \underline{m}_2^T \end{vmatrix} \cdot \underline{e}[n] . \tag{4.37}$$

Since $\underline{g}_0$ and $\underline{g}_1$ are linearly independent, it follows that $\underline{m}_1^T$ and $\underline{m}_2^T$ are also linearly independent. Consequently, since $a[n] = \underline{m}_1^T \cdot \underline{e}[n]$ it is true that $a[n-1] = \underline{m}_2^T \cdot \underline{e}[n] \Leftrightarrow a[n] = \underline{m}_2^T \cdot \underline{e}[n+1]$. In vector form the system of equations can be expressed as:

$$\begin{vmatrix} \underline{m}_1^T, -\underline{m}_2^T \end{vmatrix} \cdot \begin{vmatrix} \underline{e}[n] \\ \underline{e}[n+1] \end{vmatrix} = 0 . \tag{4.38}$$

In order to calculate $\underline{m} = \begin{vmatrix} \underline{m}_1 \\ -\underline{m}_2 \end{vmatrix}$, we start by calculating the covariance matrix $K_e = \sum_n \begin{vmatrix} \underline{e}[n] \\ \underline{e}[n+1] \end{vmatrix} \cdot \left| \underline{e}^T[n], \underline{e}^T[n+1] \right|$, then estimate the eigenvalues and eigenvectors and select the eigenvector corresponding to the minimum eigenvalue to be $\underline{m}$. The estimated sequence $\hat{a}[n]$ is derived from (4.37), once the estimates for $\underline{m}$ are known.

D. SUMMARY

In the current chapter, we first examined the theory relevant to DSSS and multi-rate signal processing in order to better appreciate the previously proposed DSSS communication scheme and the proposed receiver changes. Subsequently, the problems inherent in the previously proposed DSSS communication scheme were shortly reviewed. Although the transmitter scheme was proved to be sufficient in the original Seaweb communication scheme, the adaptive implementation of the RAKE receiver was problematic. Finally, the current chapter proposes a RAKE receiver replacement. Two blind equalization algorithms were developed in order to circumvent the adaptive tap estimation, based on whether the synchronization between the receiver and the transmitter was achieved. Both algorithms are based on the fact that the channel interference will always span a small subspace in the eigenvalues of the autocorrelation matrix of the received signal. If the subspace is very large, that means that the time coherence of the channel is too small, which leaves no margin for coherent communications of any kind. Both algorithms were analyzed in the discrete time domain for the various combinations of multipath, white and colored noise. The simulation of both algorithms in the various combinations of noise and multipath was done in MATLAB and the results are presented in the next chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

V. IMPLEMENTATION AND PERFORMANCE

This chapter discusses the MATLAB implementation of the proposed Seaweb receiver design and the performance results in simulated ocean channels. The chapter starts with the necessary reference to E_b/N_0 and Bit-Error Rate (BER) and then presents the performance of the receiver in terms of BER vs. E_b/N_0 for the various combinations of multipath and noise conditions. The block diagrams of the transmitter and the receiver structures were presented in Figures 24 through 31 of Chapter IV for the various combinations of multipath and noise. Each case is discussed here separately with particular emphasis on how varying specific design parameters (e.g., bit rate, packet size etc.) affects the (BER) of the receiver. Under the current configuration Seaweb operates with 72-bit utility packets and an available bandwidth of 5 kHz. The utility packet information bit-rate when transmitted with DSSS is proposed to be 40 to 100 bps. Finally, the performance of the receiver is measured with a simulated doubly spread underwater acoustic model and additive colored noise case, which is a representative case of the underwater channel.

A. PERFORMANCE ANALYSIS OF COMMUNICATION SYSTEMS

1. Signal Power (*S*)

The power of the transmitted signal *S* is evaluated using the expression:

$$S = \frac{\sum_{n=0}^{N-1} |x[n]|^2 \, T_S}{NT_S} \tag{5.1}$$

where $x[n]$ are the amplitudes of the DSSS signal samples and *N* is the total number of chips. If we normalize so that $|x[n]| = 1$, (5.1) can be rewritten as:

$$S = \frac{\sum_{n=0}^{N-1} |1|^2}{N} = \frac{N}{N} = 1\,. \tag{5.2}$$

2. Signal-to-Noise Ratio (*SNR*)

The *SNR* of the received signal was defined in Equation (2.2) as the ratio of the time average signal power divided by the time average noise power. Practically the SNR is the signal power excess over the channel noise and is a natural *figure of merit* used to describe the performance of communications. In digital communications, it is more common to use a scaled version of the *SNR* called E_b/N_0 where E_b is the energy-per-bit and N_0 is the noise power spectral density. The analytical expression for the relation between E_b/N_0 and *SNR* is the following:

$$\frac{E_b}{N_0} = SNR \cdot \frac{W}{R_b} \tag{5.3}$$

where W is the available bandwidth of 5 kHz and R_b the bit rate. It is mentioned in [1] that when the DSSS signal $SNR < -6$ dB the signal cannot be audibly distinguished from the background noise, however it is loud enough to be detectable by the receiver. For a 40-bps signal the corresponding E_b/N_0 threshold is approximately 22 dB. In order to enable a direct comparison with [1], the 22 dB threshold was maintained in this thesis. The range of the desired E_b/N_0 used for the simulation was from 0 through 25 dB.

3. Bit-Error Rate (BER)

BER is the metric that quantifies the reliability of a communication system as the ratio of the erratic bits to the total bits sent:

$$BER = \frac{\text{Errors}}{\text{Total Number of Bits}}. \tag{5.4}$$

It depends on the ratio E_b/N_0 and their interrelation is important when a small E_b/N_0 must be maintained while the BER must be kept to a minimum (below 10^{-3} for Seaweb). Therefore, the performance throughout the simulation was expressed in terms of BER vs. E_b/N_0 plots.

B. PERFORMANCE RESULTS

The underwater acoustic channel was modelled as described in Chapter III in order to incorporate the small-scale effects. However, the performance of the receiver was measured for all of the cases examined in Chapter IV before being measured for the doubly spread case. The simulation involved sending a thousand packets for each of the E_b/N_0 values in order to create sufficient statistics. The BER plots therefore represent Monte Carlo realizations of the performance in the various E_b/N_0 conditions.

1. AWGN Only

The case of the AWGN channel was chosen in order to test the equalization algorithm and obtain an estimate of the receiver performance in the absence of multipath. In the case of AWGN channel, white noise is added to the transmitted signal and the received signal is of the form $y[n] = c[n] * a[n] + w[n]$. The receiver configuration used is that described by Equations (4.15) through (4.19) and represented in Figure 26.

Figure 34 below demonstrates the receiver performance in AWGN when the packet size is varied from the required 72 to 1242 bits while the bit-rate is fixed at 40 bps. We observe that increasing the packet size does not degrade the receiver performance.

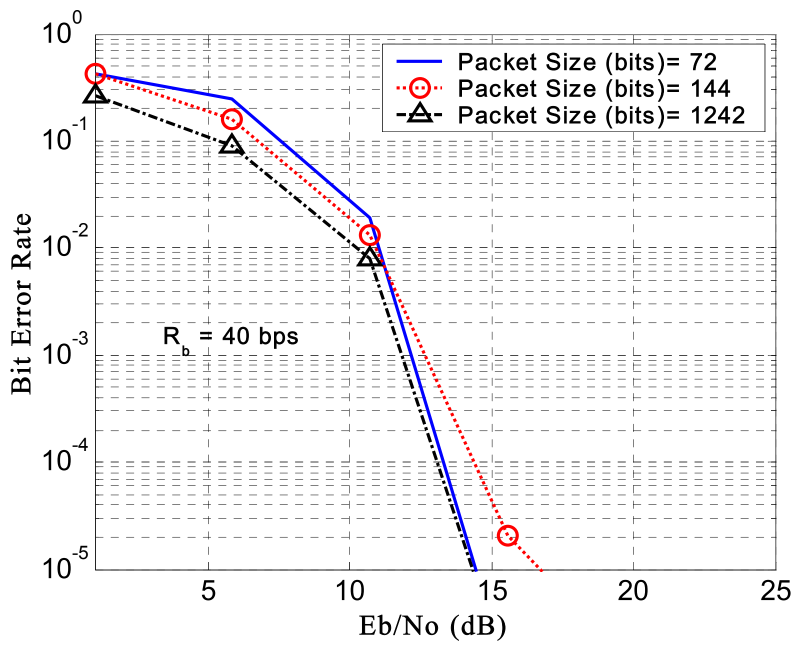


Figure 34. Performance in AWGN Only When Varying the Packet Size and Bit-Rate is Fixed at 40 bps

2. Static Channel Only

In this case, the multipath effect is isolated and the receiver performance is evaluated for stability over 100 trials. The unfiltered static impulse response, as shown in Figure 12, was used for this case. However, since there is no noise available, there can be no BER vs. E_b/N_0 evaluation. The error rate over the number of trials was used as a performance metric in this case. Since the impulse response is static, the received signal is of the form $y[n] = g[n] * (a[n] * c[n])$. The receiver configuration evaluated is that of Figure 30, and the receiver successfully equalized the multipath effect in all of the trials.

3. Static Channel with AWGN

This is the natural combination of the two previous cases in which the signal is subjected to the combined effect of multipath and AWGN. The received signal has the form $y[n] = g[n] * a[n] + w[n]$ and the BER vs. E_b/N_0 is evaluated when different design parameters are varied. The receiver structure evaluated is that of Figure 30 described by Equations (4.20) through (4.25), and the receiver performance is demonstrated in Figures 35 through 40 on the following pages.

Figure 35 demonstrates the receiver performance when the packet size is varied from the required 72 to 1242 bits while the bit-rate is fixed at 40 bps. As expected from the previous cases examined so far, increasing the packet size does not degrade the receiver performance.

Figures 36 and 37 demonstrate the receiver performance for 72 and 144-bit packets when the bit rate is varied in the range 40 to 200 bps. It is shown that the highest achievable bit-rate for the settings used is 100 bps.

Finally, Figures 38 through 40 show the effect of the channel length on packets of sizes 72 through 1242 bits at a fixed bit-rate of 40 bps. This case is interesting because up to now we eluded the fact that the algorithm is supposed to work up to a channel length of *P*. This length can be considered as the coherence time of the channel. However, it is demonstrated in the aforementioned figures that the algorithm proved robust for much longer impulse response lengths. The channel lengths are stated in samples.

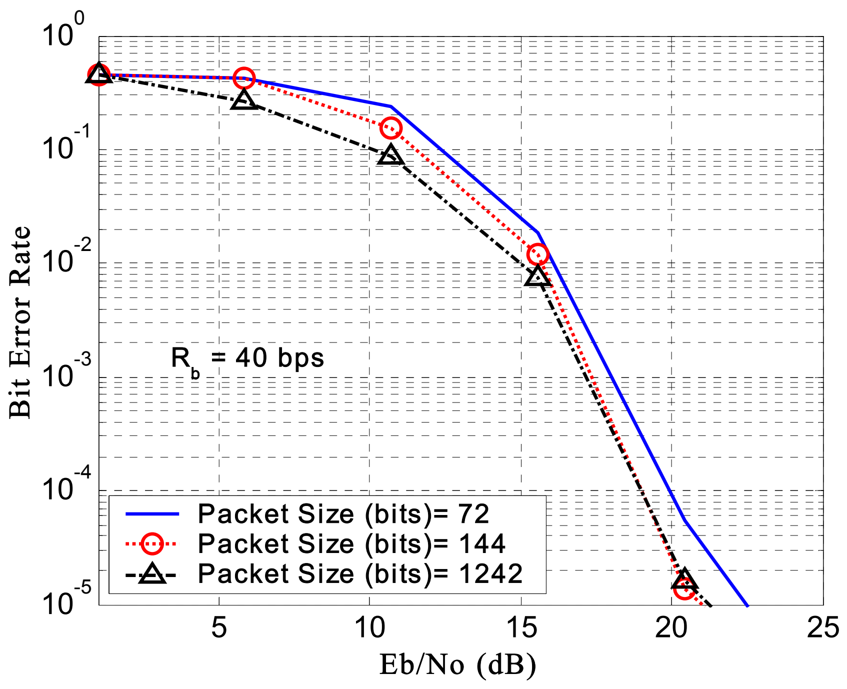


Figure 35. Performance in Multipath with AWGN When Varying the Packet Size and Bit-Rate is Fixed at 40 bps

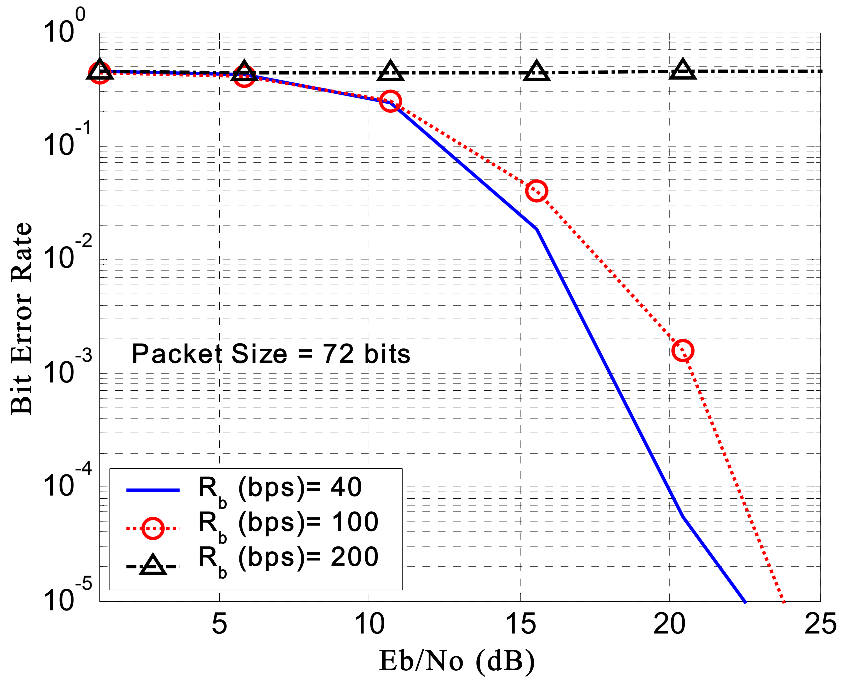


Figure 36. Performance in Multipath with AWGN When Varying the Bit-Rate and Packet Size is Fixed at 72 bits

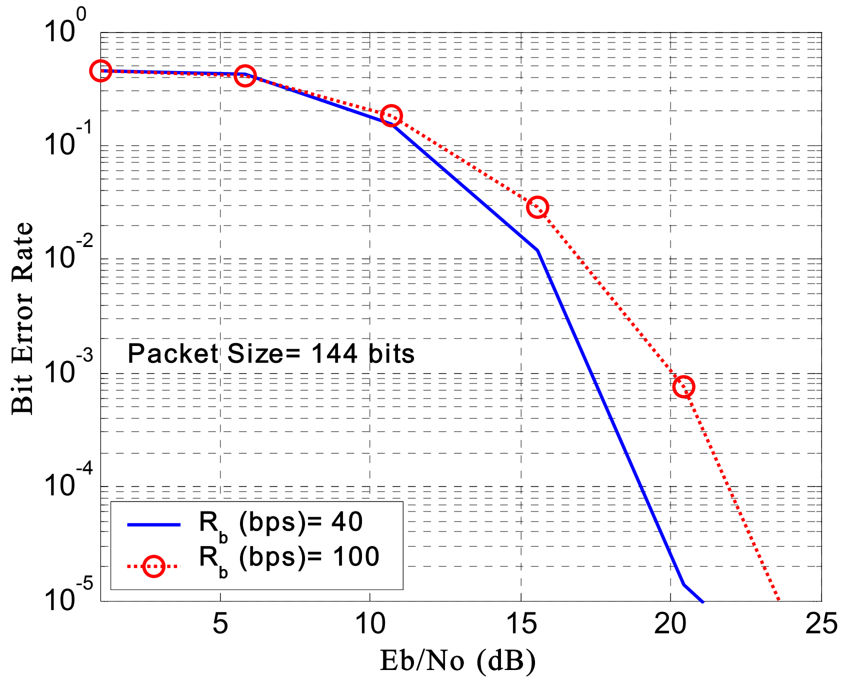


Figure 37. Performance in Multipath with AWGN When Varying the Bit-Rate and Packet Size is Fixed at 144 bits

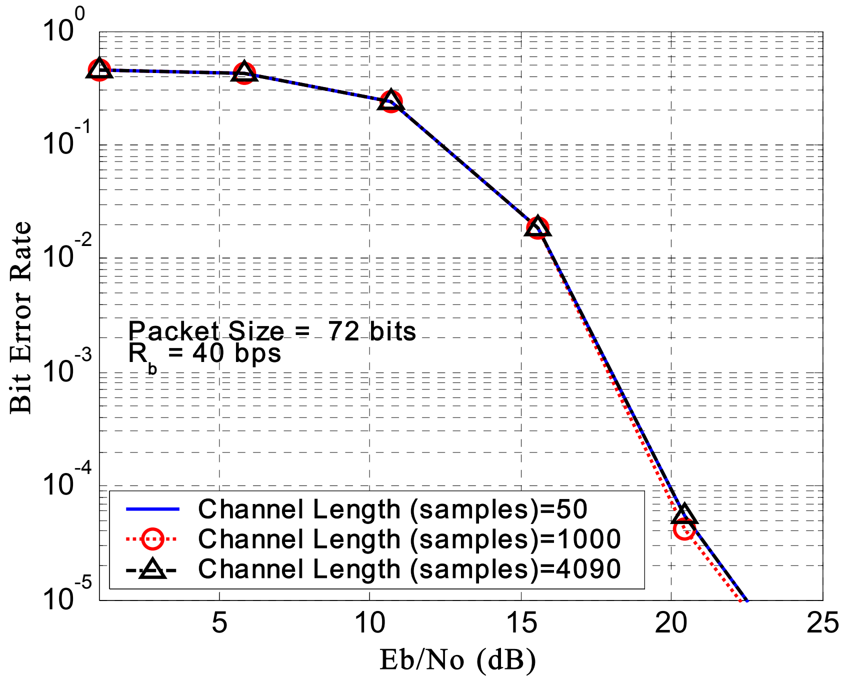


Figure 38. Performance in Multipath with AWGN When Varying the Channel Length (in samples) for Bit-Rate 40 bps and Packet Size 72 bits

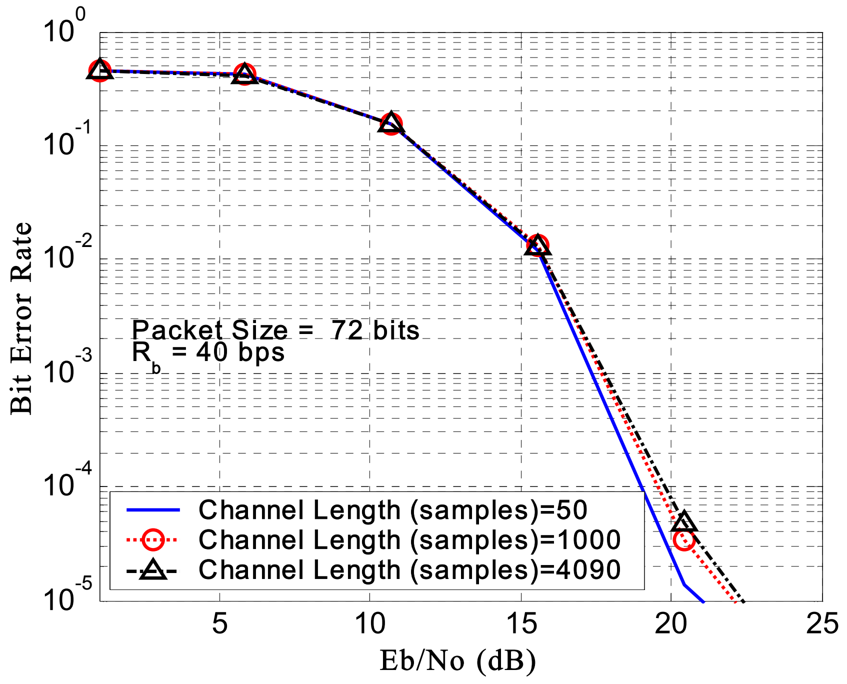


Figure 39. Performance in Multipath with AWGN When Varying the Channel Length for Bit-Rate 40 bps and Packet Size 144 bits

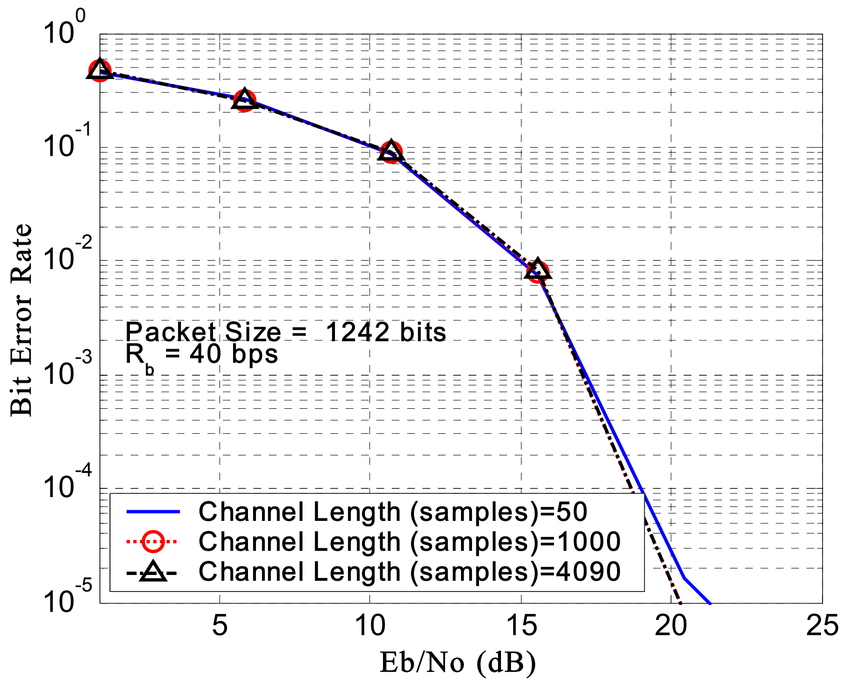


Figure 40. Performance in Multipath with AWGN When Varying the Channel Length for Bit-Rate 40 bps and Packet Size 1242 bits

4. Colored Noise Only

The colored noise was produced by filtering white noise by a 7th order FIR filter as follows:

$$H(z) = \frac{1}{z^6 - 0.9z^5 + 0.4z^4 - 0.2z^3 + 0.1z^2 - 0.1z}. \tag{5.5}$$

This manipulation changes the form of the covariance matrix R_{ww} of white noise from $\sigma^2{}_w I$, to non-diagonal by introducing off-diagonal terms so that the receiver could be tested for colored noise. In Figure 41 below white noise is shown to equally affect the frequency spectrum whereas colored noise is frequency-dependent.

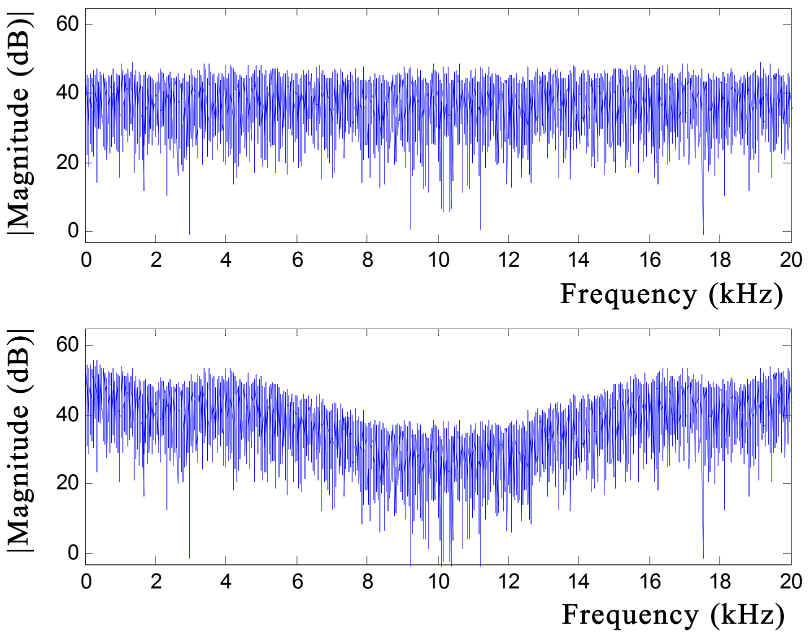


Figure 41. Comparison of White and Colored Noise Effect in the Frequency Domain

Figure 42 on the next page demonstrates the receiver performance when the packet size is varied from the required 72 to 1242 bits while the bit-rate is fixed at 40 bps. As expected from the previous cases examined so far, increasing the packet size does not degrade the receiver performance.

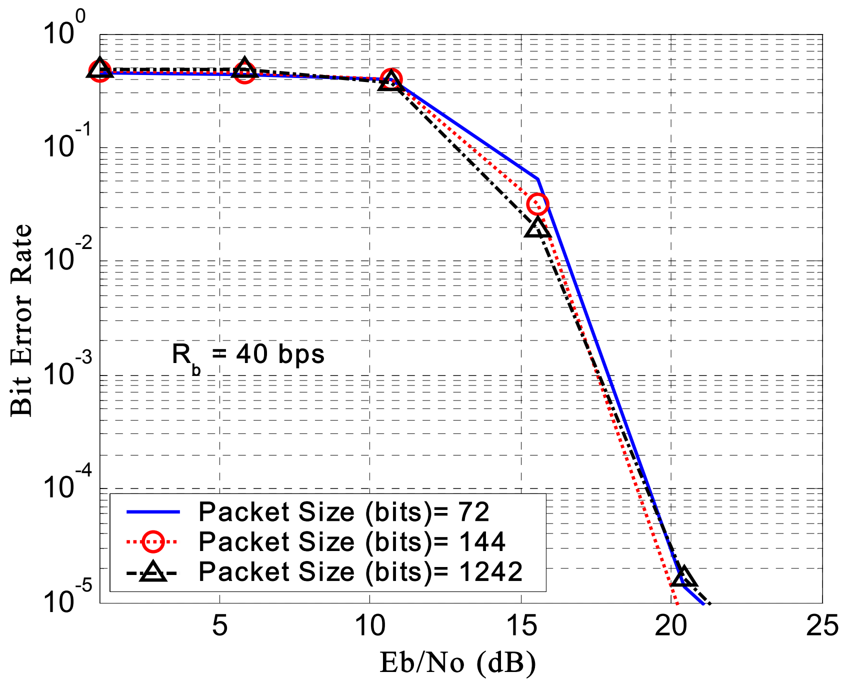


Figure 42. Performance in Colored Noise Only When Varying the Packet Size and Bit-Rate is Fixed at 40 bps

5. Static Channel with Additive Colored Noise

This case combines the effect of multipath and colored noise. The received signal has the form $y[n] = g[n]*(a[n]*c[n]) + n[n]$ and the BER vs. E_b/N_0 is evaluated when different design parameters are varied. The receiver structure evaluated is that of Figure 30, adapted for colored noise and the receiver performance is shown in Figures 43 through 48 on the following pages .

Figure 43 demonstrates the receiver performance when the packet size is varied from the required 72 to 1242 bits while the bit-rate is fixed at 40 bps. The simulation results are consistent with the previous cases examined so far and increasing the packet size does not degrade the receiver performance.

Figures 44 and 45 demonstrate the receiver performance for 72 and 144-bit packets when the bit rate is varied in the range 40 to 200 bps. It is shown that the highest achievable bit-rate for the settings used is 100 bps.

Finally, Figures 46 through 48 show the effect of the channel length on packets of sizes 72 through 1242 bits at a fixed bit-rate of 40 bps. As previously, the algorithm proved robust for impulse response lengths longer than *P*. The channel lengths are stated in samples.

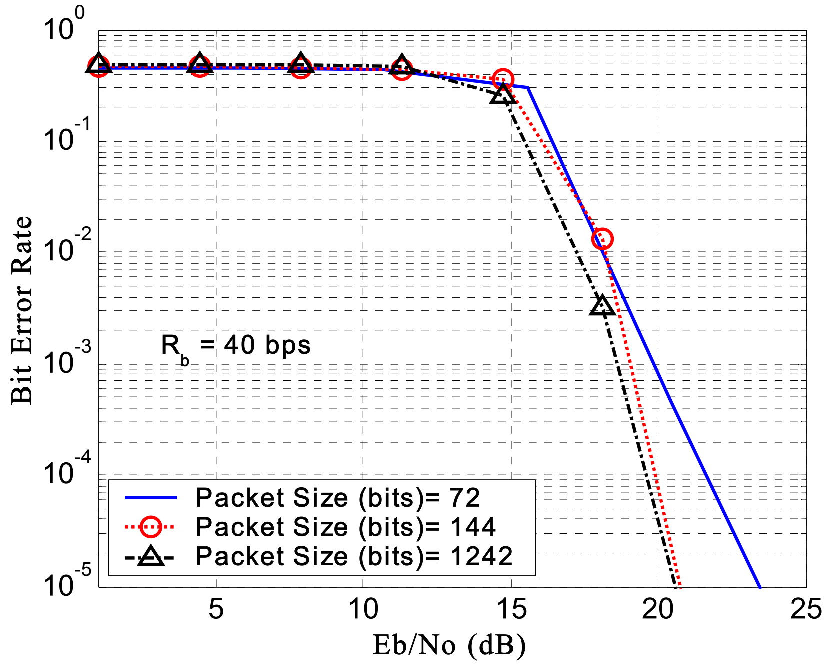


Figure 43. Performance in Multipath with Additive Colored Noise When Varying the Packet Size and Bit-Rate is Fixed at 40 bps

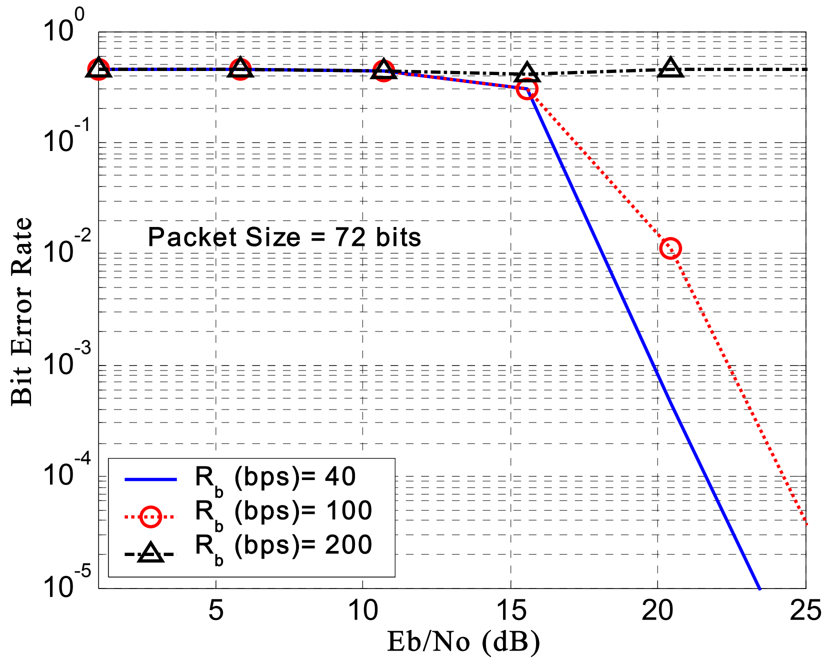


Figure 44. Performance in Multipath with Additive Colored Noise When Varying the Bit-Rate and Packet Size is Fixed at 72 bits

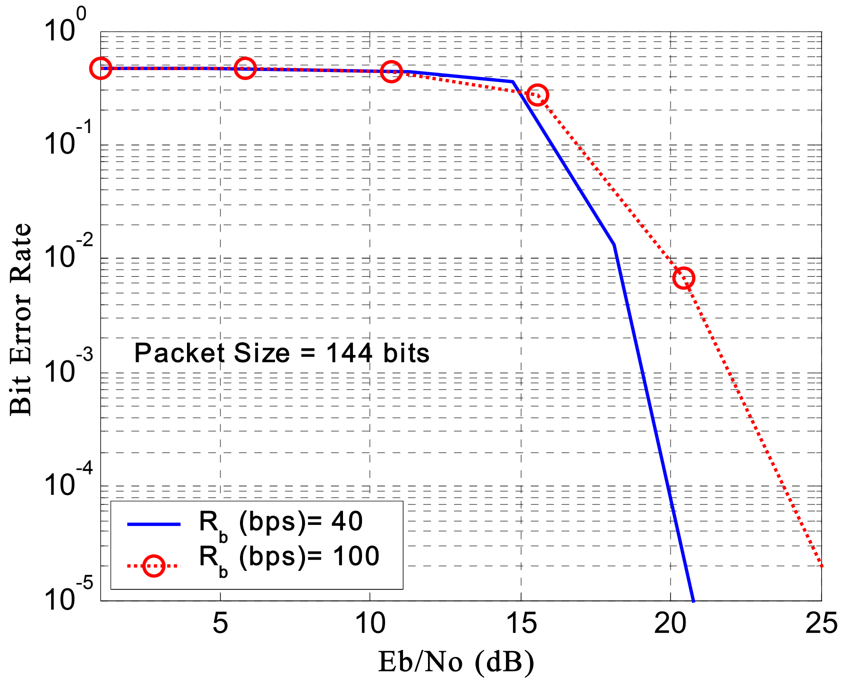


Figure 45. Performance in Multipath with Additive Colored Noise When Varying the Bit-Rate and Packet Size is Fixed at 144 bits

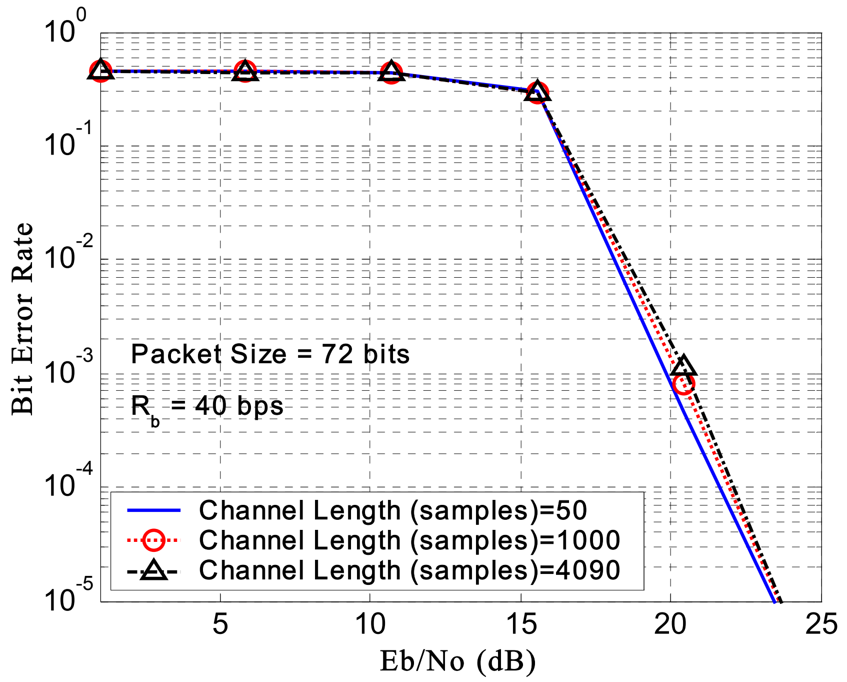


Figure 46. Performance in Multipath with Additive Colored Noise When Varying the Channel Length (in samples) for Bit-Rate 40 bps and Packet Size 72 bits

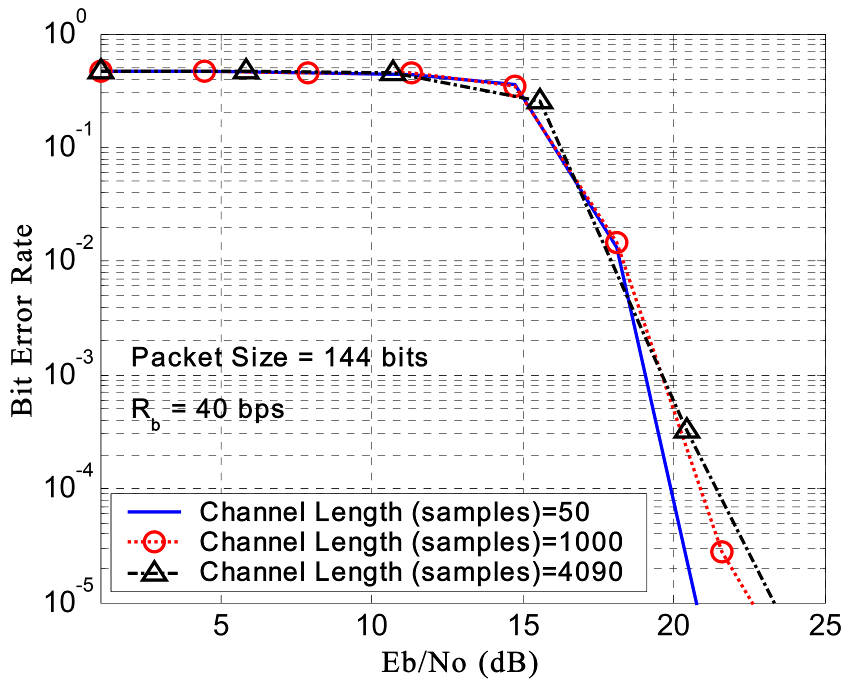


Figure 47. Performance in Multipath with Additive Colored Noise When Varying the Channel Length (in samples) for Bit-Rate 40 bps and Packet Size 144 bits

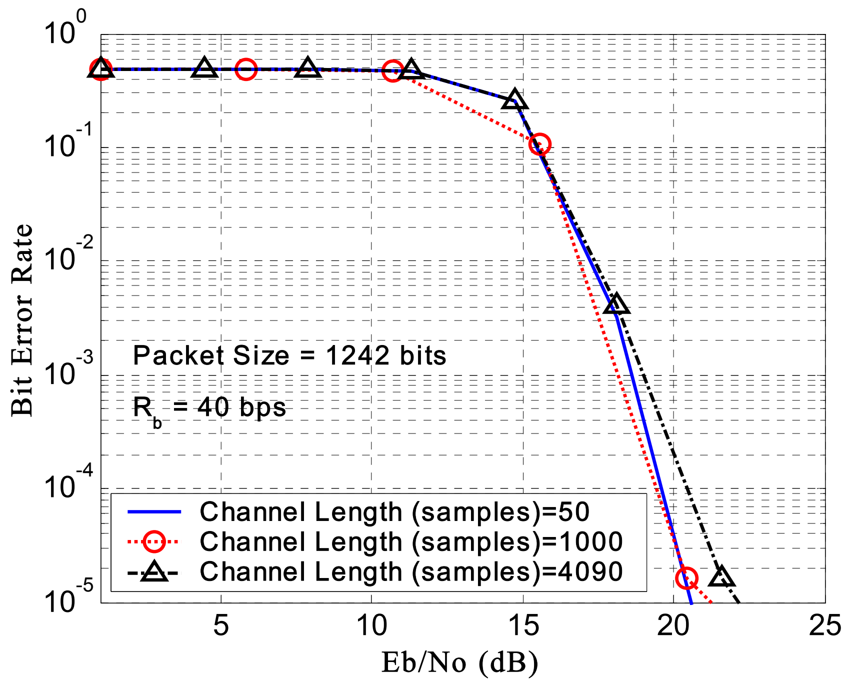


Figure 48. Performance in Multipath with Additive Colored Noise When Varying the Channel Length (in samples) for Bit-Rate 40 bps and Packet Size 1242 bits

6. Doubly Spread Channel Only

In this case the effect of multipath in a doubly spread channel is isolated from noise and the performance of the receiver configuration in Figure 30 is evaluated for stability. The impulse response shown for $\sigma^2 = 8$ in Figure 14 was used in this case. Since there is no noise available, there can be no BER vs. E_b/N_0 evaluation. The percentage error rate over the channel length was used as a performance metric in this case, given the fact that the equalizer is designed to equalize an interference of length $M < P$. Since the impulse response is time-variable, the channel is considered static only within the coherence time of the channel. The received signal can only be obtained by convolving the transmitted signal with the channel, if the signal duration is comparable to the coherence time of the channel. The performance of the receiver in the doubly spread case is shown in Figure 49 below for the specific case where $P = 50$. We observe that, although the algorithm outperformed equalization up to a channel length of $3 \cdot P = 150$, the error rate increases abruptly to 50% for longer channel lengths.

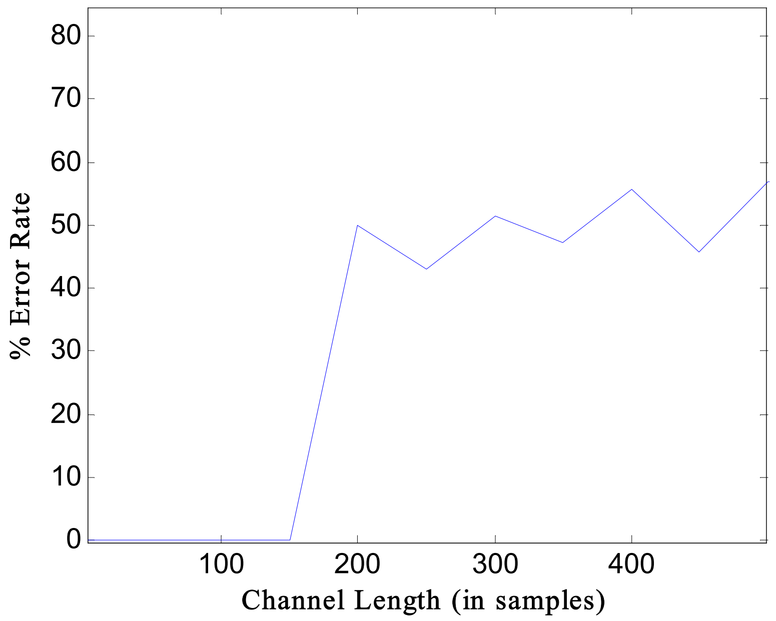


Figure 49. Error Rate vs.Channel Length (in samples) for Doubly Spread Channel

7. Doubly Spread Channel with Additive Colored Noise

This case was chosen to test the performance of the receiver shown in Figure 30 for the adverse case of a doubly spread channel with additive colored noise. Since the channel impulse response is time-variable, the received sequence can only by derived from the transmitted sequence by convolution with the channel impulse response if the signal duration is comparable to the coherence time of the channel. The colored noise is additive to enable the BER vs. E_b/N_0 evaluation. The receiver performance is demonstrated in Figures 50 through 55 on the following pages.

Figure 50 demonstrates the receiver performance when the packet size is varied from the required 72 to 1242 bits while the bit-rate is fixed at 40 bps. The simulation results are compatible with the previous cases and increasing the packet size does not degrade the receiver performance; however, more power is needed to achieve the desired BER.

Figures 51 and 52 demonstrate the receiver performance for 72 and 144-bit packets when the bit rate is varied in the range 40 to 100 bps. It is shown that the highest achievable bit-rate for the settings used is 40 bps.

Finally, Figures 53 through 55 show the effect of the channel length on packets of sizes 72 through 1242 bits at a fixed bit-rate of 40 bps. It was shown previously that in the absence of noise the maximum channel length that can be equalized using this method is $3 \cdot P = 150$ (samples). Therefore the channel lengths simulated here were in the range 50 to 150 (in samples). Consistent with the previous findings, the algorithm proved robust for impulse response lengths longer than *P*.

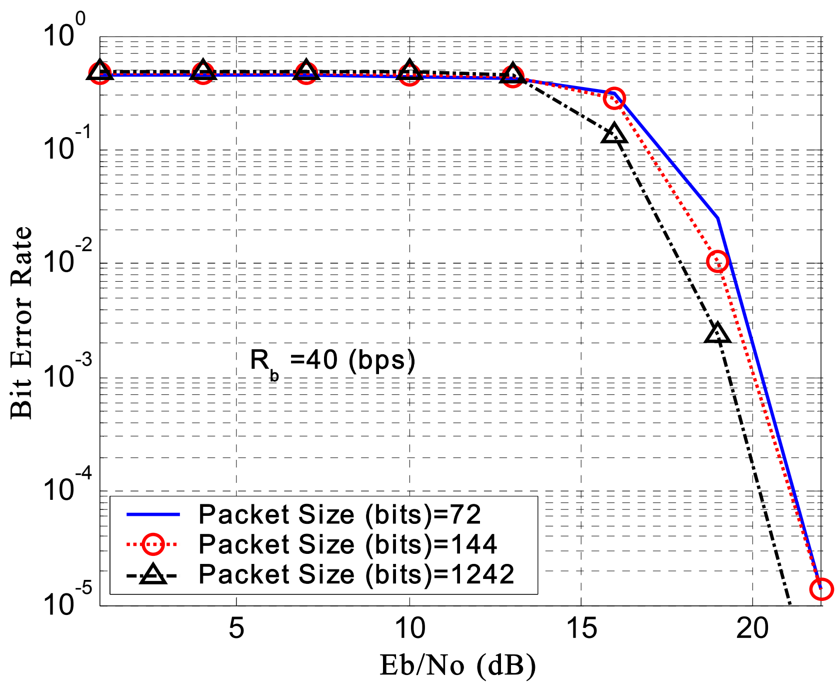


Figure 50. Performance in Doubly Spread Channel with Additive Colored Noise When Varying the Packet Size and Bit-Rate is Fixed at 40 bps

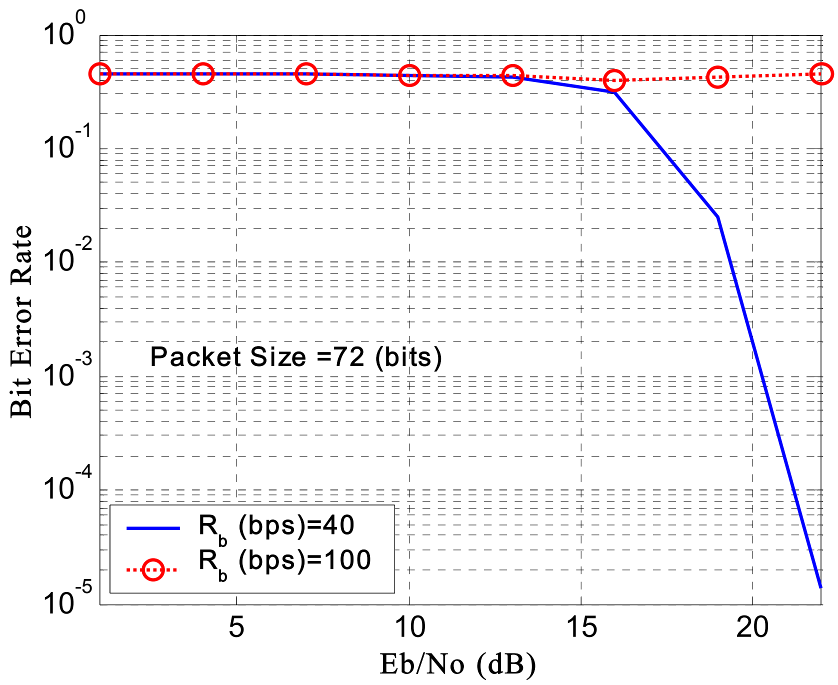


Figure 51. Performance in Doubly Spread Channel with Additive Colored Noise When Varying the Bit-Rate and Packet Size is Fixed at 72 bits

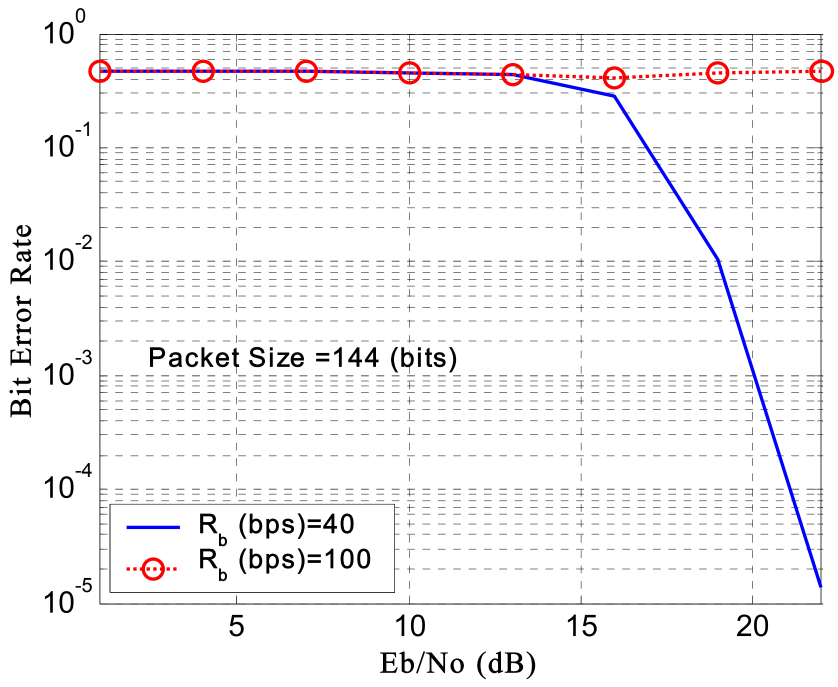


Figure 52. Performance in Doubly Spread Channel with Additive Colored Noise When Varying the Bit-Rate and Packet Size is Fixed at 144 bits

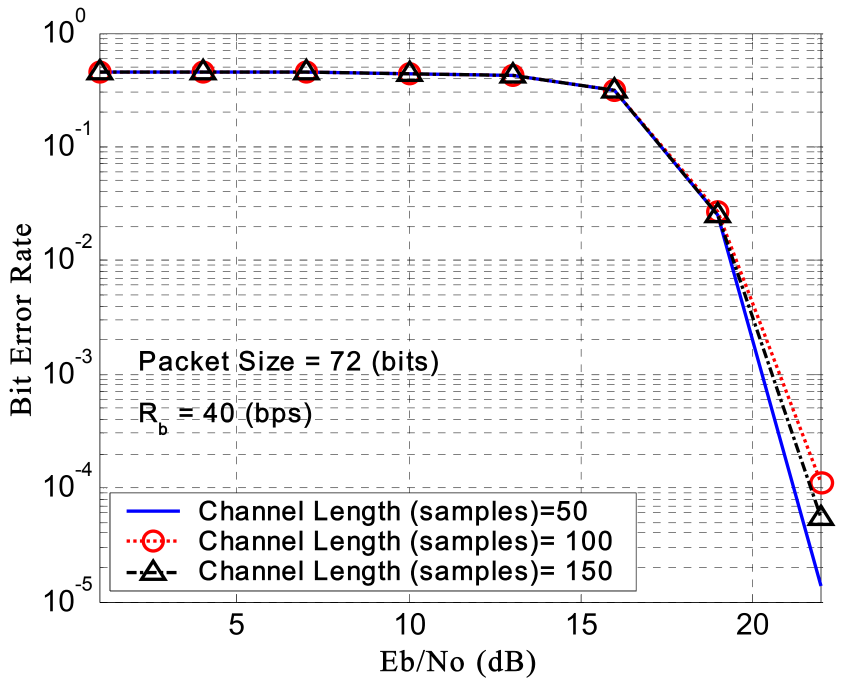


Figure 53. Performance in Doubly Spread Channel with Additive Colored Noise When Varying the Channel Length (in samples) for Bit-Rate 40 bps and Packet Size 72 bits

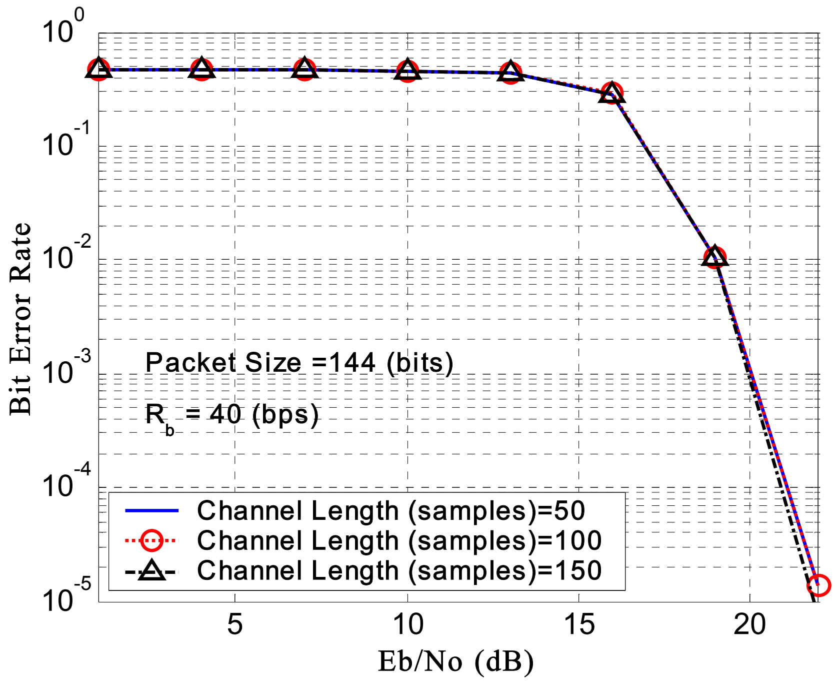


Figure 54. Performance in Doubly Spread Channel with Additive Colored Noise When Varying the Channel Length (in samples) for Bit-Rate 40 bps and Packet Size 144 bits

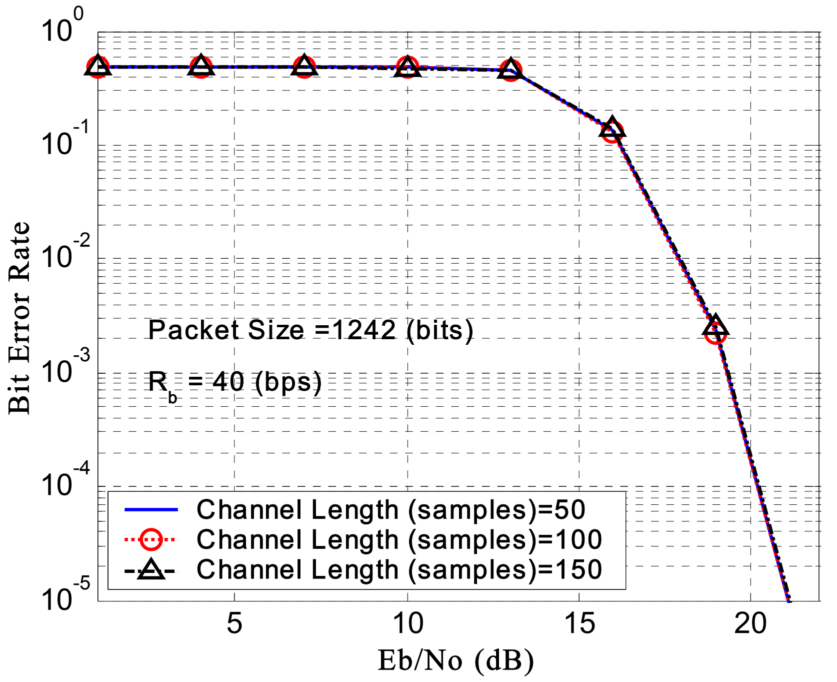


Figure 55. Performance in Doubly Spread Channel with Additive Colored Noise When Varying the Channel Length (in samples) for Bit-Rate 40 bps and Packet Size 1242 bits

8. Loss of Receiver Synchronization

In Seaweb, the receiver synchronization is achieved by appending three short pulses for acquisition and an FM chirp in the beginning of each data packet. When the PN sequence in the transmitter and the receiver are synchronized, the received DSSS sequence is despread by simple multiplication with the PN sequence in the receiver. However, due to the volatility of the underwater medium it is not unlikely that the acquisition frame may be lost, and the whole packet has to be retransmitted because the packet cannot be despread. It was shown earlier in Chapter IV that in the case where synchronization is lost, the resulting interference will appear in the eigenvalues of the covariance matrix of the received signal. Therefore, the recovery of the packet without retransmission is possible by adapting the receiver structure of Figure 30 according to Equations (4.32) through (4.39).

Loss of synchronization is represented in Figure 32. In order to simulate loss of synchronization, a random delay of duration up to half a bit was injected to the received signal so that each bit contains a portion of the previous and the next bit. The receiver performance was only tested for the doubly spread channel and additive colored noise case. The algorithm switches to loss-of-synchronization mode when the second largest eigenvalue of the covariance matrix of the received signal is greater than two tenths of the largest eigenvalue. The receiver performance when the packet size is varied from 72 to 1242 bits, while the bit-rate remains constant at 40 bps, is shown in Figure 56 on the next page. It is shown that the algorithm can achieve a BER of 10^{-3} or less for the specified boundary of 22 dB depending on the packet size used. The rest of the results closely match the results of the previous case when synchronization was achieved and will not be repeated.

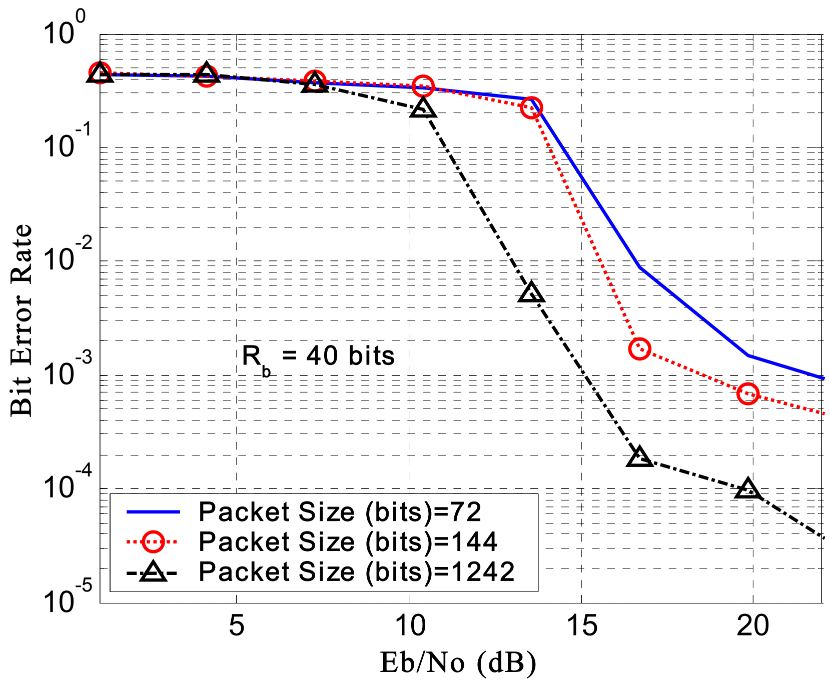


Figure 56. Performance in Doubly Spread Channel with Additive Colored Noise When Synchronization is Lost as a Function of Packet Size and Fixed Bit-Rate at 40 bps

C. SUMMARY

In the current chapter, we first reviewed the relevant theory for communication-systems performance analysis and then implemented the receiver structures that were presented in chapter IV using MATLAB. The receiver performance was presented either as a function of BER vs. E_b/N_0 or channel error rate. Each case was simulated by varying several parameters (i.e., bit-rate, packet length and channel length) independently and measuring their effect on the receiver performance. Monte Carlo simulation was used for all simulated cases and a thousand packets were sent for each E_b/N_0 value in order to gather sufficient statistics. All of the algorithms proved to work adequately even for the adverse case of the doubly spread channel with additive colored noise. In most cases, the BER reached 10^{-5} below the $E_b/N_0 = 22$ dB required for the system to possess LPD characteristics and under no conditions was there a need for retransmission due to excessive errors.

In the next and final chapter, we summarize the findings and present areas for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION

The goal of this thesis was to study a RAKE receiver replacement for use in Seaweb. The first step in the current research was to simulate a more realistic channel with time variability and time spread characteristic of undersea acoustic propagation. Therefore, a statistics-based model was implemented first in order to incorporate the small-scale effects inherent in the ocean. Subsequently, two blind equalization algorithms based on subspace-decomposition were implemented for use when receiver synchronization could be achieved or not. The performance of the receiver structures was first evaluated using a static impulse response for the various combinations of multipath and colored or white noise, and then for a simulated doubly spread channel with additive colored noise. A Monte Carlo simulation was used to measure performance in all cases, and the algorithms proved to work adequately in all of the cases simulated.

A. FINDINGS

The receiver structures simulated were shown to equalize the channel effect adequately, in most cases below the $E_b/N_0 = 22$ dB limit required for LPD communications. A BER of 10^{-5} is generally possible even if that must be done at the expense of power. In the event that the 22-dB E_b/N_0 is a stringent boundary, a BER exceeding the 10^{-3} required for Seaweb was possible and errors could be mitigated through forward-error correction or automatic repeat request.

The use of a 144-bit packet combined with a higher bit-rate of 100 bps was proven to be possible, except when the channel is very severe. As a comparison with the existing scheme, the packet size and bit-rate are twice or more than that required. In an extremely adverse environment, like the one simulated in the doubly spread channel with additive colored noise case, the higher bit rate of 100 bps was not possible. In any case, communications were achievable for the larger packet of 144 bits even if the bit-rate had to slow from 100 bps to 40 bps.

The modelled doubly spread channel allowed the testing of the algorithm in an environment in which small-scale phenomena like the coherence time of the channel are important. Given that the packet duration used was sufficiently small compared to the channel (except for the 1242-bit packet), we were able to test the algorithm in a time-varying and time-spread environment, which resembles the real ocean. The algorithm was successful up to a length three times longer than the specifications of $M < P$ in all combinations of bit-rate and packet size.

The loss of receiver synchronization algorithm was tested for a short PN sequence and for a shift up to half a bit and was proven to work with the settings shown. A BER of 10^{-3} or less for $E_b/N_0 < 22$ dB was proven possible in a simulated doubly spread channel with additive colored noise. This is a great advantage because the appropriate use of the subspace decomposition (possibly in more than the two subspaces as used in this thesis), may prove to be an attractive alternative to the existing receiver synchronization method in Seaweb.

B. FUTURE WORK

There are four major areas that can be identified for future work. First, the modeled channel would be more accurate if the statistics used were gathered from the real ocean. In this thesis, the variability and spreading imposed upon the impulse response were arbitrary. Therefore, the underwater channel may be overestimated or underestimated compared to the real ocean.

Secondly, the algorithms presented here may be implemented in a real communication system. The transmitter and receiver groups presented here were simplified forms so that the analysis could emphasize the signal processing aspect of the scheme. The transmitted signals were considered impulses and the analysis was confined to the discrete time domain. A real communication system must account for the effects in the frequency domain, as well as in the time domain.

Thirdly, the loss-of-receiver-synchronization algorithms should be adjusted for more eigenvalues in order to accommodate larger shifts. The algorithm presented here was proved to equalize adequately when two eigenvalues were used in the simulated doubly spread channel with additive colored noise and a shift of half a bit. More eigenvalues should be taken into account if the technique is to replace the existing Seaweb synchronization method in a volatile underwater medium.

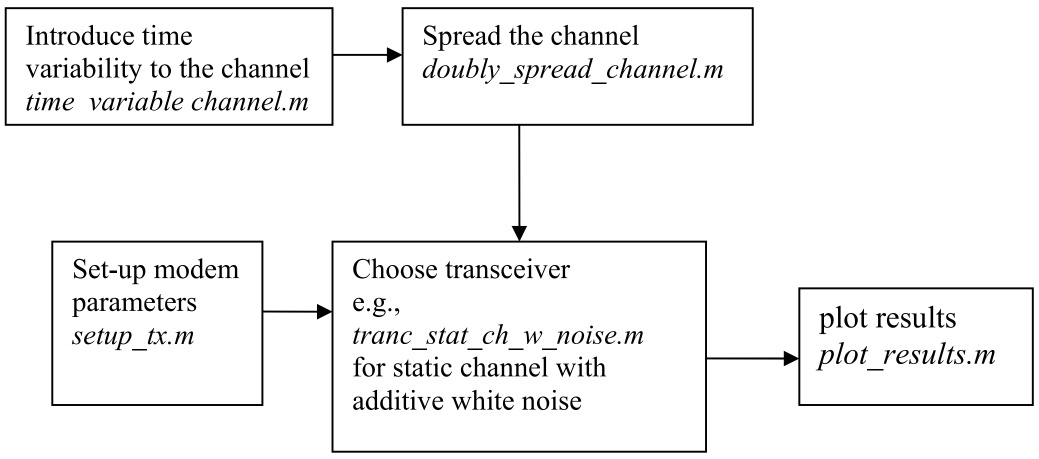
Finally, the performance of the proposed receiver structure should be tested at sea. Although the system presented here proved to work adequately in a simulated undersea environment for various multipath assumptions, noise assumptions, and synchronization assumptions, the sea trial is always the ultimate judge of success.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. SOFTWARE USERS MANUAL

This Appendix serves as a software manual for the doubly spread channel and DSSS transceiver structures simulated in this thesis. MATLAB 6.1 release 12 was used to implement all aspects of the channel, the transmitter, and the receiver. The MATLAB code does not emphasize computational efficiency or real system feasibility but instead focuses on the DSP implementation. The choice to use a separate function to initialize the modem's parameters is done to isolate the various parameters from the actual code. It is assumed here that the reader has a working knowledge of MATLAB.

The basic flow of the program will be as illustrated in the Figure 57 below:



Introduce time variability to the channel *time variable channel.m*
Spread the channel *doubly_spread_channel.m*
Set-up modem parameters *setup_tx.m*
Choose transceiver e.g., *tranc_stat_ch_w_noise.m* for static channel with additive white noise
plot results *plot_results.m*

Figure 57. Program Flow Chart

The impulse response derived from Bellhop is filtered from near zero values and saved as *ht_stat.mat*. In order to inject time variability into the static impulse response the m-file *time_variable_channel.m* must run first. The time-variable channel impulse response is saved as *ht_var.mat*. Subsequently, the time-variable impulse response is injected with time spreading using the m-file *doubly_spread_channel.m*. The doubly spread impulse response is saved as *ht_gaus.mat*. Depending on the case simulated, the appropriate impulse response is loaded to the transceiver.

Once the desired channel is available, the transceiver is setup using the function *setup_tx.m*. The function has no inputs or outputs but saves the transmitting and simulation parameters as two individual mat files, which are later loaded by the transceiver. These mat files are

- *simulation_params.mat* – defines the parameters to be used in the overall simulation. These include the number of packets and the values of E_b / N_0 to be used.
- *transmit_params.mat* – defines the parameters needed for the transmitter and receiver including bit rate, chip rate, sampling frequency, packet length and upsampling ratio.

The available transceiver functions are the following:

- *tranc_w_noise_only.m* – For the simulation and performance testing of the receiver structure shown in Figure 26, for the white noise only case. No channel input is necessary.
- *tranc_stat_ch_only.m* – For the simulation and performance testing of the receiver structure illustrated in Figure 30, for multipath only.
- *tranc_stat_ch_ w_noise.m* – For the simulation and performance testing of the receiver structure represented in Figure 30, for the combined effect of multipath and additive white noise.
- *tranc_col_noise_only.m* – For the simulation and performance testing of the receiver structure represented in Figure 30, adapted for colored noise only. No channel input is necessary.
- *tranc_stat_ch_ col_noise.m* – For the simulation and performance testing of the receiver structure represented in Figure 30, for the combined effect of multipath and additive colored noise.
- *tranc_gaus_ch_only.m* – For the simulation and performance testing of the receiver structure illustrated in Figure 30, for multipath in the doubly spread channel.
- *tranc_gaus_ch_ col_noise.m* – For the simulation and performance testing of the receiver structure represented in Figure 30, for the combined effect of multipath in a doubly spread channel and additive colored noise.

- *tranc_no_sync.m* – For the simulation and performance testing of the receiver structure represented in Figure 30, adapted for the case that no synchronization has been achieved between the receiver and the transmitter. This case is only simulated for multipath in a doubly spread channel and additive colored noise.

Each of these transceiver structures saves the simulation results as a separate mat-file. This results mat-file contains a variable called *summary*. This variable summarizes the simulation errors and simulation parameters. The exact name of the mat-file represents the channel and simulation parameters in the following order: channel name, number of packets, bit rate, packet length and channel length. For example, *color_gaus_channel_1_40_72_ch50.mat* means doubly spread channel with additive colored noise, 1 packet sent, bit-rate 40 bps, packet length 72 bits and channel length 50 samples.

Finally, the function *plot_results.m* plots the results of the simulation summary as BER vs. E_b/N_0 .

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. MATLAB CODE

```
%-----------------------------------------------------------------------------
% This program introduces time variability on the Bellhop Static impulse response
% and saves the time variable h(t) as ht_var .Run this first or use saved ht_var
% to produce the time variable impulse response.
%Georgios Pelekanos
%last revision May 8 2003
%-----------------------------------------------------------------------------
clear
clc
%load the time invariable ht from Bellhop
h=load ('cir_40.mat');
tau=h.tau';
tau=tau(487:end);
r=h.r;
hmat=h.hmat;
zs=h.zs;
zr=h.zr;
tref=h.tref;
ht_stat=h.hmat(:,501)';
ht_stat=ht_stat(487:end);%neglect negative times
save ht_stat ht_stat;

%cleanup values less than 0.03
for i=1:length(ht_stat);
   if abs(ht_stat(i))<0.03;
      ht_stat(i)=0;
   end
end
%find the non-zero locations
taps = find(ht_stat~=0);

%neglect the direct path which is stable
taps1=taps(9:end);
```

```
disp('started')
for kk=1:1000
   kk
%Create random lags the length of taps
lags_aux=round(5*randn(1,length(taps1)));
%The first lag set positive
if lags_aux(1)<0
   lags_aux(1)=abs(lags_aux(1));
end

%Create time variability as rand_lags
rand_lags=taps1+lags_aux;
%Insert the variability by concatenating the direct and variable multipath
%portions of taps
var_taps=[taps(1:8) rand_lags];
var_taps=sort(var_taps);
%Check for Duplicates and re-sort
for d=1:length(var_taps)-1
   if var_taps(d)==var_taps(d+1)
      var_taps(d)=var_taps(d)-1;
   end
end
var_taps=sort(var_taps);

% %Create uniformly distributed phase
% phi=unifrnd(-pi,pi,1,length(taps));
%The new time variable impulse response

ht_var=zeros(size(ht_stat));
for i=1:length(taps)
   ht_var(var_taps(i))=ht_stat(taps(i));
end

%Normalize  and truncate to keep the original length
%ht_var=ht_var/max(max(ht_var));
ht_var(kk,:)=[ht_var(1:length(ht_stat))];
 end
```

```
ht_var=sum(ht_var);
save ht_var ht_var;
disp('end')

figure ;
subplot(2,1,1)
plot (tau,ht_var)
title('Time Variable Impulse Response ')
xlabel('Time(sec)');
ylabel('Magnitude');
axis([0 0.085 -0.9 0.3]);
subplot (2,1,2)
plot(tau,angle(ht_var))
title('Time Variable Impulse Response')
xlabel('Time(sec)');
ylabel('Phase(rad)');
axis([0 0.085 -4 4]);
```

```
%-----------------------------------------------------------------------------
% This program introduces time spreading on the Time Variable impulse response
% and saves the Gaussian variable h(t) as ht_Gaus .Run this second and use the
% saved ht_var to reproduce the time variable impulse response.
% Georgios Pelekanos
% Last revision May 8 2003
%-----------------------------------------------------------------------------

clear
clc

%_____________________________________________
%load the time invariable impulse response
%_____________________________________________

h=load('cir_40.mat');
tau=h.tau';
tau=tau(487:end);
r=h.r;
hmat=h.hmat;
zs=h.zs;
zr=h.zr;
tref=h.tref;
fs=40960;
ht_stat=h.hmat(:,501)';
ht_stat=ht_stat(487:end);

%cleanup values less than 0.05
for i=1:length(ht_stat);
    if abs(ht_stat(i))<0.03;
        ht_stat(i)=0;
    end
end

%_____________________________________________
%load the gold code for spread spectrum
%_____________________________________________

load code;
tx=code(1:50);

% tx=sign(randn(1,50));
%_____________________________________________
%load the time  variable impulse response
%_____________________________________________

load ht_var;

%Create a gaussian shaped pulse
var=[8,16];
y=[];


figure ;
for i=1:2
y=exp(-(-30:30).^2/(2*var(i)));
```

```
        %Convolve to shift it to the variable taps position
        ht_gaus=conv(ht_var,y);

        %The first 100-300 samples correspond to the direct path
        %and are set equal to a delta function at the original
        %time variable location

        ht_gaus(1,100:250)=ht_stat(1,100:250);
        ht_gaus(1,600:650)=ht_var(1,600:650);

        ht_gaus=ht_gaus(1:length(ht_var));

        subplot(2,1,i)
        plot(tau,ht_gaus)
        title(sprintf('Time Domain Representation of Doubly Spread Channel Impulse Response for var=
%.0f',var(i)))
        xlabel('Time(sec)');
        ylabel('Magnitude');
        axis([0 0.015 -0.8 0.3]);

        end

        t = tau;

        figure ;
        specgram(ht_stat,256,fs,kaiser(256,5),220)
        title('Spectrogram of a Static Impulse Response')
        xlabel('Time(sec)');
        ylabel('Frequency(Hz)');
        colorbar('vert')
        axis([0 0.06 0 15000]);

        figure ;
        for i=1:2

        y=exp(-(-30:30).^2/(2*var(i)));
        %Convolve to shift it to the variable taps position
        ht_gaus=conv(ht_var,y);
        ht_gaus(1,100:250)=ht_stat(1,100:250);
        ht_gaus(1,600:700)=ht_var(1,600:700);

        %The first 590 samples correspond to the direct path
        %and are set equal to a delta function at the original
        %time variable location

        ht_gaus=ht_gaus(1:length(ht_var));

        subplot(2,1,i)
        specgram(ht_gaus,256,fs,kaiser(256,5),220)
        title(sprintf('Spectrogram of Time Variable and Time Spread Impulse Response for var=
%.0f',var(i)))
        xlabel('Time(sec)');
        ylabel('Frequency(Hz)');
        colorbar('vert')
        axis([0 0.06 0 15000]);
```

```
hold on;
end

yx=conv(tx,ht_stat);
yx_var=conv(tx,ht_var);
yx_gaus=conv(tx,ht_gaus);


HT=fft(ht_stat);
HT_var=fft(ht_var);
HT_gaus=fft(ht_gaus);

F= ( 0: (fs/2)/4090: (fs/2)*4089/4090 );

figure ;
subplot(2,1,1)
plot(F,20*log10(fftshift(abs(HT(1:4090)))))
title('Spectral Content of a Static Impulse Response ')
xlabel('Frequency(Hz)');
ylabel('Magnitude(dB)');

subplot(2,1,2)
plot(F,20*log10(fftshift(abs(HT_gaus(1:4090)))))
title(sprintf('Spectral Content of a Doubly Spread Impulse Response for var= %.0f',var(2)))
xlabel('Frequency(Hz)');
ylabel('Magnitude(dB)');


Yx=fft(yx);
Yx=fft(yx_var);
Yx_gaus=fft(yx_gaus);

F= ( 0: (fs/2)/4090: (fs/2)*4089/4090 );

figure ;
subplot(2,1,1)
plot(F,20*log10(fftshift(abs(Yx(1:4090)))))
title('Spread Spectrum over a Static Channel ')
xlabel('Frequency(Hz)');
ylabel('Magnitude(dB)');

subplot(2,1,2)
plot(F,20*log10(fftshift(abs(Yx_gaus(1:4090)))))
title(sprintf('Spread Spectrum over a Doubly Spread Impulse Response for var= %.0f',var(2)))
xlabel('Frequency(Hz)');
ylabel('Magnitude(dB)');

stat_corr=xcorr(ht_stat);
gaus_corr=xcorr(ht_gaus);


figure ;
subplot(2,1,1)
plot (stat_corr)
axis ([1000 7000 -0.1 1])
subplot(2,1,2)
```

```
plot (gaus_corr)
axis ([1000 7000 -0.3 2.5])

save ht_gaus ht_gaus;

% [Pxx_stat,w1] = pburg(yx,12,512,40960,'onesided');
% [Pxx_gaus,w2] = pburg(yx_gaus,12,512,40960,'onesided');

[Pxx_stat,w1] = pwelch(ht_stat,[],[],40960)
[Pxx_gaus,w2] = pwelch(ht_gaus,[],[],40960)

figure ;
subplot(2,1,1)
psdplot(Pxx_stat,w1,'Hz')
subplot(2,1,2)
psdplot(Pxx_gaus,w2,'Hz')
```

```
function setup_tx
clear
clc

%*****************************************
%
% This function sets up all transmit parameters
% and saves them as separate *.mat files
% Two separate mat files are generated
%   1. simulation_params.mat
%   2. transmit_params.mat
%
% developed by Georgios Pelekanos February 2003
% last modified 27/2
%******************************************************

%Transmit parameters

Rb = 40;
Rc = 5000;              % chip rate
Rs = 40960; % sampling rate
fs = Rs;                % sampling frequency
Tb = 1/Rb;  % bit length
Tc = 1/Rc;  % chip length
Ts = 1/fs;              % sampling interval
fcarrier = 12000; % carrier frequency
samples_per_bit  = Rs/Rb; % samples per bit
samples_per_chip = Rs/Rc; % samples per chip
chips_per_bit = Rc/Rb; % chips per bit
packet_length = 72; % number of dbits transmitted per packet
P=Tb/Tc;                  % Upsampling required to match chip sequence

% save settings to "transmit_params.mat" file

save transmit_params  Rb Rc Rs fs Tb Tc Ts fcarrier P ...
                    samples_per_bit samples_per_chip chips_per_bit...
                    packet_length ;


% Channel Simulation Parameters

N = 1;                  %number of packets
EbNo_dB = 35;           %EbNo values to use

% save channel settings to "simulation_params.mat" file

save simulation_params  N EbNo_dB;
```

```
% Tranceiver v.1
% WHITE NOISE ONLY
% developed by Georgios Pelekanos February 2003
% last modified 15/4

clear;
clc;
tic;                 % starts clock to measure simulation run time
disp('started');
load code;           % load PN chipping sequence code.mat
load transmit_params;   % load transmit parameters
load simulation_params; % load simulation parameters
load ht_stat;

EbNo_dB=linspace(1 , EbNo_dB, 10);

for pp=1:1000
pp

for kl = 1:length(EbNo_dB);  %loop through each of the EbNo values

%______________________________________________________________________________

%%%%%% %%%%%        TRANSMITTER SECTION  %%%%%%%%%%%%%%%%%%
%______________________________________________________________________________

%------------------------------------------------
% Generate a differential bit sequence of 1,-1 of
% length=N*packet_length  simulating the signal
%------------------------------------------------

signal_an=sign(randn(1,N*packet_length));

%------------------------------------------------
% Upsample by P=Tb/Tc
%------------------------------------------------

upsamples =[1,ones(1,P-1)];

for i=1:N*packet_length
   matrix(:,i) = signal_an(i).*upsamples'  ;
end

[m,n]=size(matrix);

up_sampled_an=reshape(matrix,1,m*n);

%------------------------------------------------------------------
% Chip the data using as c[n] a multiple/portion of original gold code
% first P samples of gold code only)
%------------------------------------------------------------------

% Determine repetitions of orignal gold code required
% to chip entire upsampled series

total_chips_needed = length(signal_an).*chips_per_bit;
```

```
number_of_repetitions = ceil(total_chips_needed./P);
c_n = repmat(code(1:P), 1, number_of_repetitions);%long PN sequence

%produce the transmit signal by multiplication
%which is the same as c(n)*up_sampled_an

tx_signal=c_n.*up_sampled_an;

%---------------------------------------------------
% Calculate the signal power and associated Eb/No
%---------------------------------------------------

tx_signal_power = (sum(tx_signal.^2))./length(tx_signal);

% generate the noise power associated with EbNo

EbNo(kl) = 10.^(EbNo_dB(kl)/10);
Eb = tx_signal_power.*Tb;
No = Eb./EbNo(kl);
sigma_n = sqrt(No./(2*Ts));
noise_power = sigma_n.^2;
SNR_dB = 10.*log10(tx_signal_power./noise_power);

%_____________________________________________________________________________

%%%%%%%%%%%%%%%          CHANNEL SECTION       %%%%%%%%%%%%
%_____________________________________________________________________________

%---------------------------------------------------
% Add Gaussian White noise no multipath channel
%---------------------------------------------------

noisestd = sqrt(noise_power);

 y_n = tx_signal + randn(1,length(tx_signal))*noisestd;

%_____________________________________________________________________________

%%%%%%%%%%%%%%%      RECEIVER  SECTION      %%%%%%%%%%%%%%%%
%_____________________________________________________________________________


% Create a replica of the transmitted code such that c(n)*c(-n)=delta(n)
% determine repetitions/portion of orignal gold code required
% to decode entire output

number_of_repetitions = ceil(length(y_n)/P);
c_minus_n = repmat(code(1:P), 1, number_of_repetitions);%long PN sequence

%Decode the sequence using the replica and assuming synchronization

decode_rx= y_n(1:m*n).*c_minus_n(1:m*n);%decode

%The channel estimation algorithm

% Delay by z^-1 Downsample by P new revised 3/4 works
```

```
decode_rx=reshape(decode_rx,P,length(decode_rx)/P);

%Use maximization matched filter for white noise

R_y=decode_rx*decode_rx';
[Eig_Vectors,Eig_Values] = eig(R_y);
lambda = diag(Eig_Values);
f_bar=Eig_Vectors(:,P);

% lambda=-1/norm(code(1:P));
% f_bar=lambda*code(1:P)';

%-------------------------------
%Apply the matched filter
%-------------------------------

a_hat_n=f_bar'*decode_rx ;
a_hat_n=sign(a_hat_n);

%---------------------------
%Calculate  Errors
%---------------------------
counter=find(signal_an-a_hat_n~=0);

if length(counter)<=10^-6
   channel_errors(kl)=10^-6;
else
channel_errors(kl)=length(counter);
end

summary(pp,:)=channel_errors./packet_length;
end

errors=mean(summary);

overall=[EbNo_dB;errors];

% save the simulation results

save montecarlo_white_only_144_40_50 overall ;

disp('finished');
t = toc
```

```
% Tranceiver v.2 FINAL
% MULTIPATH STATIC CHANNEL ONLY
% developed by Georgios Pelekanos February 2003
% last modified 15/4

clear;
clc;
tic;              % starts clock to measure simulation run time
disp('started');
load code;            % load PN chipping sequence code.mat
load transmit_params;   % load transmit parameters
load simulation_params; % load simulation parameters
load ht_stat;


for kl = 1:100;  %Repeat the simulation 50 times

%____________________________________________________________________________

%%%%%%%%%%%%%%%%%%%%%%%%%%%%  TRANSMITTER SECTION  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%____________________________________________________________________________

%-----------------------------------------------
% Generate a differential bit sequence of 1,-1 of
% length=N*packet_length  simulating the signal
%-----------------------------------------------

signal_an=sign(randn(1,N*packet_length));

%-----------------------------------------------
% Upsample by P=Tb/Tc
%-----------------------------------------------

upsamples =[1,ones(1,P-1)];

for i=1:N*packet_length
   matrix(:,i) = signal_an(i).*upsamples'  ;
end

[m,n]=size(matrix);

up_sampled_an=reshape(matrix,1,m*n);

%--------------------------------------------------------------------
% Chip the data using as c[n] a multiple/portion of original gold code
% first P samples of gold code only)
%--------------------------------------------------------------------

% Determine repetitions of orignal gold code required
% to chip entire upsampled series

total_chips_needed = length(signal_an).*chips_per_bit;
number_of_repetitions = ceil(total_chips_needed./P);
c_n = repmat(code(1:P), 1, number_of_repetitions);
```

```
%produce the tranmit signal by multiplication
%which is the same as c(n)*up_sampled_an

tx_signal=c_n.*up_sampled_an;

%____________________________________________________________________________

%%%%%%%%%%%%%%%%%      CHANNEL SECTION  %%%%%%%%%%%%%%%%
%____________________________________________________________________________

%---------------------------------------------------
%convolve with the channel to produce y(n)
%---------------------------------------------------

y_n = conv(tx_signal,ht_stat(590:end));

%____________________________________________________________________________

%%%%%%%%%%%%%%%%%%%% RECEIVER  SECTION %%%%%%%%%%%%%%%%%%%%
%____________________________________________________________________________

% Create a replica of the transmitted code such that c(n)*c(-n)=delta(n)
% determine repetitions/portion of orignal gold code required
% to decode entire output

number_of_repetitions = ceil(length(y_n)/P);
c_minus_n = repmat(code(1:P), 1, number_of_repetitions);%long PN sequence


%Decode the sequence using the replica and assuming synchronization

decode_rx= y_n(1:m*n).*c_minus_n(1:m*n);%decode

%The channel estimation algorithm

% Delay by z^-1 Downsample by P new revised 3/4 works

   decode_rx=reshape(decode_rx,P,length(decode_rx)/P);

%Use maximization matched filter for other than w.noise

R_y=decode_rx*decode_rx';
[Eig_Vectors,Eig_Values] = eig(R_y);
lambda = diag(Eig_Values);
f_bar=Eig_Vectors(:,P);

%-------------------------------
%Apply the matched filter
%-------------------------------

a_hat_n=f_bar'*decode_rx ;
a_hat_n=sign(a_hat_n);
```

```
%---------------------------
%Calculate  Errors
%---------------------------
counter=find(signal_an-a_hat_n~=0);
if length(counter)<=10^-6
   channel_errors(kl)=10^-6;
else
channel_errors(kl)=length(counter);
end

percent_ch_errors(kl)=channel_errors(kl)/(N*packet_length)*100;

% generate a summary of all the simulation results

   summary(:,kl) = [N*packet_length, kl, channel_errors(kl),percent_ch_errors(kl)]';

end

% save the simulation results
save simulation_static_only summary

disp('finished');
t = toc
```

```
% Tranceiver v.3 FINAL
% MULTIPATH + WHITE NOISE
% developed by Georgios Pelekanos February 2003
% last modified 15/4

clear;
clc;
tic;                    % starts clock to measure simulation run time
disp('started');
load code;              % load PN chipping sequence code.mat
load transmit_params;   % load transmit parameters
load simulation_params; % load simulation parameters
load ht_stat;

EbNo_dB=linspace(1 , EbNo_dB, 10);

for pp=1:1000
pp

for kl = 1:length(EbNo_dB);  %loop through each of the EbNo values

%________________________________________________________________________

%%%%%%%%%%%%%%%%%  TRANSMITTER SECTION  %%%%%%%%%%%%%%%%%
%________________________________________________________________________

%------------------------------------------------
% Generate a differential bit sequence of 1,-1 of
% length=N*packet_length  simulating the signal
%------------------------------------------------

signal_an=sign(randn(1,N*packet_length));

%------------------------------------------------
% Upsample by P=Tb/Tc
%------------------------------------------------

upsamples =[1,ones(1,P-1)];

for i=1:N*packet_length
   matrix(:,i) = signal_an(i).*upsamples'  ;
end

[m,n]=size(matrix);

up_sampled_an=reshape(matrix,1,m*n);

%--------------------------------------------------------------------
% Chip the data using as c[n] a multiple/portion of original gold code
% first P samples of gold code only)
%--------------------------------------------------------------------

% Determine repetitions of orignal gold code required
% to chip entire upsampled series

total_chips_needed = length(signal_an).*chips_per_bit;
```

```
number_of_repetitions = ceil(total_chips_needed./P);
c_n = repmat(code(1:P), 1, number_of_repetitions);%long PN sequence

%produce the tranmit signal by multiplication
%which is the same as c(n)*up_sampled_an

tx_signal=c_n.*up_sampled_an;

%---------------------------------------------------
% Calculate the signal power and associated Eb/No
%---------------------------------------------------

tx_signal_power = (sum(tx_signal.^2))./length(tx_signal);

% generate the noise power associated with EbNo

EbNo(kl) = 10.^(EbNo_dB(kl)/10);
Eb = tx_signal_power.*Tb;
No = Eb./EbNo(kl);
sigma_n = sqrt(No./(2*Ts));
noise_power = sigma_n.^2;
SNR_dB = 10.*log10(tx_signal_power./noise_power);

%______________________________________________________________________________

%%%%%%%%%%%%%%%% CHANNEL SECTION  %%%%%%%%%%%%%%%%%%
%______________________________________________________________________________

%---------------------------------------------------
%convolve with the channel to produce y(n)
%---------------------------------------------------

y_n = conv(tx_signal,ht_stat(590:640));

%---------------------------------------------------
% Add Gaussian White Noise
%---------------------------------------------------

noisestd = sqrt(noise_power);

 y_n = y_n + randn(1,length(y_n))*noisestd;

%______________________________________________________________________________

%%%%%%%%%%%%%%%% RECEIVER  SECTION %%%%%%%%%%%%%%%%%%
%______________________________________________________________________________


% Create a replica of the transmitted code such that c(n)*c(-n)=delta(n)
% determine repetitions/portion of orignal gold code required
% to decode entire output

number_of_repetitions = ceil(length(y_n)/P);
c_minus_n = repmat(code(1:P), 1, number_of_repetitions);%long PN sequence

%Decode the sequence using the replica and assuming synchronization
```

```
decode_rx= y_n(1:m*n).*c_minus_n(1:m*n);%decode

%The channel estimation algorithm

% Delay by z^-1 Downsample by P new revised 3/4 works

   decode_rx=reshape(decode_rx,P,length(decode_rx)/P);

%Matched filter SNR maximization Last revised 17/3 works

if channel_type == 0;

   %Use maximization matched filter for white noise

   lambda=-1/norm(c_minus_n(1:P));
   f_bar=lambda*c_minus_n(1:P)';

else

   %Use maximization matched filter for other than w.noise

   R_y=decode_rx*decode_rx';
   [Eig_Vectors,Eig_Values] = eig(R_y);
   lambda = diag(Eig_Values);
   f_bar=Eig_Vectors(:,P);

end
%-------------------------------
%Apply the matched filter
%-------------------------------

a_hat_n=f_bar'*decode_rx ;
a_hat_n=sign(a_hat_n);

%---------------------------
%Calculate  Errors
%---------------------------
counter=find(signal_an-a_hat_n~=0);
if length(counter)<=10^-6
   channel_errors(kl)=10^-6;
else
channel_errors(kl)=length(counter);
end

summary(pp,:)=channel_errors./packet_length;
end

errors=mean(summary);
overall=[EbNo_dB;errors];

% save the simulation results
save montecarlo_white_only_144_40_50 overall ;
disp('finished');
t = toc
```

```
% Tranceiver v.4
% COLORED NOISE ONLY
% developed by Georgios Pelekanos February 2003
% last modified 5/12

clear;
clc;
tic;                % starts clock to measure simulation run time
disp('started');
load code;          % load PN chipping sequence code.mat
load transmit_params;   % load transmit parameters
load simulation_params; % load simulation parameters

%___________________________________________________________________________

%%%%%%%%%%%%%% TRANSMITTER SECTION  %%%%%%%%%%%%%%%%%%%%%
%___________________________________________________________________________

EbNo_dB=linspace(1 , EbNo_dB, 10);

for pp=1:1000
pp
for kl = 1:length(EbNo_dB);  %loop through each of the EbNo values

%-------------------------------------------------
% Generate a differential bit sequence of 1,-1 of
% length=N*packet_length  simulating the signal
%-------------------------------------------------

signal_an=sign(randn(1,N*packet_length));

%-------------------------------------------------
% Upsample by P=Tb/Tc
%-------------------------------------------------

upsamples =[1,ones(1,P-1)];

for i=1:N*packet_length
   matrix(:,i) = signal_an(i).*upsamples'  ;
end

[m,n]=size(matrix);

up_sampled_an=reshape(matrix,1,m*n);

%---------------------------------------------------------------------
% Chip the data using as c[n] a multiple/portion of original gold code
% first P samples of gold code only)
%---------------------------------------------------------------------

% Determine repetitions of orignal gold code required
% to chip entire upsampled series

total_chips_needed = length(signal_an).*chips_per_bit;
number_of_repetitions = ceil(total_chips_needed./P);
c_n = repmat(code(1:P), 1, number_of_repetitions);%long PN sequence
```

```
%produce the tranmit signal by multiplication
%which is the same as c(n)*up_sampled_an

tx_signal=c_n.*up_sampled_an;

%---------------------------------------------------
% Calculate the signal power and associated Eb/No
%---------------------------------------------------

tx_signal_power = (sum(tx_signal.^2))./length(tx_signal);

% generate the noise power associated with EbNo

EbNo(kl) = 10.^(EbNo_dB(kl)/10);
Eb = tx_signal_power.*Tb;
No = Eb./EbNo(kl);
sigma_n = sqrt(No./(2*Ts));
noise_power = sigma_n.^2;
SNR_dB = 10.*log10(tx_signal_power./noise_power);
%______________________________________________________________________________

%%%%%%%%%%%%%%%% CHANNEL SECTION  %%%%%%%%%%%%%%%%%%
%______________________________________________________________________________

%---------------------------------------------------
% Add Colored noise
%---------------------------------------------------

noisestd = sqrt(noise_power);
white_noise=randn(1,length(tx_signal));
color_noise=filter(1,[1 -0.9 0.87 -0.89 0.9 -0.88],white_noise);
y_n = tx_signal + color_noise*noisestd ;

%______________________________________________________________________________

%%%%%%%%%%%%%%%% RECEIVER  SECTION %%%%%%%%%%%%%%%%%%
%______________________________________________________________________________

% Create a replica of the transmitted code such that c(n)*c(-n)=delta(n)
% determine repetitions/portion of orignal gold code required
% to decode entire output

number_of_repetitions = ceil(length(y_n)/P);
c_minus_n = repmat(code(1:P), 1, number_of_repetitions);%long PN sequence

%Decode the sequence using the replica and assuming synchronization

decode_rx= y_n(1:m*n).*c_minus_n(1:m*n);%decode

%The channel estimation algorithm

% Delay by z^-1 Downsample by P new revised 3/4 works

decode_rx=reshape(decode_rx,P,length(decode_rx)/P);
```

```
%Use maximization matched filter for colored noise
R_y=decode_rx*decode_rx';

color_noise=[0,color_noise];
R_n=color_noise*color_noise';
[Eig_n,Val_n]=eig(R_n);
Rnsq_inv=inv(sqrtm(Val_n));
R_n_sq_inv=Eig_n*Rnsq_inv*Eig_n';
R_mah=R_n_sq_inv*R_y*R_n_sq_inv;

[Eig_mah,Val_mah]=eig(R_mah);
[Eig_mah,Val_mah] = sortem(Eig_mah,Val_mah);
h_pr=Eig_mah(:,1);
h_opt=R_n_sq_inv*h_pr;

if h_opt(2:4)<0
   h_opt=-h_opt;
end

%-------------------------------
%Apply the matched filter
%-------------------------------

a_hat_n=h_opt'*decode_rx ;
a_hat_n=sign(a_hat_n);

%---------------------------
%Calculate  Errors
%---------------------------
counter=find(signal_an-a_hat_n~=0);
if length(counter)<=10^-6
   channel_errors(kl)=10^-6;
else
channel_errors(kl)=length(counter);
end

summary(pp,:)=channel_errors./packet_length;
end

errors=mean(summary);

overall=[EbNo_dB;errors];

% save the simulation results

save montecarlo_color_only_144_40_50 overall ;
disp('finished');
t = toc
```

```
% Tranceiver v.5
% COLORED NOISE and STATIC CHANNEL or DOUBLY SPREAD CHANNEL
% for static load ht_stat
% for doubly spread load ht_gaus
% developed by Georgios Pelekanos February 2003
% last modified 5/12

clear;
clc;
tic;              % starts clock to measure simulation run time
disp('started');
load code;             % load PN chipping sequence code.mat
load transmit_params;   % load transmit parameters
load simulation_params; % load simulation parameters
load ht_gaus;
%___________________________________________________________________________

%%%%%%%%%%%%%% TRANSMITTER SECTION  %%%%%%%%%%%%%%%%%%%%%
%___________________________________________________________________________

EbNo_dB=linspace(1 , EbNo_dB, 8);

for pp=1:1000
pp

for kl = 1:length(EbNo_dB);  %loop through each of the EbNo values

%-----------------------------------------------
% Generate a differential bit sequence of 1,-1 of
% length=N*packet_length  simulating the signal
%-----------------------------------------------

signal_an=sign(randn(1,N*packet_length));


%-----------------------------------------------
% Upsample by P=Tb/Tc
%-----------------------------------------------

upsamples =[1,ones(1,P-1)];

for i=1:N*packet_length
   matrix(:,i) = signal_an(i).*upsamples'  ;
end

[m,n]=size(matrix);

up_sampled_an=reshape(matrix,1,m*n);

%--------------------------------------------------------------------
% Chip the data using as c[n] a multiple/portion of original gold code
% first P samples of gold code only)
%--------------------------------------------------------------------
```

```
% Determine repetitions of orignal gold code required
% to chip entire upsampled series

total_chips_needed = length(signal_an).*chips_per_bit;
number_of_repetitions = ceil(total_chips_needed./P);
c_n = repmat(code(1:P), 1, number_of_repetitions);%long PN sequence


%produce the tranmit signal by multiplication
%which is the same as c(n)*up_sampled_an

tx_signal=c_n.*up_sampled_an;

%---------------------------------------------------
% Calculate the signal power and associated Eb/No
%---------------------------------------------------

tx_signal_power = (sum(tx_signal.^2))./length(tx_signal);

% generate the noise power associated with EbNo

EbNo(kl) = 10.^(EbNo_dB(kl)/10);
Eb = tx_signal_power.*Tb;
No = Eb./EbNo(kl);
sigma_n = sqrt(No./(2*Ts));
noise_power = sigma_n.^2;
SNR_dB = 10.*log10(tx_signal_power./noise_power);
%________________________________________________________________________

%%%%%%%%%%%%%%%CHANNEL SECTION  %%%%%%%%%%%%%%%%%%%%%%%
%________________________________________________________________________


%---------------------------------------------------
%convolve with the channel to produce y(n)
%---------------------------------------------------

% y_n = conv(tx_signal,ht_stat(590:640));
y_n = conv(tx_signal,ht_gaus(90:190));

%---------------------------------------------------
% Add Colored Noise
%---------------------------------------------------

noisestd = sqrt(noise_power);
white_noise=randn(1,length(y_n));
color_noise=filter(1,[1 -0.9 0.4 -0.3 0.2 0.1],white_noise);

y_n = y_n + color_noise*0.7*noisestd ;

%________________________________________________________________________

%%%%%%%%%%%%%%%% RECEIVER  SECTION %%%%%%%%%%%%%%%%%%%%%
%________________________________________________________________________
```

```
% Create a replica of the transmitted code such that c(n)*c(-n)=delta(n)
% determine repetitions/portion of orignal gold code required
% to decode entire output


number_of_repetitions = ceil(length(y_n)/P);
c_minus_n = repmat(code(1:P), 1, number_of_repetitions);%long PN sequence


%Decode the sequence using the replica and assuming synchronization


decode_rx= y_n(1:m*n).*c_minus_n(1:m*n);%decode

%The channel estimation algorithm

% Delay by z^-1 Downsample by P new revised 3/4 works

decode_rx=reshape(decode_rx,P,length(decode_rx)/P);


%Use maximization matched filter for colored noise
R_y=decode_rx*decode_rx';

color_noise=[0,color_noise];
R_n=color_noise*color_noise';
[Eig_n,Val_n]=eig(R_n);
Rnsq_inv=inv(sqrtm(Val_n));
R_n_sq_inv=Eig_n*Rnsq_inv*Eig_n';
R_mah=R_n_sq_inv*R_y*R_n_sq_inv;

[Eig_mah,Val_mah]=eig(R_mah);
[Eig_mah,Val_mah] = sortem(Eig_mah,Val_mah);
h_pr=Eig_mah(:,1);
h_opt=R_n_sq_inv*h_pr;

if h_opt(2:4)<0
   h_opt=-h_opt;
end

%-------------------------------
%Apply the matched filter
%-------------------------------

a_hat_n=h_opt'*decode_rx ;
a_hat_n=sign(a_hat_n);

%---------------------------
%Calculate  Errors
%---------------------------
counter=find(signal_an-a_hat_n~=0);

if length(counter)<=10^-6
   channel_errors(kl)=10^-6;
```

```
else
channel_errors(kl)=length(counter);
end

summary(pp,:)=channel_errors./packet_length;
end

errors=mean(summary);

overall=[EbNo_dB;errors];

% save the simulation results

save montecarlo_gaus_color_1242_40_100 overall ;

% %-------------------------------------------------
% % Plot overall simulation results
% %-------------------------------------------------
%
% es = summary(5,:)./(summary(2,:));

for i=1:length(EbNo_dB)
   if overall(2,i)==0
      overall(2,i)=10^-6
   end
end

figure;
      semilogy(overall(1,:),overall(2,:),'-','MarkerSize',8);
      grid
      ylim([10^-5 1]); xlim([1 22]);
      ylabel('Bit Error Rate','fontsize',13);
      xlabel('Eb/No (dB)','fontsize',13);
      set(gca,'fontsize',13);

disp('finished');
t = toc
```

```
% Blind Tranceiver when no sync
%Tranceiver No Synch with MonteCarlo
% developed by Georgios Pelekanos April 2003
% last modified 5/12

clear;
clc;
tic;                % starts clock to measure simulation run time
disp('started');
load code;          % load PN chipping sequence code.mat
load transmit_params;   % load transmit parameters
load simulation_params; % load simulation parameters
load ht_stat;
load ht_gaus;
EbNo_dB=linspace(1,EbNo_dB,8);

for pp=1:1000
pp
for kl = 1:length(EbNo_dB);  %loop through each of the EbNo values
%_____________________________________________________________________

%%%%%%%%%%%%%%%%%%%%%%%%%  TRANSMITTER SECTION  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%_____________________________________________________________________

%------------------------------------------------
% Generate a differential bit sequence of 1,-1 of
% length=N*packet_length  simulating the signal
%------------------------------------------------

signal_an=sign(randn(1,N*packet_length));
%------------------------------------------------
% Upsample by P=Tb/Tc
%------------------------------------------------

upsamples =[1,ones(1,P-1)];

for i=1:N*packet_length
   matrix(:,i) = signal_an(i).*upsamples'  ;
end

[m,n]=size(matrix);

up_sampled_an=reshape(matrix,1,m*n);

%--------------------------------------------------------------------
% Chip the data using as c[n] a multiple/portion of original gold code
% first P samples of gold code only)
%--------------------------------------------------------------------

% Determine repetitions of orignal gold code required
% to chip entire upsampled series

total_chips_needed = length(signal_an).*chips_per_bit;
number_of_repetitions = ceil(total_chips_needed./P);
c_n = repmat(code(1:P), 1, number_of_repetitions);%long PN sequence
```

```
%produce the tranmit signal by multiplication
%which is the same as c(n)*up_sampled_an

tx_signal=c_n.*up_sampled_an;
%---------------------------------------------------
% Calculate the signal power and associated Eb/No
%---------------------------------------------------
tx_signal_power = (sum(tx_signal.^2))./length(tx_signal);

% generate the noise power associated with EbNo

EbNo(kl) = 10.^(EbNo_dB(kl)/10);
Eb = tx_signal_power.*Tb;
No = Eb./EbNo(kl);
sigma_n = sqrt(No./(2*Ts));
noise_power = sigma_n.^2;
SNR_dB = 10.*log10(tx_signal_power./noise_power);

%______________________________________________________________________________

%%%%%%%%%%%%%%%%%% CHANNEL SECTION  %%%%%%%%%%%%%%%%%%%%%
%______________________________________________________________________________

% g=ht_stat(590:640);
g=ht_gaus(90:140);

%---------------------------------------------------
%convolve with the channel to produce y(n)
%---------------------------------------------------

y_n = conv(tx_signal,g);

%---------------------------------------------------
% Add Gaussian White Noise
%---------------------------------------------------

noisestd = sqrt(noise_power);

 y_n = y_n + randn(1,length(y_n))*0.7*noisestd;

%______________________________________________________________________________

%%%%%%%%%%%%%%%%%% RECEIVER  SECTION %%%%%%%%%%%%%%%%%%%%%
%______________________________________________________________________________

% To simulate lack of synchronization we add random initial time t0
% uniformly distributed between 0 and P.

t0=round(rand*P/2)+1;
y_n=y_n(t0:length(y_n));

% Create a replica of the transmitted code such that c(n)*c(-n)=delta(n)
% determine repetitions/portion of orignal gold code required
% to decode entire output
```

```
number_of_repetitions = ceil(length(y_n)/P);
c_minus_n = repmat(code(1:P), 1, number_of_repetitions);%long PN sequence

%Decode the sequence using the replica and assuming synchronization

decode_rx= y_n(1:m*n).*c_minus_n(1:m*n);%decode

%The channel estimation algorithm

% Delay by z^-1 Downsample by P new revised 3/4 works

decode_rx=reshape(decode_rx,P,length(decode_rx)/P);

%Use maximization matched filter for other than w.noise

R_y=decode_rx*decode_rx';
[Eig_Vectors,Eig_Values] = eig(R_y);
Eig_Values=diag(Eig_Values);

if Eig_Values(P-1)>3/10*Eig_Values(P)

% Assume there is no sync
e=Eig_Vectors(:,P-1:P)'*decode_rx;
E=[e(:,1:length(e)-1); e(:,2:length(e))];
Re=E*E';
[Ve, De]=eig(Re);
lam_e=diag(De);
I=find(lam_e==min(abs(lam_e)));
me=Ve(:,I);
m1=me(1:2);
m2=-me(3:4);

%-------------------------------
% Estimate the sequence
%-------------------------------
a_hat_n=[m1',m2']*E;
a_hat_n=sign(a_hat_n);

else
% Assume there is sync
f_bar=Eig_Vectors(:,P);
%-------------------------------
%Apply the matched filter
%-------------------------------
a_hat_n=f_bar'*decode_rx ;
a_hat_n=sign(a_hat_n);
end

%---------------------------
%Calculate  Errors
%---------------------------
diff=length(a_hat_n)-length(signal_an);

if diff>0
   counter=find(signal_an-a_hat_n(diff:length(signal_an)+diff-1)~=0);
elseif diff<0
```

```
   diff=abs(diff);
   counter=find(signal_an(diff:length(a_hat_n)+diff-1)-a_hat_n~=0);
else
   counter=find(signal_an-a_hat_n~=0);
end

if length(counter)<=10^-6
   channel_errors(kl)=10^-6;
else
channel_errors(kl)=length(counter);
end

summary(pp,:)=channel_errors./packet_length;
end

errors=mean(summary);

overall=[EbNo_dB;errors];

% save the simulation results

save montecarlo_no_sync_144_40 overall ;

disp('finished');
t = toc
break
```

```
function plot_result

clear
clc
%****************************************
%
% This function sets plots the results
%
% developed by Georgios Pelekanos February 2003
% last modified 27/4
%*******************************************************

load montecarlo_no_sync_144_40 ;

% %-------------------------------------------------
% % Plot overall simulation results
% %-------------------------------------------------
%

for i=1:length(EbNo_dB)
   if overall(2,i)==0
       overall(2,i)=10^-6
   end
end

figure;
       semilogy(overall(1,:),overall(2,:),'-','MarkerSize',8);
       grid
       ylim([10^-5 1]); xlim([1 23]);
       ylabel('Bit Error Rate','fontsize',13);
       xlabel('Eb/No (dB)','fontsize',13);
       set(gca,'fontsize',13);
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] Duke Peter, “Direct-Sequence Spread-Spectrum Modulation for Utility Packet Transmission in Underwater Acoustic Communication Networks”, Thesis, September 2002.

[2] J. A. Rice, R. K. Creber, C. L. Fletcher, P .A. Baxley, D .E. Rogers, and D. C. Davsion, “Seaweb Underwater Acoustic Nets,” *Space and Naval Warfare Systems Center San Diego Biennial Review 2001*, pp. 234 – 243, 2001.

[3] NATO SACLANT, “Acoustic Models”, [http://www.saclantc.nato.int/frameset-ch6.html], January 2003

[4] E. M. Sozer, J. G. Proakis, M. Stojanovic, J. A. Rice, A. Benson, and M. Hatch, “Direct-Sequence Spread-spectrum Based Modem for Under Water Acoustic Communication and Channel Measurement,” presented at IEEE OCEANS’99 Conference, Seattle, WA, September 1999.

[5] R. J. Urick, *Principles of Underwater Sound,* 3rd ed. New York: McGraw-Hill, 1983.

[6] F. B. Jensen, W. A. Kuperman, M. B. Porter, and H. Schmidt, *Computational Ocean Acoustics*. New York: Springer-Verlag, 2000.

[7] L. Brekhovskikh and Y. Lysanov, *Fundamentals of Ocean Acoustics.* Berlin: Springer-Verlag, 1982.

[8] R. Coates, *Underwater Acoustic Systems.* New York: John Wiley & Sons Inc, 1989.

[9] Lawrence J. Ziomek, *Fundamentals of Acoustic Field Theory and Space-Time Signal Processing*. Florida: CRC Press, 1995.

[10] D. B. Kilfoyle and A. B. Baggeroer, “The State of the Art in Underwater Acoustic Telemetry,” *IEEE Journal of Oceanic Engineering,* Vol. 25, pp. 4 – 27, 2000.

[11] T. S. Rappaport, *Wireless Communications Principles and Practice,* 2nd ed. New Jersey: Prentice Hall, 2002.

[12] R. L. Peterson, R. E. Ziemer, and D. E. Borth, *Introduction to Spread-spectrum Communications.* New Jersey: Prentice Hall, 1995.

[13] John G. Proakis and Dimitris Manolakis, *Digital Signal Processing, 3rd ed.* New Jersey: Prentice Hall, 1996.

[14] M. K. Tsatsanis and G. B. Giannakis, “Blind estimation of direct-sequence spread-spectrum signals in multipath”, *IEEE Trans. Signal Proc.*, Vol. 45, no. 5, pp. 1241–1252, May 1997.

[15] Charles W. Therrien, *Discrete Random Signals and Statistical Signal Processing*. New Jersey: Prentice Hall, 1992.

[16] Tan F.Wong and Tat M.Lok, “Transmitter Adaptation in Multicode DS-CDMA Systems”, *IEEE Journal on Selected Areas in Communications*, Vol. 19, no.1 pp. 69 – 82, Jan 2001.

[17] K. Karkkainen, “Optimized PN Sequences Available for Simulation of CDMA Systems,” [http://www.ee.oulu.fi/~kk/], December 2002.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Chairman, Code EC
   Electrical and Computer Engineering Department
   Naval Postgraduate School
   Monterey, California

4. Chairman, Code PH/Sk
   Engineering Acoustics Academic Committee
   Naval Postgraduate School
   Monterey, California

5. Dr. Roberto Cristi, Code EC/Cx
   Electrical and Computer Engineering Department
   Naval Postgraduate School
   Monterey, California

6. Joe Rice, Code PH/Rj
   Physics Department
   Naval Postgraduate School
   Monterey, California