1

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Services des thèses canadiennes

Ottawa, Canada
K1A 0N4

CANADIAN THESES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

**THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED**

THÈSES CANADIENNES

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

Si manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

**LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE**

AN INVESTIGATION OF BLOCK BY BLOCK
ADAPTIVE EQUALIZATION ON A
FADING HF CHANNEL

by

Gordon W. Davidson, B.Sc.

A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of
Master of Engineering

Department of Systems and Computer Engineering
Faculty of Engineering
Carleton University
Ottawa, Ontario

July 1986

© copyright G.W. Davidson, 1986

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-33430-4

The undersigned recommends to the Faculty of Graduate Studies
and Research, acceptance of the thesis

"An Investigation of Block By Block Adaptive Equalization
on a Fading HF Channel"

submitted by Gordon W. Davidson, B.Sc., in partial fulfillment
of the requirements for the degree of Master of Engineering.

Co-Supervisor

Co-Supervisor

Chairman, Department of Systems
and Computer Engineering

Carleton University

August 1986

ABSTRACT

The HF channel is known to suffer from time and frequency dispersion. To allow high speed data transmission, adaptive decision feedback equalization has been considered. The Kalman algorithm has been shown to be superior to the LMS algorithm in tracking the variation of the optimum equalizer coefficients. However, the Kalman algorithm suffers under decision direction, even when periodic retraining is used. In addition, the computational complexity is quite high. In this thesis, a block by block approach is considered. It is found that using constant equalizer coefficients over a block does not allow an adequate block length because of the occurrence of abrupt changes in the optimum coefficients. Additionally, the use of an LMS algorithm does not improve the results. Finally, it is found that linearly interpolating the coefficients between the values at the ends of the block gives good performance if the fade rate is low enough. This also leads to a great reduction in computational complexity. Again, the use of an LMS algorithm does not improve performance.

ACKNOWLEDGEMENTS

I would like to thank Dr. Falconer for his ideas, suggestions, and for reviewing this thesis. I would also like to thank Dr. Sheikh for his suggestions, and Dr. Evangelos Eleftheriou for his help with the simulations.

TABLE OF CONTENTS

Abstract	iii
Acknowledgements	iv
Table of Contents	v
List of Figures	vii
List of Symbols	viii
 CHAPTER 1 -- INTRODUCTION	 1-1
 CHAPTER 2 -- THE HF CHANNEL	 2-1
2.1 Multipath Propagation	2-1
2.2 Time Dispersion	2-4
2.3 Frequency Dispersion	2-8
2.4 Techniques for Data Transmission	2-9
 CHAPTER 3 -- EQUALIZATION OF THE HF CHANNEL	 3-1
3.1 Linear Equalization	3-1
3.2 Decision Feedback Equalization	3-3
3.3 Finite Length Decision Feedback Equalizer	3-10
 CHAPTER 4 -- ADAPTIVE ALGORITHMS	 4-1
4.1 Symbol By Symbol Algorithms	4-2
4.1.1 Stochastic Gradient	4-2
4.1.2 Recursive Least Squares	4-9
4.1.3 Lattice Filters	4-15
4.1.4 Problems for Adaptation on the HF Channel	4-18
4.2 Block Algorithms	4-23
 CHAPTER 5 -- CHANNEL SIMULATION	 5-1

CHAPTER 6 -- SIMULATION OF BLOCK BY BLOCK EQUALIZATION	6-1
6.1 Training Block Length	6-3
6.2 Data Block Length	6-9
6.3 Block By Block With Interpolation	6-21
CHAPTER 7 -- CONCLUSIONS	7-1
7.1 Concluding Discussion	7-1
7.2 Suggestions for Further Research	7-2
REFERENCES	R-1
APPENDIX A	
IIR FILTER FOR MULTIPATH TAP GAIN GENERATION	A-1
APPENDIX B	
COMPUTATION OF ADDITIVE NOISE VARIANCE	B-1
APPENDIX C	
SUBROUTINE LISTINGS FOR INITIALIZATION AND CHANNEL SIMULATION	C-1
APPENDIX D	
SUBROUTINE LISTINGS FOR COEFFICIENT ESTIMATION AND OPTIMUM COEFFICIENT CALCULATION	D-1
APPENDIX E	
LIST OF MAIN PROGRAM FOR SIMULATION OF BLOCK BY BLOCK WITH INTERPOLATION	E-1

LIST OF FIGURES

2.1	Multipath Propagation in the Ionosphere [1]	2-2
2.2	Multipath Spread and Coherence Bandwidth	2-5
2.3	Scatter Function	2-10
2.4	P_e vs. SNR for QPSK on a Fading Channel With L'th Order Diversity [2]	2-12
2.5	Ionogram [4]	2-13
3.1	Magnitude of Transfer Function of 3 Path Channel in (3.6)	3-4
3.2	DFE Structure	3-6
3.3	Finite Length DFE	3-11
4.1	Tracking of an Optimum Coefficient for $r=50$ and 100 [21]	4-10
4.2	Recursive Least Squares Lattice Equalizer [13]	4-16
4.3	Gradient Lattice Equalizer [13]	4-16
4.4	Convergence of Different Adaptive Algorithms [5]	4-19
4.5	Performance of Algorithms for Different Channel Variation Rates [2]	4-21
4.6	Performance with Periodic Retraining	4-24
5.1	Simulation Block Diagram	5-2
5.2	Pulse Shaping Filter	5-3
5.3a)	Multipath Model and Additive Noise	5-5
5.3b)	Tap Gain Generation	5-5
5.4	Tap Gain Autocorrelation Functions for a Fade Rate of One	5-7
6.1	Training Block and Data Block	6-5
6.2	Δsnr at Beginning of Data Block vs. Training Block Length for Different Fade Rates, DFE(10,4)	6-7
6.3	Δsnr at Beginning of Data Block vs. Training Block Length for Different Numbers of Equalizer Coefficients, Fixed Channel	6-8
6.4	Average Equalization Interval vs. Fade Rate	6-13
6.5a)	Equalization Interval Distributions. Fade Rate=0.2	6-15
6.5b)	Equalization Interval Distributions. Fade Rate=0.5	6-16
6.6	Real Part of Equalizer Coefficient, 2 Path Channel, Fade Rate=0.5 Hz	6-18
6.7	Variation of c_2 With a	6-20
6.8	Training Blocks Used in Interpolation	6-22
6.9	Comparison of Constant Coefficients With Interpolation. Fade Rate=0.3 HZ, Channel snr = 15 dB	6-27
6.10	Comparison of Interpolation with the Modified Algorithm. Fade Rate=0.5 Hz, Channel snr = 15 dB	6-28
6.11	Error Rate for Modified LMS and Interpolation Fade Rate=0.5, $\alpha=0.1$, Data Block Length=200	6-20
6.12	Error Rate Comparison of Kalman to Block by Block With Interpolation. Fade Rate=0.5 Hz	6-32
6.13	Error Rate Comparison of Kalman to Block by Block With Interpolation. Fade Rate=0.15 Hz	6-34

LIST OF SYMBOLS

α	constant in LMS with normalized step size
r_0	term in expansion of causal factor in (3.8)
μ	step size in LMS
$\epsilon_{\min}$	minimum mean squared error
$\epsilon_{\Delta n}$	excess mean squared error
ϵ_{est}	estimation noise
ϵ_{lag}	mean squared error due to lag
λ_i	frequency shift in Gaussian model of tap gain spectrum
λ_p	p'th eigenvalue of R
$\lambda_{\max}$	maximum eigenvalue of R
$\bar{\lambda}$	average eigenvalue
Λ	diagonal matrix with elements equal to the eigenvalues
λ	exponential weighting factor in RLS
ρ	ratio of maximum to average eigenvalue
σ_i	half the frequency spread in Gaussian model of tap gain spectrum
σ_r^2	estimated received signal power
σ_n^2	variance of additive noise
σ_s^2	average signal power
σ_u^2	signal power out of pulse shaping filter
$\sigma_{G_j}^2$	variance of tap gain in multipath model
$\tau_i(t)$	delay of path in multipath
$\phi_{nn}(m)$	correlation of additive noise

a, b	coefficients in example impulse response
$A_i(t)$	passband path gain in multipath
$b_m^2(k, n)$	squared backward predictor error in lattice filter
B	data block length
$C(\tau, \Delta t)$	correlation of time-variant channel impulse response
$C(f)$	transversal filter part of optimum linear equalizer
$\underline{c}$	vector of equalizer coefficients
$\underline{c}_{opt}$	optimum coefficients
$\Delta \underline{c}(z)$	z-transform of coefficient error vector
$\underline{c}_{opt}(z)$	z-transform of time-varying optimum coefficients
$\underline{c}_{t1}, \underline{c}_{t2}$	coefficients estimated from training blocks
$\underline{c}_\Delta, \underline{c}_\epsilon$	components of coefficients in LMS-aided interpolation
d_n	data symbol
e_n	equalizer output error
f_c	carrier frequency
f_r	fade rate
$f_m^2(k, n)$	squared forward predictor error in lattice filter
$g_i(t)$	complex baseband path gain in multipath
$G_j(t)$	resolvable path gain
$g(t)$	impulse response of optimum linear equalizer
$G(f)$	transfer function f_0 optimum linear equalizer
$G_m(n-1)$	coefficient in m 'th stage of least-squares lattice
$h(\tau, t)$	time-variant channel impulse response
$H(f, t)$	time-variant channel transfer function
$h(t)$	time-invariant channel impulse response

$H(f)$	time-invariant channel transfer function
$H_m(n-1)$	coefficient in m'th stage of least-squares lattice
k_n	Kalman gain
$k_m(n)$	coefficient in m'th stage of gradient lattice
L	number of terms in summation in covariance method
M	training block length
N	number of equalizer coefficients
N_f	number of forward filter coefficients
N_b	number of feedback filter coefficients
$n(t)$	additive noise
N_0	spectrum of additive noise
N_{lpf}	number of delays in pulse-shaping filter
N_s	number of samples per symbol in simulation
$\underline{p}$	cross-correlation vector
$\hat{\underline{p}}$	estimated cross-correlation vector
p	number of new samples per symbol interval input to equalizer
p_n	coefficient in pulse-shaping filter
$P(f)$	transfer function of pulse-shaping filter
$r(t)$	complex baseband received signal
$R(\Delta f)$	frequency correlation
R	autocorrelation matrix
$\hat{R}$	estimated autocorrelation matrix
$s(t)$	passband transmitted signal
$S(\tau, \lambda)$	scatter function
SNR_{opt}	output SNR with optimum coefficients
ΔSNR	degradation in output SNR from SNR_{opt} at the end of the training block

T	symbol interval
T_p	time constant at p'th mode in LMS
$u(t)$	complex baseband equivalent transmitted signal
$\underline{v}_n$	transformed mean coefficient error vector
w	exponential weighting factor in signal power estimation
w_n	window function
$x(t)$	passband received signal
$\underline{x}_n$	vector of windowed received signal and data samples
$X(f)$	spectrum of T-spaced matched-filter output samples
$\underline{y}_n$	vector of received signal and data samples
z_k	equalizer output sample

Acronyms

BER	bit error rate
DFE	decision feedback equalizer
FSE	fractionally spaced equalizer
HF	high frequency
ISI	intersymbol interference
LMS	least-mean-square algorithm, also known as the stochastic gradient algorithm
MLSE	maximum likelihood sequence estimation
MMSE	minimum mean squared error
MSE	mean squared error
MUF	maximum useable frequency
OWF	optimum working frequency
QPSK	quadrature phase shift keying
RLS	recursive least squares
SNR	signal to noise ratio, either in the channel or at the output of the equalizer

CHAPTER 1

INTRODUCTION

The high frequency (HF) channel refers to the 3-30 MHz range of the spectrum. This channel provides a useful and economical medium for communication over large distances, and it has been used for commercial shipping and aircraft communications [1]. Although satellite communication is increasingly used, HF should remain popular among people for whom satellite communication is too expensive or unavailable. Another use of HF is in military applications.

The major constraint on data transmission in HF is the fading and dispersive characteristics of the channel. Because of these constraints, 2.4 kbit/s transmission in a 3 KHz bandwidth is considered high rate. Such rates are required if digital voice is transmitted, or if spread spectrum is used to protect against interference. In the past, low symbol rates have been used on HF to avoid intersymbol interference (ISI) and diversity has been used to combat fading. This has resulted in low bit rates. It has been found [2], [3], however, that the use of an adaptive decision feedback equalizer may enable high speed serial transmission by removing ISI and providing a diversity effect as well.

Such adaptive equalizers are implemented digitally. However, the computational requirements of symbol by symbol adaptive algorithms are quite high. In addition, these algorithms make use of symbol by symbol decisions in forming an error signal. It has been found [2] that in

the HF channel, the occurrence of incorrect decisions may severely impair the adaptive algorithm. As a possible solution to this problem, an alternative approach to adaptive equalization is investigated. This involves the transmission of data in blocks. Each block is preceded by a known training sequence that is used to compute the equalizer coefficients. This algorithm is not affected by incorrect decisions since only known symbols are used. Also, if the block is long enough, a significant reduction in the computational requirement may be achieved.

This thesis investigates this block by block approach on a simulated HF channel. Transmission of QPSK modulated, 2.4 kbit/s data in a 3 KHz bandwidth is assumed. The remainder of the thesis is organized as follows. Chapter 2 describes the HF channel and the problems it poses for data communication. It also briefly reviews some techniques that have been used to overcome these problems, and introduces adaptive equalization as a possible solution. Chapter 3 reviews some theory on equalization, and describes why a fractionally spaced, decision feedback equalizer is suited to the HF channel. Next, Chapter 4 discusses the possible adaptive algorithms that could be used. The problems that are encountered on the HF channel are described, and the block by block algorithm is explained. Chapter 5 describes the simulation of the HF channel that is used in the investigation, and Chapter 6 presents the results. Finally, concluding remarks and suggestions for further work are given in Chapter 7.

CHAPTER 2

THE HF CHANNEL2.1 Multipath Propagation

Long distance communication in the frequency range 3-30 MHz is made possible by the ionosphere. The ionosphere consists of several layers of ionized gas located 50 to 300 kilometers above the earth. These layers have the ability to refract electromagnetic waves, so that a signal transmitted through the ionosphere will be refracted back to the earth some distance away. At this point it may be reflected by the earth back into the ionosphere, resulting in multihop propagation. Distances up to 6400 km may be covered in this manner. Modes of propagation that use the ionosphere are called skywave modes. Another mode of propagation is the groundwave, in which the signal is transmitted along the surface of the earth for up to 640 km [4].

The transmitted signal can be refracted by each layer of the ionosphere. Additionally, within each layer, there are high and low rays and two magnetoionic components for each ray. All these, combined with different numbers of hops, result in many possible modes of propagation. That is, several replicas of the transmitted signal appear at the receivers. This is called multipath propagation and is illustrated in Figure 2:1 [1].

Figure 2.1
Multipath Propagation in the Ionosphere [1]

Each mode has an associated delay and attenuation. Due to random fluctuations of the ionosphere, the delay and attenuation of each of the skywave modes varies randomly with time. Thus, if a bandpass signal $s(t)$ is transmitted, the received signal will have the form

$$x(t) = \sum_{i=1}^N A_i(t) s(t - \tau_i(t)) \quad (2.1)$$

where N is the number of skywave modes and $A_i(t)$ is the attenuation and $\tau_i(t)$ is the delay of the i^{th} mode. For large distances, the groundwave component can be neglected.

It is desired to work with the complex baseband equivalent of the system [5]. Therefore, let $u(t)$ be the complex baseband equivalent transmitted signal, defined by

$$s(t) = \text{Re}[u(t) e^{j2\pi f_c t}] \quad (2.2)$$

where f_c is the carrier frequency. The complex baseband equivalent received signal is then

$$\begin{aligned} r(t) &= \sum_{i=1}^N A_i(t) e^{-j2\pi f_c \tau_i(t)} u(t - \tau_i(t)) \\ &= \sum_{i=1}^N g_i(t) u(t - \tau_i(t)) \end{aligned} \quad (2.3)$$

where $g_i(t)$ is the random, time-varying complex gain associated with the i^{th} mode.

2.2 Time Dispersion

Since each mode has a different delay, the received signal is spread in time. To characterize this, consider the time-varying impulse response

$$h(\tau, t) = \sum_{i=1}^N g_i(t) \delta(\tau - \tau_i(t)) \quad (2.4)$$

Since this varies randomly, the spread of the channel must be described statistically. That is, the value of $h(\tau, t)$ at each delay τ is a random process. First, however, it is assumed that the statistics of these processes can be considered stationary for some interval of time. Furthermore, it is assumed that the processes for different delays are uncorrelated with each other. Then, an autocorrelation function of the impulse response for each τ can be defined as:

$$C(\tau, \Delta t) = E[h^*(\tau, t) h(\tau, t + \Delta t)] \quad (2.5)$$

where $E[]$ denotes expectation and $*$ denotes complex conjugate. Setting Δt to zero results in a function of τ , $C(\tau, 0)$, called the multipath intensity profile. This is a measure of the average power gain of the propagation mode with delay τ . Values of $C(\tau, 0)$ are significant over a certain range of τ , and the width of this range is called the multipath spread at the channel [5]. An example is shown in Figure 2.2.

Since the attenuation of the channel increases with distance, fewer modes will be significant at greater distances and the multipath spread tends to decrease. Typical values of multipath spread for the HF channel are given in Table 1 [1].

Figure 2.2
Multipath Spread and Coherence Bandwidth

Table 1

MULTIPATH SPREAD FOR THE HF CHANNEL

Distance (km)	Spread (ms)
200	8
1000	5
2500	3

The effect of the time spread on transmitted data depends on the bandwidth of the signal. For example, if the multipath spread is much less than the symbol interval, then negligible time dispersion of the signal will result. In this case it is said that the multipath is not resolvable in time. To find the signal bandwidth for which the channel is dispersive, first find the time-varying transfer function of the channel by taking the Fourier transform of the impulse response:

$$H(f,t) = \sum_{i=1}^N \exp(-j2\pi f r_i) g_i(t) \quad (2.6)$$

This is a random process over f and t , and the autocorrelation function with respect to the frequency variable is:

$$R(\Delta f) = E[H^*(f,t) H(f+\Delta f,t)] \quad (2.7)$$

This function can also be expressed as the Fourier transform of the multipath intensity profile [5]:

$$R(\Delta f) = \sum_{i=1}^N \exp(-j2\pi r_i \Delta f) C(r_i, 0) \quad (2.8)$$

as illustrated in Figure 2.2.

By definition, $R(\Delta f)$ is the correlation of the transfer function at two frequencies which are Δf apart. Thus, the largest value of Δf for

which $R(\Delta f)$ is significant indicates the largest bandwidth for which the channel can be considered nondispersive. This is called the coherence bandwidth of the channel [5]. Because of the Fourier transform relationship between $R(\Delta f)$ and $C(\tau, 0)$, the coherence bandwidth is approximately the reciprocal of the multipath spread.

If the signal bandwidth is much smaller than the coherence bandwidth, the channel is nondispersive. That is, none of the modes are resolvable in time from one another. This is the case of flat fading. If the signal bandwidth is greater than the coherence bandwidth, then some of the modes are resolvable in time and channel dispersion is present. This is known as frequency selective fading. Although the spread of the channel results in dispersion, not all the modes may be resolvable from each other. Two components are resolvable if the difference in their delays is greater than about one quarter of the reciprocal of the signal bandwidth [6]. If two or more modes are not resolvable from each other, they can be represented by one complex gain and one delay. Thus, an expression for the received signal becomes

$$r(t) = \sum_{j=1}^M G_j(t) u(t - \tau_j(t)) \quad (2.9)$$

where $G_j(t)$ is the complex gain for the j^{th} resolvable mode. For a symbol rate of 1200 symbols per second, the multipath spreads given in Table 1 indicate that the HF channel is dispersive for even large distances.

Experimental work [6] has shown that the HF channel can be adequately represented by a tapped delay line consisting of one tap for

each resolvable component. Each tap has a delay τ_j and a complex gain $G_j(t)$. Measurements on the complex gains indicated that the majority are zero mean, Gaussian random processes, producing Rayleigh fading on each path. In addition, the gains for different delays can be assumed to be statistically independent.

2.3 Frequency Dispersion

As previously mentioned, the complex gains and delays associated with each mode vary randomly with time. This results from variations in the height and refractive index of the ionospheric layers [1]. Therefore, if a continuous sine wave is transmitted, it is modulated by the relatively slowly varying gain of the channel and the spectrum of the received signal is no longer an impulse. Hence the name frequency dispersion. In addition to frequency spreading, any linear component in the time variation will also cause a frequency shift in the received signal.

To characterize the time variation of the channel, refer to the autocorrelation function of the impulse response in equation (2.5). For a given value of r , $C(r, \Delta t)$ gives the correlation of the impulse response at two instants in time, separated by Δt . The largest value of Δt for which the response is correlated with itself is referred to as the coherence time of the channel [5]. Thus a relatively small coherence time indicates a fast varying channel.

Channel variation can also be described in the frequency domain. Taking the Fourier transform of $C(r, \Delta t)$ with respect to Δt gives a two

dimensional function called the scatter function, $S(\tau, \lambda)$ [5]. An example is shown in Figure 2.3. This shows the power spectrum, as a function of Doppler frequency, λ , of the time-varying impulse response at each value of τ . Thus, the scatter function describes both the time and frequency spreading of the channel. The width of $S(\tau, \lambda)$ along the τ axis gives the multipath spread, and the width along the λ axis gives the frequency spread or Doppler spread.

When the tapped delay line model is used, the scatter function becomes a power spectral density function for each tap gain. In reference [6] it was found that a good model for the tap gain power spectrum is a Gaussian function of frequency:

$$S(\tau_i, \lambda) = \frac{C(\tau_i, 0)}{\sqrt{2\pi} \sigma_i} \exp\left(-\frac{(\lambda - \lambda_i)^2}{2\sigma_i^2}\right) \quad (2.10)$$

where λ_i is the frequency shift and $2\sigma_i$ is the Doppler spread for the i^{th} tap. The Doppler spread is typically 0.1 - 1.0 Hz [5], while the frequency shift is usually 0.01 - 1.0 Hz [6]. However, when the ionosphere is disturbed by solar flares for example, the frequency shift can be 5 - 10 Hz [6].

2.4 Techniques for Data Transmission

Multipath propagation can significantly impair data communication even when there is no time dispersion. In the case of flat fading, each mode has relatively the same delay but a different complex gain. When the complex gains add destructively, complete elimination of the signal

Figure 2.3
Scatter Function

can result. Thus, when the received signal is observed over time, it is characterized by occasional drops in signal power, called fades.

The signal to noise ratio during a fade can be extremely low, causing such a high probability of error that the channel is unusable at that time. To alleviate this problem a technique called diversity is used, in which the signal is transmitted over several independently fading channels. In so doing, the transmitted information is lost only if all channels fade simultaneously, and with enough channels the probability of this occurrence can be made very small.

To achieve diversity, independently fading channels are obtained by separating them in frequency, time, space or polarization. The improvement in performance made possible by diversity is indicated in Figure 2.4, which shows probability of error versus signal to noise ratio for different numbers of channels [2].

Flat fading can be achieved in the HF channel by a careful choice of operating frequency. This is due to the fact that at higher frequencies fewer modes survive the attenuation of the channels. The highest frequency that can be used for HF communication at any particular time is called the maximum usable frequency (MUF). Slightly below this is the optimum working frequency (OWF), at which there is generally only one resolvable path. This is illustrated in Figure 2.5 by an ionogram, which is a plot of delay versus frequency over the HF band [4].

Figure 2.4
 P_e vs. SNR for QPSK on a Fading Channel with
 L 'th Order Diversity [2]

Figure 2.5
Ionogram [4]

Typically, communication over the HF channel has been accomplished by using the optimum working frequency and combating flat fading with space diversity. In this case time dispersion is not much of a problem [1]. However, only a fraction of the allocated frequency band is used, and the use of space diversity represents a considerable overhead.

A technique that has been used to enable the use of the entire HF band is the transmission of many parallel tones within a 3 kHz bandwidth. Each tone carries information at a very low symbol rate to avoid dispersion. Also, differential detection is used in that information is encoded in the phase difference between two symbols adjacent in time or adjacent in frequency [1]. Thus, this scheme relies on the correlation of the channel over one symbol interval in time or over the tone separation in frequency. Diversity can be achieved by using redundant, widely separated tones. A problem with this scheme is its high peak to mean power ratio, resulting in inefficiency when the maximum power is constrained [2]. Also, the variation of the channel in time or frequency results in an inaccurate reference for differential detection. This causes the probability of error to approach a nonzero limiting value as the signal to noise ratio is increased [7], [8]. This irreducible error rate is a function of the time spread-frequency spread product of the channel [9].

Consider serial, 1200 symbol/second data transmission in a 3 kHz bandwidth. At frequencies where several resolvable paths are present, intersymbol interference results. However, fading is less severe than

in the nondispersive case because if one path fades, information is still carried in another. This diversity effect is inherent in the dispersive nature of the channel. Thus, if equalization can remove the ISI without significant noise enhancement, the resolvable paths can be used to achieve an improvement in performance [3].

This scheme relies on the ability of an adaptive algorithm to track the optimum equalizer setting as it varies in time. However, any adaptive algorithm will suffer some misadjustment from the optimum setting due to lag and estimation noise [10], [11]. The misadjustment from lag increases with the rate of channel variation and leads to an irreducible error rate. In addition, for a given rate of variation, the misadjustment increases with the length of the equalizer. Thus, the performance of an adaptive equalizer is also limited by the time spread-frequency spread product. Nevertheless, it has been shown that adaptive equalization has the potential for a performance that is orders of magnitude better than that of parallel tones [9].

CHAPTER 3

EQUALIZATION OF THE HF CHANNEL

In a data communication system, the term equalization refers to the minimization of intersymbol interference (ISI). Referring again to a complex baseband equivalent model, let $\{d_n\}$ be the transmitted sequence of complex data symbols, and $n(t)$ be additive noise. The term channel will now refer to the combination of the transmission medium, transmitter filters, and receiver filters, and will be described by the impulse response $h(t)$. The received signal is then

$$r(t) = \sum_n d_n h(t-nT) + n(t) \quad (3.1)$$

where T is the symbol interval. Given this signal it is desired to detect the transmitted sequence. However, it can be seen from equation (3.1) that a T -spaced sample of $r(t)$ contains not only the current symbol but a combination of other symbols as well. This is the intersymbol interference due to the dispersion of the channel.

3.1 Linear Equalization

A linear receiver for estimation of the transmitted sequence is simply a filter with impulse response, say $g(t)$. The symbol rate sampled output of this filter is then:

$$z_k = \int_{-\infty}^{\infty} g(t) r(kT-t) dt \quad (3.2)$$

and the error between z_k and the transmitted symbol d_k consists of ISI and noise. When the noise is white and the data symbols are uncorrelated with unit power, the minimization of mean squared error (MSE) gives an optimum filter with frequency response [12], [13]:

$$G(f) = \frac{H^*(f)}{N_0 + X(f)} = H^*(f) C(f) \quad (3.3)$$

$H(f)$ is the transform of $h(t)$ and N_0 is the power spectral density of the noise in (3.1). $X(f)$ is given by:

$$X(f) = \frac{1}{T} \sum_n |H(f - \frac{n}{T})|^2 \quad f' \leq \frac{1}{2T} \quad (3.4)$$

and can be interpreted as the signal spectrum of a matched filter output, after being sampled at the symbol rate.

In (3.3), $G(f)$ is written as the cascade of a matched filter and a filter with transfer function $C(f)$. Note that $C(f)$ is periodic in frequency with period $1/T$ and thus represents a T -spaced transversal filter. The symbol rate sampled input to $C(f)$ has a signal spectrum $X(f)$, and the output has a spectrum $X(f)/[N_0 + X(f)]$. Thus the transversal equalizer attempts to remove ISI by flattening the aliased signal spectrum at its input.

The resulting noise power-spectral density at the output of the receiver is

$$\frac{N_0 X(f)}{(N_0 + X(f))^2} \quad (3.5)$$

which, for relatively high signal to noise ratios, becomes large at frequencies where $X(f)$ is small. Thus, when there are dips in the aliased spectrum at the input to the equalizer, the attempt to remove ISI results in noise enhancement.

The performance of a linear equalizer, when used on the HF channel, suffers significant degradation because of noise enhancement. This is due to the severe amplitude variation in the magnitude response of a multipath channel. As a particular example, consider a three path channel with impulse response:

$$a\delta(t) + \delta(t-2T) + a\delta(t-4T) \quad (3.6)$$

where $0 \leq a \leq 0.5$. The magnitude of the corresponding transfer function is

$$1 + 2a \cos(2\pi f 2T) \quad (3.7)$$

which is illustrated in Figure 3.1. The variation in frequency is evident, with a null forming as a approaches 0.5. The poor performance of a linear equalizer in the presence of nulls in the frequency response is well known [2], [5], [14].

3.2 Decision Feedback Equalization

Because of the presence of postcursors and precursors in the received pulse, intersymbol interference exists from symbols transmitted before and after the one being detected. At any point in time, however, the previously transmitted symbols have already been detected. The decision feedback equalizer makes use of these symbols to subtract off

Figure 3.1
Magnitude of Transfer Function
of 3-Path Channel in (3.6)

the corresponding ISI. Since detected symbols are used, there is no degradation due to noise. However, the price paid for this is the possibility of an incorrect decision that causes the wrong quantity to be subtracted. When this happens, the probability of making an error in the following symbols increases, and error propagation may result.

The structure of the decision feedback equalizer (DFE) is shown in Figure 3.2. The feedback filter, $B(f)$, is used to subtract the ISI that occurred from previously detected symbols, and $W(f)$ is the forward filter for minimization of ISI from as yet undetected symbols. The output of the equalizer, $\{\hat{d}_n\}$, is an estimate of the transmitted sequence. Assuming correct decisions, minimization of the mean squared error between the input to the quantizer and the transmitted symbol gives the optimal $W(f)$ and $B(f)$. While the assumption of correct decisions will obviously give optimistic results, it has been shown that, in the presence of error propagation, the DFE can still give better performance than the linear equalizer [15].

The expressions for $W(f)$ and $B(f)$ can be explained in terms of the structure of the DFE. Since $B(f)$ acts only on previously detected symbols, it can only subtract ISI that is causal. Thus the job of $W(f)$ is to minimize the anticausal ISI, within the constraint provided by additive noise.

The optimal forward filter consists of a matched filter followed by the anticausal factor of $C(f)$ in equation (3.3). Let

$$N_0 + X(f) = ([N_0 + X(f)]_+) ([N_0 + X(f)]_-) \quad (3.8)$$

Figure 3.2
DFE Structure

where $[N_0 + X(f)]_+$ is the causal factor and can be written [14]

$$[N_0 + X(f)]_+ = \sum_{n=0}^{\infty} \gamma_n e^{-j2\pi fnT} \quad (3.9)$$

The optimum $W(f)$ is [14]:

$$W(f) = \frac{H^*(f)}{\gamma_0 [N_0 + X(f)]_-} \quad (3.10)$$

and, to cancel causal ISI, $B(f)$ has the form [14]:

$$B(f) = \frac{[N_0 + X(f)]_+}{\gamma_0} - 1 \quad (3.11)$$

At this point some observations can be made about the DFE. First, note that the signal spectrum after sampling the output of $W(f)$ is

$$\frac{X(f)}{\gamma_0 [N_0 + X(f)]_-} \neq \frac{[N_0 + X(f)]_+}{\gamma_0} \quad (3.12)$$

which indicates that, for the MMSE criterion some residual ISI remains.

Also note that the noise power spectral density after sampling the forward filter output is

$$\frac{N_0 X(f)}{\gamma_0 (N_0 + X(f))} \quad (3.13)$$

By comparing (3.5) to (3.13) it can be seen that the DFE suffers from far less noise enhancement than the linear equalizer. Thus, since the forward filter is only required to minimize anticausal ISI, it does not have to completely remove amplitude variation, and less noise enhancement results.

This improvement in the presence of amplitude distortion can also be seen from the expression for mean squared error [14]:

$$MSE_{DFE} = \exp \left[T \int_{-\frac{1}{2T}}^{\frac{1}{2T}} \ln \left[\frac{1}{1 + \frac{X(f)}{N_0}} \right] df \right] \leq T \int_{-\frac{1}{2T}}^{\frac{1}{2T}} \frac{1}{1 + \frac{X(f)}{N_0}} df = MSE_{linear} \quad (3.14)$$

The DFE and linear equalizer have the same MSE only when $X(f)$ is constant, and the MSE for the DFE becomes smaller when $X(f)$ varies.

Therefore, due to its superior performance in the presence of amplitude distortion, the DFE is more suited for use on the HF channel.

In a time-varying channel, the DFE must be made adaptive. Also, for a given channel response, an equalizer with more coefficients is slower to adapt. This consideration leads to two possible implementations of the forward filter. The first is a T-spaced transversal filter. The second is a transversal filter with a tap spacing that is some fraction of the symbol interval, most commonly $T/2$.

The T-spaced filter is suboptimal because it operates on an aliased signal when no matched filtering has been done. This makes the performance sensitive to timing phase [2], [12]. The fractionally spaced filter, on the other hand, can directly realize the optimum forward filter because there is no aliasing at its input. The disadvantage of the fractionally spaced equalizer (FSE) is that it requires more coefficients to cover the same time span.

An equalizer that is insensitive to timing phase is particularly important for the HF channel. This is because the timing phase is difficult to track on HF, and also because of the multipath structure. Since delays between paths may not be multiples of the symbol interval,

a timing phase that is best for one path may not be best for another. Thus a fractionally spaced equalizer allows a greater resolution of all paths, taking more advantage of the potential diversity [2].

The increased number of coefficients in the FSE causes slower adaptation, but this is compensated by the FSE's advantages for all but relatively high rates of variation. In [6] it was found that the performance of a DFE with a $T/2$ -spaced forward filter was better than that of a DFE with a T -spaced forward filter for frequency spreads less than 2 Hz.

The DFE can be thought of as a kind of nonlinear processing which allows a reduction in mean squared error from that of a linear filter. Another type of nonlinear reception is maximum likelihood sequence estimation (MLSE). This method directly minimizes probability of error over the entire transmitted sequence, when the noise is Gaussian, and is thus the theoretically optimum receiver.

For QPSK and an impulse response length of L symbol intervals, MLSE requires a computation complexity proportional to 4^{L+1} operations per symbol [5]. Since, at 1200 symbols/second, the impulse response for the HF channel can reach 10 symbol intervals or more, the complexity of MLSE is prohibitive. Also, MLSE requires knowledge of the channel impulse response, so that in a time-varying environment an adaptive system identification algorithm must be used as well. Finally, it was concluded in [16] that the theoretical performance of the DFE was within 1 dB of that of MLSE, so that the DFE offers a better performance/complexity tradeoff than does MLSE.

3.3 Finite Length Decision Feedback Equalizer

The transversal filters discussed above are theoretically infinite in length. In practice, however, the DFE is implemented by finite length filters. In this section the solution for the optimum, finite set of DFE coefficients is presented, as well as some comments on the length of the filters.

Consider the case where the input to the forward filter are $T/2$ spaced samples of the received signal $r(t)$. In addition, assuming the decisions are correct, the input to the feedback filter are the T spaced data symbols d_n . The structure is shown in Figure 3.3, where N_f is the number of forward filter coefficients and N_b is the number of feedback coefficients.

The input to the quantizer is z_n , given by

$$z_n = \sum_{k=0}^{N_f-1} f_k r(nT - k \frac{T}{2}) + \sum_{k=1}^{N_b} b_k^* d_{n-k} \quad (3.15)$$

which can also be represented in vector form as

$$z_n = \underline{c}^{*T} \underline{y}(n) \quad (3.16)$$

where

$$\underline{c}^T = [f_0, f_1, \dots, f_{N_f-1} \vdots b_1, \dots, b_{N_b}]$$

$$\underline{y}(n)^T = [r(nT) \ r(nT - \frac{T}{2}) \ \dots \ r(nT - (N_f-1)\frac{T}{2}) \vdots d_{n-1}, \dots, d_{n-N_b}]$$

The mean squared error between the data symbol and the quantizer input is

Figure 3.3
Finite Length DFE

then written as

$$\text{MSE} = E[|d_n - \underline{c}^{*T} \underline{y}(n)|^2] \quad (3.17)$$

By defining an autocorrelation matrix

$$\underline{R} = E[\underline{y}(n) \underline{y}(n)^{*T}] \quad (3.18)$$

and a cross correlation vector

$$\underline{p} = E[d_n \underline{y}(n)^{*}] \quad (3.19)$$

the MSE can also be expressed as [17]:

$$\text{MSE} = 1 - \underline{p}^{*T} \underline{R}^{-1} \underline{p} + (\underline{R}^{-1} \underline{p} - \underline{c})^{*T} \underline{R} (\underline{R}^{-1} \underline{p} - \underline{c}) \quad (3.20)$$

where it has been assumed that $E[|d_n|^2] = 1$.

The solution for the coefficients which minimize the MSE is seen to be

$$\underline{c}_{\text{opt}} = \underline{R}^{-1} \underline{p} \quad (3.21)$$

and the minimum mean squared error is:

$$\text{MMSE} = 1 - \underline{p}^{*T} \underline{R}^{-1} \underline{p} \quad (3.22)$$

Also, it can be seen that for any suboptimum set of coefficients $\underline{c}$, the resulting mean squared error is

$$\text{MSE} = \text{MMSE} + (\underline{c}_{\text{opt}} - \underline{c})^{*T} \underline{R} (\underline{c}_{\text{opt}} - \underline{c}) \quad (3.23)$$

where the second term on the right is an excess MSE over the minimum.

Finally, since the $T/2$ spaced samples of $r(t)$ can be written as

$$r(m \frac{T}{2}) = \sum_n d_n h(\frac{mT}{2} - nT) + n(\frac{nT}{2}) \quad (3.24)$$

and the data symbols are assumed uncorrelated, the matrix R and vector p can be expressed as a function of the channel impulse response and noise correlation. Because of the two different filters in a DFE, R has the form of a partitioned matrix

$$R = \begin{bmatrix} R_{11} & R_{21} \\ R_{12} & R_{22} \end{bmatrix} \quad (3.25)$$

R_{11} is an $N_f \times N_f$ Hermitian matrix with elements:

$$R_{11}(i,j) = \sum_k h(kT - (i-1)\frac{T}{2}) h^*(kT - (j-1)\frac{T}{2}) + \epsilon_{nn}((i-j)\frac{T}{2}) \quad (3.26)$$

$$1 \leq i, j \leq N_f$$

where $\epsilon_{nn}(\cdot)$ is the autocorrelation function of the noise in the channel.

Also, R_{12} is an $N_b \times N_f$ matrix with elements

$$R_{12}(i,j) = h^*(iT - (j-1)\frac{T}{2})$$

$$1 \leq i \leq N_b, \quad 1 \leq j \leq N_f \quad (3.27)$$

and $R_{12} = R_{21}^{*T}$. Lastly, R_{22} is an $N_b \times N_b$ identity matrix. The vector p has $N_f + N_b$ elements given by

$$\begin{aligned} p(j) &= h(-(j-1)\frac{T}{2}) \quad 1 \leq j \leq N_f \\ &= 0 \quad N_f + 1 \leq j \leq N_f + N_b \end{aligned} \quad (3.28)$$

The feedback filter must have enough taps to be able to subtract off-causal ISI. Thus, for an HF channel it need only span the multipath spread. The forward filter, on the other hand, tries to realize the response in (3.10), and theoretically an infinite number of taps are required. However, an increased number of coefficients slows adaptation, so a certain tradeoff is involved. In [2] it was found that a good compromise was reached when the forward section spanned 1 - 1.5 times the multipath spread.

CHAPTER 4

ADAPTIVE ALGORITHMS

The optimum coefficients of a decision feedback equalizer are chosen to minimize the mean squared error. If the channel impulse response and noise correlation are known, then these coefficients can be computed exactly by solving the system of equations in (3.21). When the channel characteristics are unknown, however, an adaptive algorithm must be used to estimate the optimum coefficients. Such an algorithm makes use of the received signal samples and the transmitted data symbols to minimize some performance function of the squared error. Further difficulties arise when the channel is time-varying. In this case the optimum coefficient vector varies in time, and the adaptive algorithm must not only estimate this vector, but also track it as it changes.

As indicated above, coefficient estimation requires knowledge of the transmitted data symbols. These can be provided for the adaptive equalizer by transmitting a known pseudorandom sequence that is generated synchronously at the receiver. This is the ideal reference mode. During actual data transmission when the transmitted sequence is unknown to the receiver, the equalizer can use decisions in place of known symbols. This is known as the decision directed mode.

Two approaches are possible in estimating and tracking the optimum coefficients. The most common approach is symbol by symbol estimation. In this case the error between the transmitted symbol and the quantizer input at each symbol interval is used to update the coefficient vector. Usually, the ideal reference mode is used during an initial training period to allow the algorithm to converge. When the mean squared error reaches an acceptable level, data transmission is allowed to begin, and decision direction is used to follow changes in the optimum coefficient vector.

Another approach is to divide the data stream into blocks. Each data block is preceded by a known pseudorandom sequence referred to as the training block. Using this known sequence the equalizer coefficients are computed, and these coefficients are used to equalize the channel for the data block that follows.

In this chapter some symbol by symbol algorithms are described, as well as problems encountered in the HF channel. Following this, some block algorithms for coefficient estimation are discussed.

4.1 Symbol by Symbol Algorithms

4.1.1 Stochastic Gradient

The stochastic gradient (or LMS) algorithm is based on the steepest descent method for minimization of the mean squared error. This method makes use of the fact that the MSE in equation (3.20) is a quadratic function of the equalizer coefficients, and thus has a unique minimum.

Let N be the dimension of the coefficient vector. Such a minimum can be visualized by imagining an N -dimensional space of equalizer coefficients, in which the MSE surface has the shape of a paraboloid. Any point on this surface has a corresponding vector of coefficients. The steepest descent method works by taking the gradient of the MSE, multiplying by a small constant, and subtracting this from the vector of coefficients. This moves the MSE toward the minimum, and the coefficient vector toward the optimum. The iteration is described by the equation [17]

$$\underline{c}_{n+1} = \underline{c}_n - \frac{\Delta}{2} \frac{\partial}{\partial \underline{c}_n} E[|e_n|^2] \quad (4.1)$$

where $\underline{c}_n$ is the N -dimensional vector of coefficients and Δ is a small constant. The error is given by

$$e_n = d_n - \underline{c}_n^{*T} \underline{y}_n \quad (4.2)$$

as discussed in Section 3.3. Taking the gradient gives

$$\underline{c}_{n+1} = \underline{c}_n + \Delta E[\underline{y}_n e_n^*] \quad (4.3)$$

Since the expectation in equation (4.3) is generally unavailable, the stochastic gradient algorithm approximates the steepest descent method by using a noisy but unbiased estimate of the gradient. The algorithm becomes [18]

$$\underline{c}_{n+1} = \underline{c}_n + \Delta \underline{y}_n e_n^* \quad (4.4)$$

Note that because of the noisy estimate of the gradient, the coefficient vector will have some random fluctuation about a mean value.

Two considerations for an adaptive algorithm are: (1) convergence to an unknown but constant optimum coefficient vector, $\underline{c}_{opt}$; and (2) tracking such a vector when it varies in time. An algorithm with faster convergence should also be capable of faster tracking. Additionally, an insight into limitations of the algorithm's speed of adaptation can be gained by examining its convergence properties.

The convergence of the stochastic gradient algorithm can first be described by examining the behavior of the mean coefficient vector. Taking expectations of both sides of (4.4), assuming $\underline{c}_n$ and $\underline{y}_n$ are independent [19], gives

$$E[\underline{c}_{n+1}] = (I - \Delta R) E[\underline{c}_n] + \Delta \underline{p}. \quad (4.5)$$

Subtracting $\underline{c}_{opt}$ from both sides and performing a similarity transformation results in

$$\underline{v}_{n+1} = (I - \Delta \Lambda) \underline{v}_n \quad (4.6)$$

where $\underline{v}_n$ is a transformed mean coefficient error vector and Λ is a diagonal matrix whose elements are the eigenvalues of R . From (4.6) it can be seen that the convergence of the mean coefficient error depends on N modes, one for each eigenvalue. The time constant, in symbol intervals, of the mode corresponding to the eigenvalue λ_p is [10]

$$\tau_p = \frac{1}{\Delta \lambda_p}. \quad (4.7)$$

Thus the convergence is limited by the minimum eigenvalue but increases with the step size. However, from (4.6) it can also be seen that for

stability the step size must satisfy

$$0 < \Delta < \frac{2}{\lambda_{\max}} \quad (4.8)$$

where $\lambda_{\max}$ is the maximum eigenvalue of R . Therefore, a large spread in the eigenvalues results in a slower convergence of the mean coefficient vector.

Another consideration of the algorithm is the random fluctuation of $\underline{c}_n$ about its mean. A larger step size will increase the variance of these fluctuations, which may in itself lead to instability. Recall from Section 3.3 that the mean squared error for a given coefficient vector may be written as

$$\epsilon_{\min} + (\underline{c}_{\text{opt}} - \underline{c}_n)^* R (\underline{c}_{\text{opt}} - \underline{c}_n) \quad (4.9)$$

where $\epsilon_{\min}$ denotes the minimum MSE. Averaging over $\underline{c}_n$ gives

$$\begin{aligned} \epsilon_{\min} + E[(\underline{c}_{\text{opt}} - \underline{c}_n)^* R (\underline{c}_{\text{opt}} - \underline{c}_n)] \\ = \epsilon_{\min} + \epsilon_{\Delta n} \end{aligned} \quad (4.10)$$

where $\epsilon_{\Delta n}$ is the average excess mean squared error. This is due to both the mean coefficient error and the random fluctuations of $\underline{c}_n$.

In [20] an approximate relation is found for the convergence of $\epsilon_{\Delta n}$, and is expressed in [13] as

$$\epsilon_{\Delta n+1} = [1 - 2\Delta\bar{\lambda} + \Delta^2 N_c(\bar{\lambda})^2] \epsilon_{\Delta n} + \Delta^2 \epsilon_{\min} N_c(\bar{\lambda})^2 \quad (4.11)$$

where $\bar{\lambda}$ is the average of the eigenvalues of R and ρ is the ratio of the

maximum to average eigenvalue. Although these results were derived for the linear equalizer, it is assumed that the approximations would hold adequately for the DFE. The range of Δ for stability is found to be

$$0 < \Delta < \frac{2}{N_{\text{L}} \bar{\lambda}} = \frac{2}{N_{\text{L}} \lambda_{\text{max}}} \quad (4.12)$$

which is much smaller than that of (4.8). This indicates that increasing the number of coefficients also limits the adaptation speed.

After convergence of the algorithm an excess MSE remains because of the random fluctuations of $\underline{c}_n$. This is referred to as estimation noise. A value for the excess MSE can be obtained by finding the steady state value of $\epsilon_{\Delta n}$ in (4.11), and the result is [13]

$$\epsilon_{\text{est}} = \frac{\Delta \epsilon_{\text{min}} N_{\text{L}} \bar{\lambda}}{2 - \Delta N_{\text{L}} \bar{\lambda}} \quad (4.13)$$

Note that $N_{\text{L}} \bar{\lambda} = \text{tr}(R)$, the trace of the autocorrelation matrix [20]. In general, this will increase with the signal power. In a time varying channel such as the HF channel, an increase in signal power could cause the step size to exceed the stability bound in (4.12). Thus it is desirable to use a normalized step size such as

$$\Delta = \frac{\alpha}{\text{tr}(R)} \quad (4.14)$$

where α is a small constant that must satisfy

$$0 < \alpha < \frac{2}{\bar{\lambda}} \quad (4.15)$$

The value of α can be chosen for a worst case value of ϵ , and the step size computed as in (4.14). To estimate $\text{tr}(R)$, note that for a fractionally spaced DFE:

$$\begin{aligned} \text{tr}(R) = E[|r(nT)|^2 + |r(nT - \frac{T}{2})|^2 + \dots + |r(nT - (N_f - 1)\frac{T}{2})|^2 \\ + |d_{n-1}|^2 + \dots + |d_{n-N_b}|^2] \end{aligned} \quad (4.16)$$

Assuming the signal power to stay approximately constant over the length of the equalizer, we get

$$\text{tr}(R) \approx \frac{N_f}{2} E[|r(nT)|^2 + |r(nT - \frac{T}{2})|^2] + N_b \quad (4.17)$$

Then, the average power of the signal, $E[|r(nT)|^2 + |r(nT - \frac{T}{2})|^2]/2$, can be estimated from the received signal samples as

$$\epsilon_r^2(n+1) = w \epsilon_r^2(n) + (1-w) \left[\frac{|r(nT)|^2 + |r(nT - \frac{T}{2})|^2}{2} \right] \quad (4.18)$$

where W is an exponential weighting factor that is less than but close to one. Finally, an estimate of $\text{tr}(R)$ is:

$$\text{tr}(R) \approx N_f \epsilon_r^2 + N_b \quad (4.19)$$

Using $\alpha \approx \alpha/\text{tr}(R)$ in (4.13), the resulting excess MSE due to estimation noise is

$$\epsilon_{\text{est}} = \frac{\alpha}{2 - \alpha} \epsilon_{\text{min}} \quad (4.20)$$

For values of α on the order of $1/\epsilon$, ϵ_{est} is approximately ϵ_{min} .

As mentioned above, an adaptive algorithm must be able to track changes in the optimum coefficient vector. Because an algorithm cannot react instantly, there will be some lag between the optimum coefficients and the estimated ones. In the steady state the two sources of excess mean squared error, estimation noise and lag, can be distinguished as follows. The coefficient error vector is expressed as the sum of two parts [10]

$$\underline{c}_n - \underline{c}_{opt}(n) = (\underline{c}_n - E(\underline{c}_n)) + (E(\underline{c}_n) - \underline{c}_{opt}(n)) \quad (4.21)$$

The first part represents the fluctuation of $\underline{c}_n$ about its mean, and the second term is the lag of the mean coefficients from the optimum ones. Using (4.21) the excess mean squared error is written as [10]:

$$\begin{aligned} \epsilon_e &= E[(\underline{c}_n - E(\underline{c}_n))^* R(n) (\underline{c}_n - E(\underline{c}_n))] \\ &\quad + E[(E(\underline{c}_n) - \underline{c}_{opt}(n))^* R(n) (E(\underline{c}_n) - \underline{c}_{opt}(n))] \\ &= \epsilon_{est}(n) + \epsilon_{lag}(n) \end{aligned} \quad (4.22)$$

The first term is due to estimation noise and the second is due to lag. The mean squared error of the DFE in a time varying channel is then

$$\begin{aligned} MSE(n) &= \epsilon_{min}(n) + \epsilon_{est}(n) + \epsilon_{lag}(n) \\ \epsilon_{min}(n) &= 1 - \underline{p}(n)^* \underline{c}_{opt}(n) \end{aligned} \quad (4.23)$$

and all quantities depend on the channel characteristic of that time.

The value for $\epsilon_{\text{lag}}(n)$ will also depend on the adaptive algorithm and on the nature of the channel variation. Recall that (4.7) gives the time constant for convergence of the p 'th mode. In the case of equalization, however, the eigenvalues of R are time varying. This makes a characterization of $\epsilon_{\text{lag}}(n)$ in terms of the algorithm parameters very difficult. In [21], however, computer simulation was used to demonstrate the lag of an estimated coefficient. In Figure 4.1, their results for the LMS algorithm with a normalized step size are presented. The variable τ in the figure corresponds to $1/\alpha$. As can be seen, a larger α allows a faster adaptation. This would decrease $\epsilon_{\text{lag}}(n)$ but, according to equation (4.20), this would also increase the estimation noise. Thus there is a tradeoff between lag and estimation noise when selecting the parameters of the adaptive algorithm.

4.1.2 Recursive Least Squares

Strictly speaking, finding the mean of the squared error requires an average over the ensemble. Since this is generally unavailable, an alternative performance criterion can be used. Such a criterion is a sum of the squared errors, which leads to the class of least squares adaptive algorithms. Recursive least squares (RLS) algorithms minimize the function [5]

$$\sum_{k=0}^n \lambda^{n-k} \|d_k - \underline{c}_n^* Y_k\|^2 \quad (4.24)$$

with respect to the coefficient vector $\underline{c}_n$. Note that this function assumes that $\underline{c}_n$ is used in forming all errors up to the present time. However, the constant λ , which is less than but close to one, provides

Figure 4.1

Tracking of an Optimum Coefficient
for $\tau = 50$ and 100 [21]

an exponential weighting of past errors. This allows the algorithm to follow changes in the optimum coefficient vector.

The solution for $\underline{c}_n$ is given by [5]:

$$\underline{c}_n = \hat{R}_n^{-1} \hat{p}_n \quad (4.25)$$

where

$$\hat{R}_n = \sum_{k=0}^n \lambda^{n-k} \underline{y}_k \underline{y}_k^{*T} \quad (4.26)$$

$$\hat{p}_n = \sum_{k=0}^n \lambda^{n-k} d_k^* \underline{y}_k \quad (4.27)$$

This solution can be generated recursively by making use of the fact that

$$\hat{R}_n = \lambda \hat{R}_{n-1} + \underline{y}_n \underline{y}_n^{*T} \quad (4.28)$$

$$\hat{p}_n = \lambda \hat{p}_{n-1} + d_n^* \underline{y}_n \quad (4.29)$$

Using the matrix inversion lemma, $\hat{R}_n^{-1}$ can be expressed in terms of $\hat{R}_{n-1}^{-1}$ and $\underline{y}_n$. Combining this with the recursive definition of $\hat{p}_n$ gives the following algorithm for $\underline{c}_n$ [5]:

$$\underline{c}_n = \underline{c}_{n-1} + \underline{k}_n e_n^* \quad (4.30)$$

where

$$e_n = d_n - \underline{c}_{n-1}^T \underline{y}_n \quad (4.31)$$

The vector $\underline{k}_n$, referred to as the Kalman gain, is defined as

$$\underline{k}_n = \hat{R}_n^{-1} \underline{y}_n \quad (4.32)$$

A recursive least squares algorithm was first derived by Godard [22] by applying Kalman filtering theory to the estimation of constant optimum coefficients. It was found that convergence of the mean squared error to within 3 dB of the minimum was obtained in $2N$ iterations. Note that $(1/n)\hat{R}_n$ and $(1/n)\hat{p}_n$ are estimates of R and p . Thus the n 'th coefficient estimate is the best possible for the received sequence. In this sense the RLS (or Kalman) algorithm has the fastest convergence because it makes the best use of all the data up to time n .

An important advantage of the RLS algorithm is its independence of the eigenvalues of R . To see this, substitute for k_n into (4.30):

$$c_n = c_{n-1} + \hat{R}_n^{-1} y_n e_n^* \quad (4.33)$$

This expression is similar to the LMS algorithm, except that the step size μ is replaced by an estimate of the inverse of R . This effectively performs a transformation to orthogonalize the signal vector, and the resulting eigenvalues are approximately equal [13].

As mentioned above, $(1/n)\hat{R}_n$ and $(1/n)\hat{p}_n$ are estimates of R and p . However, the use of exponential weighting results in a finite window effect. That is, even as n approaches infinity, $(1/n)\hat{R}_n$ and $(1/n)\hat{p}_n$ will be misadjusted from their optimal values. As a result, some coefficient error will remain after convergence. This is an estimation noise which leads to an excess mean squared error, ϵ_{est} . For λ close to 1, the steady state value of ϵ_{est} is approximately [11]

$$\epsilon_{est} = \frac{1-\lambda}{1+\lambda} N \epsilon_{min} \quad (4.34)$$

In a time varying environment, there will also be an excess mean squared error due to lag. Equations (4.22) and (4.23) describing the components of the mean squared error apply equally well to the RLS algorithm. However in this case a more mathematical description of the lag error vector, $\underline{\epsilon}_n = E(\underline{c}_n) - \underline{c}_{opt}(n)$, can be given. Assuming that the channel variations are slow with respect to the memory of the algorithm, it is shown in [11] that

$$E(\underline{c}_n) = E(\underline{c}_{n-1}) + (1-\alpha) (\underline{c}_{opt}(n-1) - E(\underline{c}_{n-1})) \quad (4.35)$$

Letting $\underline{\epsilon}_c(z)$ be the z transform of $\underline{\epsilon}_n$, this gives the relation [11]

$$\underline{\epsilon}_c(z) = \frac{z^{-1} - 1}{1 - \alpha z^{-1}} \underline{c}_{opt}(z) \quad (4.36)$$

This transfer function represents a system whose input is the optimum coefficient vector, and whose output is the lag error vector. The error vector is thus a geometrically converging process, and each component has a time constant of

$$\frac{1}{1-\alpha} \quad (4.37)$$

Note that in this case, there is no dependence on the eigenvalues of R .

An exact calculation of ϵ_{lag} would depend on the nature of the channel variation. However, some observations can be made about the tradeoff between lag and estimation noise. Note that a value of α closer to one decreases estimation noise but increases lag. Also, if ϵ_{est} is held constant, an increase in N in equation (4.34) necessitates an increase in α . Thus, for a given estimation noise, the adaptation speed is again limited by the number of coefficients.

Despite its fast convergence and tracking ability, the Kalman algorithm has some disadvantages, namely, computational complexity and numerical instability. As originally derived, the algorithm requires about $3N^2 + 3N$ multiplications per symbol interval [23]. Additionally, when implemented with 32 bit floating point arithmetic, accumulated roundoff errors causes instability of the coefficient values [2]. To overcome these problems, other versions of the RLS algorithm have been derived. Although these algorithms differ in their computational efficiency and their numerical properties, they minimize the same performance function and thus have the same convergence.

A 'fast Kalman' algorithm is derived by Falconer [23]. This method achieves a complexity that is proportional to N by making use of the fact that successive signal vectors differ by only a shift in the signal samples. A generalization is made to equalizers in which p new samples are input and p old samples are removed at every symbol interval. In this algorithm it is necessary to invert and multiply $p \times p$ matrices, resulting in a complexity of $7Np + Np^2 + 3p^2 + 2N + 4/3 p^3 - p/3$ multiplications per symbol interval [23]. (Divisions are counted as multiplications). For the case of a fractionally spaced DFE, $p=3$, and the number of multiplications becomes $32N + 52$. Unfortunately, while the fast Kalman algorithm is very computationally efficient, it is also the most numerically unstable. This problem has been tackled by the use of restarting procedures in [11].

An algorithm that is more stable for single precision floating point arithmetic is derived by Hsu [24]. This is known as the square-root

Kalman algorithm because it recursively updates factors of R_n^{-1} rather than the matrix itself. In this way, the numerical ranges of the variables are reduced, making the algorithm less sensitive to roundoff errors. Additionally, the complexity is about $1.5N^2 + 6.5N$ operations per iteration [25]. Although this is about half that of the Kalman algorithm, it is still proportional to N^2 .

4.1.3. Lattice Filters

The adaptive algorithms discussed so far have assumed a transversal filter structure. Other algorithms, however, are based on the lattice filter structure shown in Figures 4.2 and 4.3 [13]. In this filter, the $N-1$ stages implement forward and backward predictors of order 1 to $N-1$. The outputs of the m 'th stage, $f_m(n)$ and $b_m(n)$, are the m 'th order forward and backward predictor errors, respectively. An important property of the backward predictor errors of different orders is that they are uncorrelated. As can be seen from the figures, these backward predictor errors are weighted and summed to form the equalized signal.

The recursive least squares lattice equalizer minimizes the same performance criterion as (4.24), but in this case the error is obtained differently. Part of the least squares lattice algorithm involves updating the coefficients in the lattice stages, $G_m(n-1)$ and $H_m(n-1)$ as shown in Figure 4.2. These are selected to minimize the following sums [5]:

$$\sum_{k=0}^n \epsilon_m^{n-k} f_m^2(k, n) \quad (4.38)$$

Figure 4.2
Recursive Least Squares Lattice Equalizer [13]

Figure 4.3
Gradient Lattice Equalizer [13]

$$\sum_{k=0}^n \lambda^{n-k} b_m^2(k, n) . \quad (4.39)$$

The notation $f_m(k, n)$ indicates the error at time k , formed using the coefficients of time n . Since the same performance criterion is minimized, the RLS lattice has the same convergence as the RLS algorithm discussed previously.

The lattice equalizer in Figure 4.3 is adapted using the gradient lattice algorithm. This algorithm is less complex because it has fewer adjustable parameters, but it has a slower convergence. In updating the lattice stages, the criterion that is minimized is [5]:

$$\sum_{k=0}^n \lambda^{n-k} [f_m^2(k) + b_m^2(k)] \quad (4.40)$$

where the predictor errors are defined by the relations

$$f_m(n) = f_{m-1}(n) - k_m(n) b_m(n-1) \quad (4.41)$$

$$b_m(n) = b_{m-1}(n) - k_m(n) f_{m-1}(n) . \quad (4.42)$$

This differs from the RLS lattice in that the two coefficients of the m 'th lattice stage are assumed to be identical. Also, the coefficients which weight the backward predictor errors to form the equalizer output are adapted by a stochastic gradient algorithm. Note, however, that once the lattice stages converge, the backward predictor errors are orthogonal. Thus the gradient lattice does not suffer the same degradation as the transversal stochastic gradient algorithm in the presence of a large eigenvalue spread. A comparison of the convergence

of the least squares, gradient lattice, and stochastic gradient algorithm is shown in Figure 4.4 [5].

The lattice algorithms have a complexity that is proportional to N , but the constant of proportionality is quite high. This is especially true for the multichannel algorithms that process p dimensional vectors so that p samples can be shifted at a time. Such algorithms are necessary for the implementation of a fractionally spaced DFE. Another problem is the requirement that the number of stages for each channel be the same. Effectively, this means that for a $T/2$ spaced forward filter, the number of feedback coefficients must equal half the number of forward filter coefficients. While a general structure for a lattice DFE is presented in [25], it does not have a fractionally spaced forward section.

The greatest advantage of lattice filters is their numerical stability. In [25] it is reported that lattice DFE's did not become unstable, even when 8 bit fixed point arithmetic was used.

4.1.4 Problems for Adaptation on the HF Channel

As previously mentioned, the HF channel is both dispersive and time varying. The possibility of taking advantage of the inherent diversity depends on the tracking ability of an adaptive equalizer. The performance of LMS and RLS adapted DFE's has been compared on simulated HF channels in [2], [21], and [25]. A general conclusion is that the LMS DFE is less effective in tracking the time variations that are typical of this channel. This poor performance is due to the large eigenvalue

Figure 4.4
Convergence of Different Adaptive Algorithms [5]

spreads often encountered on HF channels [25].

The performance of the exponentially weighted RLS algorithm is sensitive to the value of λ , the optimal value being dependent on the rate of channel variation [2]. The effect of λ can be interpreted as follows. For a constant channel, a value of λ that is closer to one gives a longer memory and thus more time for averaging. The result is a better estimate of the optimum coefficients. For a time varying channel, however, variation of the optimum coefficients will degrade the estimate so the memory must be reduced. Thus there is a tradeoff between the time needed for averaging and the amount of variation that can be tolerated. As the rate of channel variation increases, a fundamental limitation is reached where the memory is too short to provide a good coefficient estimate. At this point even the fastest algorithm is incapable of tracking the channel.

An investigation of algorithm performance for different channel variation rates is reported in [2]. The performance curves for a symbol spaced DFE with nine coefficients are shown in Figure 4.5. In this figure, the rate of variation is indicated by the 3 dB frequency of the filter used in generating the tap gains of an HF channel model. First of all, the very poor performance of LMS relative to the RLS algorithm can be observed. Also, for RLS, the performance degrades sharply for rates above 0.5 Hz, indicating the limit of the adaptation speed in this example.

Figure 4.5
Performance of Algorithms for Different
Channel Variation Rates [2]

In the simulations discussed above, known symbols are used at the receiver so that the problems resulting from an incorrect decision are avoided. In a realistic situation, however, decision direction is used to track changes in the optimum equalizer coefficients. In this case, an incorrect decision has several effects. First, there is the direct effect in the DFE of subtracting the wrong quantity from the forward filter output. Additionally, the wrong value for the error signal is input to the adaptive algorithm. Finally, since the signal vector is also used in the adaptive algorithm, an incorrect decision in the feedback filter will affect adaptation for the next N_b symbol intervals. In a fast varying channel, a relatively large weight must be applied to each new update of the coefficient estimate. Thus a decision error can cause the coefficients to deviate significantly from their correct trajectory. Additionally, during a fade the probability of making an error is quite high, causing the adaptive equalizer to completely lose track of the optimum coefficient vector. This has been referred to as an equalizer crash.

Following a crash, it has been found that the coefficients do not converge under decision directed mode [21]. Therefore it is necessary to include known symbols in the data stream. A strategy described in [21] involves the use of a full duplex link over which the receiver can request a training sequence when the output SNR falls below a certain level. The performance of this scheme is quite close to that of the ideal reference mode, because training is provided exactly when it is needed.

If such a full duplex link is not available, a training sequence can be transmitted periodically. In this case the sequence may not be sent when it is needed. However, if a crash occurs, then the equalizer will lose track of the channel only until the arrival of the next training sequence. Also, periodic retraining helps keep the coefficients from deviating too far from their proper values, thus reducing the probability of a crash. As part of the work for this thesis, a square root Kalman algorithm was simulated for a fractionally spaced DFE. The details of the simulation are discussed in the next chapter. Some of the results comparing the ideal reference mode to the decision directed mode for two different rates of channel variation are shown in Figure 4.6. For the 0.5 Hz fade rate, a block of 65 training symbols is transmitted for every 200 data symbols, giving a training overhead of 25%. For the 0.15 Hz fade rate, the 65 symbol block was sent for every 300 data symbols, or 18% training. The very serious degradation under decision direction can be observed. This agrees with the results in [2]. Because of this degradation, the rate of variation for which an adaptive algorithm can be used is limited even further.

4.2 Block Algorithms

The requirement of periodic training sequences on the HF channel and the poor performance under decision direction suggest the use of a block by block approach. In this method the entire training sequence is used in an algorithm to compute the set of equalizer coefficients. In doing this, it is desirable to obtain an accurate estimate with a training block that is as short as possible. This reduces overhead and

Figure 4.6

Performance with Periodic Retraining

allows less channel variation within the block.

In the following discussion, it is assumed that a block of M data symbols are known: $d_1, d_2, \dots, d_m$. Data symbols outside this block are unknown. Additionally, it is assumed that $T/2$ spaced samples of the received signal, $r(nT - T/2), r(nT), \dots$, appear at the receiver. Whether or not signal samples from outside the block are used depends on the block algorithm. Three algorithms will be discussed for the DFE: the autocorrelation method, the prewindowed method, and the covariance method. In each case a performance function of the squared error is minimized, resulting in a system of equations to be solved for the equalizer coefficients.

The autocorrelation method minimizes the sum

$$\sum_{n=-\infty}^{\infty} |w_n d_n - \underline{c}^* \underline{x}_n|^2 \quad (4.43)$$

where

$$\underline{x}_n^T = [w_n r(nT - T/2) \ w_{n-1} r(nT - T) \ \dots \ w_{n-N_f/2} r(n - N_f/2) \ T - T/2] \\ w_{n-1} d_{n-1} \ \dots \ w_{n-N_b} d_{n-N_b}] \quad (4.44)$$

The sequence w_n is a window function that is zero for $n < 1$ and $n > M$. This is applied to the data symbols and the received samples so that the summation in (4.43) can be taken to infinity. In this way, the mean squared error for the windowed data and signal is minimized, where the mean is obtained by time averaging rather than ensemble averaging. Note that when n is slightly greater than 1 or slightly less than M , the

vector $\underline{x}_n$ contains nonzero elements and zero elements. Thus a value for the error is formed by assuming that some of the data symbols and received samples are zero. The effect of this is mitigated by using a window that approaches zero gradually at each end of the block. However, the windowing assumption still decreases the accuracy of the coefficient estimate.

The autocorrelation method has an advantage when a T-spaced linear filter is used, because in this case the resulting autocorrelation matrix is Toeplitz [17]. This leads to a computationally efficient solution for the coefficients. However, for a decision feedback equalizer this matrix is not Toeplitz. Thus the autocorrelation method offers no advantage in computing the coefficients of a fractionally spaced DFE.

The second algorithm to be discussed is the prewindowed method. In this method a windowing assumption is made at the beginning of the block, but not at the end. This can be seen from the sum that is minimized:

$$\sum_{n=1}^M w_n^2 |d_n - \underline{c}^{*T} \underline{y}_n|^2 \quad (4.45)$$

where $\underline{y}_n$ is the same vector as in (3.16), and w_n is a weighting function. Since the summation is only taken up to time M, no data or signal samples past the end of the block are included. At the beginning of the block, however, the vector $\underline{y}_n$ contains data and signal samples for $n < 1$. In the prewindowed method these elements are assumed to be zero, so that errors at the beginning of the block are computed with some zero samples. The function w_n is used to apply less weight to these errors. By comparing (4.45) and (4.24), it can be seen that RLS algorithms use prewindowing

with $w_n = \delta^{M-n}$.

The covariance method is similar to the prewindowed method, but in this case no windowing is assumed at all. However, because only an M symbol training sequence is available, only data from within the training block can be used. This leads to the performance function

$$J = \sum_{n=N_b+1}^M d_n - \underline{c}^{*T} \underline{y}_n \quad (4.46)$$

where the summation starts at N_b+1 because $\underline{y}_n$ contains the data symbols d_{n-1} to d_{n-N_b} . Note that, if $N_b < (N_f/2) - 1$, then at the beginning of the block the signal samples $r(-((N_f/2) - N_b)T + 3T/2)$ to $r(0)$ are used. These samples must be known because in the covariance method they are not assumed to be zero.

Although the covariance method may require knowledge of signal samples outside the block, the lack of any windowing makes it the most accurate. Thus, the covariance method is used for the investigation of block by block equalization in this thesis.

Minimization of (4.46) with respect to $\underline{c}$ gives

$$\underline{c} = \hat{R}^{-1} \hat{p} \quad (4.47)$$

where in this case

$$\hat{R} = \sum_{n=N_b+1}^M \underline{y}_n \underline{y}_n^{*T} \quad (4.48)$$

$$\hat{p} = \sum_{n=N_b+1}^M d_n \underline{y}_n \quad (4.49)$$

If the training block is long enough, R is positive definite so that Choleski's method may be used to solve the system of equations in (4.47) [26].

The computation involved in forming the sums in (4.48) and (4.49) and then solving the system of equations is

$$\left(\frac{N}{2}(N+1) + N\right) + \frac{N^3 + 9N^2 + 2N}{6} \quad (4.50)$$

multiplications and divisions, where $c=M-N_b$ is the number of terms in the summation. The computed coefficients are used for the corresponding data block. Thus the computational load per symbol interval can be found by dividing (4.50) by the sum of the training block length and the data block length.

An alternative method for minimizing (4.46) is derived by Cioffi [27]. This Block Fast Transversal Filter (BFTF) algorithm requires a computation that increases with N^2 instead of N^3 . For a fractionally spaced DFE, the multichannel version of this algorithm must be used. In this version the total number of coefficients is $M=N'p$, where p is the number of 'channels' and N' is the dimension of each channel. In the case of a DFE with a $T/2$ spaced forward filter, $p=3$. Assuming Gaussian elimination is used to invert $p \times p$ matrices, the computational requirement is

$$\begin{aligned} & N'(p^2+p) + \frac{N'}{2} (15p^3+26p^2) + \frac{N'^2}{2} (3p^3+8p^2) - (3p^3+3p^2+p) \\ &= N(p+1) + \frac{N}{2} (15p^2+26p) + \frac{N^2}{2} (3p+8) - (3p^3+3p^2+p) \end{aligned} \quad (4.51)$$

which for $p=3$ becomes

$$41N + 106N + 8N^2 - 111$$

(4.52)

Thus, if the number of feedback coefficients can be constrained to be equal to half the number of forward filter coefficients, and N is large enough, the BFTF algorithm offers a computationally efficient method for solution of the equalizer coefficient vector.

CHAPTER 5

CHANNEL SIMULATION

To investigate the feasibility of block by block equalization of the HF channel, a simulation program has been written. In this simulation, a complex baseband equivalent model is used to describe the transmitted signal, the fading multipath channel, the additive noise, and the equalizer. A block diagram is shown in Figure 5.1.

A simulation cycle starts with the input of a data symbol, d_n . These symbols are independent and take on values of $\exp(j\pi/4)$, $\exp(j3\pi/4)$, $\exp(-j3\pi/4)$, and $\exp(-j\pi/4)$ with equal probability. This is a representation of QPSK modulation, and a symbol rate of 1200 symbols/second is assumed.

In order to simulate the transmission of a pulse through a channel, the data value is made into an impulse followed by zeros for the remainder of the symbol interval. This impulse is then sent through a pulse shaping lowpass filter, as shown in Figure 5.2. The output of this filter is thus a sequence representing samples of the transmitted signal. As seen in the figure, 4 samples per symbol interval are used, giving a sampling rate of 4800 Hz.

Figure 5.1
Simulation Block Diagram

Figure 5.2
Pulse Shaping Filter

The desired pulse has a raised cosine frequency response. The cutoff frequency is 600 Hz and the rolloff factor is one, giving a total baseband bandwidth of 1200 Hz. To simulate this, the FIR filter coefficients are set to the samples of the corresponding delayed impulse response.

The output samples of the pulse shaping filter are sent through the multipath model. As illustrated in Figure 5.3(a), this is a tapped delay line with taps corresponding to resolvable paths. Each tap gain is a complex Gaussian random variable with zero mean, producing Rayleigh fading on each path. The multipath structure used in most of the simulations has three taps with delays of 0 , $2T$ and $4T$, and each tap gain has a variance of one. Also, the different paths fade independently.

A complex tap gain is produced by passing complex Gaussian white noise through a second order IIR Butterworth filter, as shown in Figure 5.3(b). The simulation program accepts as input the 3 dB frequency of the filter, and this is used to calculate the corresponding coefficients. The details of this are in Appendix A. The filters for each tap gain are identical, so the 3 dB frequency controls the rate of channel variation. This is termed the fade rate in this thesis, and corresponds to half the frequency spread defined in Chapter 2.

It was mentioned in Chapter 2 that for the HF channel, a good model for the tap gain power spectrum is a Gaussian function of frequency. In the simulation this is approximated by the squared magnitude response of a second order filter. A second order filter has also been used in [2], [21] and [25]. At this point some comments can be made about the effect

Figure 5.3a)
Multipath Model and Additive Noise

Figure 5.3b)
Tap Gain Generation

of the filter used to generate the tap gains. In early simulations of this work, a filter with a first order rolloff was used. This led to rapid tap gain variation and very poor performance, even when the fade rate was made relatively small. The poor performance for this type of filter has also been observed in [2] and [11]. Some insight into this behavior can be gained from a comparison of the autocorrelation functions corresponding to different tap gain spectrums. Figure 5.4 shows autocorrelations for a Gaussian, first order, and second order spectrum. A noticeable feature is the discontinuous derivative at zero for the first order filter. This indicates that even for a large coherence time, the tap gain generated from a first order filter becomes uncorrelated with itself very quickly. Also, it can be seen that while a first order filter gives a poor approximation to the Gaussian spectrum, a second order filter does quite well.

At the output of the multipath model, noise is added as shown in Figure 5.3. The noise is complex Gaussian with zero mean, and it is filtered in order to introduce some correlation. The noise filter has the same structure as the pulse shaping filter. The frequency response is a raised cosine with a cutoff frequency of 1200 Hz and a rolloff that is calculated to give a total baseband bandwidth of 1500 Hz. This is done to simulate a typical 3 KHz passband channel using the complex baseband equivalent model. Again, the sampling rate in the simulation is 4800 Hz. Also, the variance of the additive noise is scaled to achieve a desired SNR. The computation of the noise variance is discussed in Appendix B.

Figure 5.4
Tap Gain Autocorrelation Functions
for a Fade Rate of One

Finally, the output of the channel is sampled at $T/2$ second intervals for input to the equalizer. Thus, in every symbol interval, a single data value is input to the channel simulator and two $T/2$ spaced samples are output. The subroutines for initialization and channel simulation are listed in Appendix C.

CHAPTER 6

SIMULATION OF BLOCK BY BLOCK EQUALIZATION

As an alternative to symbol by symbol estimation of equalizer coefficients, a block by block approach is investigated. As mentioned previously, a block by block approach assumes that a known training sequence precedes each data block. Using the symbols in the training block, the equalizer coefficients are estimated using the covariance method, and these are used to equalize the channel. This alternative is considered for several reasons. In symbol by symbol algorithms, the occurrence of equalizer crashes necessitates the transmission of periodic training sequences. Also, the resulting performance under decision direction is still significantly degraded from that of the ideal reference mode. This is mainly because of the effect of decision errors on the adaptation algorithm [2]. On the other hand, the block by block approach uses only known symbols in estimating the coefficients. Thus, for a given overhead, the performance of a symbol by-symbol algorithm can be compared to that of the block by block method.

Another reason for looking at the block by block approach is a possible reduction in the computational complexity. This is because the computation of equalizer coefficients takes place once per block. Therefore, if the data block is long enough, the effective computation per symbol interval may be much less than that of a symbol by symbol algorithm. Also, the ability to distribute the computation over an entire block may ease implementation.

The block by block approach is investigated by simulating a decision feedback equalizer. The length of the feedback filter is chosen to span the expected multipath spread, while the $T/2$ spaced forward filter is slightly longer. Simulations are done to determine the lengths of the training and data blocks that give the best performance for a given fade rate. Another factor to be considered is overhead, which is the percentage of transmitted symbols that are used for training. To increase the information rate, overhead should be minimized.

The criterion used to evaluate the block lengths is the MSE, defined in Chapter 3. However, instead of presenting the results in terms of the MSE, an output signal to noise ratio is defined:

$$\text{output SNR} = 10 \log_{10} \left(\frac{1}{\text{MSE}} \right) \quad (6.1)$$

This can be considered as the signal to noise ratio of the output of the equalizer since the transmitted symbols have unit amplitude. If the theoretically optimum coefficients are used, then the minimum MSE results. This also has a corresponding output SNR, denoted SNR_{opt} . Since the actual MSE depends on the MMSE, performance results are indicated as a degradation in output SNR from SNR_{opt} .

Section 6.1 investigates the degradation in output SNR from the SNR_{opt} at the end of the training block. The degradation results from imperfect estimation of coefficients, and simulations are used to determine this as a function of training block length and fade rate. Next, Section 6.2 considers the situation in which the optimum coefficients

are known at some point in time. At this point, then, the output SNR is equal to SNR_{opt} . If the equalizer coefficients are kept fixed at these values, however, then the output SNR will fall below SNR_{opt} because of variation in the channel impulse response. Thus, by simulating the equalizer and observing the output SNR, possible values for the data block length are obtained. Finally, Section 6.3 investigates the use of coefficients that are linearly interpolated between the values at each end of the data block. The optimum coefficients at each end of the block are used, and the worst output SNR in the data block is observed for different block lengths. Also in Section 6.3, training blocks and data blocks are simulated together in determining probability of error curves for different data block lengths and fade rates.

6.1 Training Block Length

In the case of a constant channel, the coefficient estimate improves as the training block length increases. However, when the optimum coefficients vary with time, a longer block allows more variation to take place while the coefficients are being estimated. This leads to two opposing requirements for the training block length. That is, the block must be long enough to estimate the coefficients, but short enough to avoid significant variation. As the rate of variation increases, a point is reached where both conditions cannot be satisfied simultaneously, indicating the limitations of the block algorithm in estimating the coefficients for a time varying channel.

Consider the training block-data block combination in Figure 6.1. Say that the training block has just been used to compute the coefficients. These are now to be used to equalize the channel starting at time n , the beginning of the data block. Corresponding to the channel impulse response at this time, there exists a MMSE, as discussed in Chapter 3. This is the MSE that would result if the optimum coefficients were used. Note that the optimum coefficients are not available in a practical system that does not know the channel impulse response. Given the actual equalizer coefficients, there also exists a MSE corresponding to the channel impulse response at that time. The MSE will be greater than the MMSE because of imperfect estimation of coefficients. Equivalently, the output SNR at the beginning of the data block will be lower than the SNR_{opt} at that time. Note that this initial degradation from SNR_{opt} is probably the smallest degradation that will be attained in the data block. This is because the coefficient estimate will most likely get worse as the channel impulse response varies over the data block.

For a given fade rate, it is desired to find a training block length that minimizes this initial degradation. However, actual values of MMSE and MSE depend on the particular channel impulse response. To find a measure of degradation that depends only on fade rate and training block length, $\langle \text{MMSE} \rangle$ and $\langle \text{MSE} \rangle$ are defined to be averages over many channel impulse responses. That is, simulations are run in which 60 training blocks of a specific length are sent through a channel with a given fade rate. For each block, the actual channel impulse response at the end of the block (which would not be known to a practical system) is used in equation (3.22). This gives the MMSE at the end of that

Figure 6.1
Training Block and Data Block

training block. Also, the training block is used to estimate the equalizer coefficients, and (3.23) is used to calculate the MSE at the end of that training block. The channel is allowed to vary randomly between successive blocks, and the values of MMSE and MSE are averaged to get $\langle \text{MMSE} \rangle$ and $\langle \text{MSE} \rangle$, respectively. Using these averages, a degradation in output SNR at the beginning of the data block is defined as:

$$\Delta \text{SNR} = 10 \log_{10} \frac{\langle \text{MSE} \rangle}{\langle \text{MMSE} \rangle} \quad (6.2)$$

This indicates the degradation, due to using estimated coefficients, for a specific training block length and fade rate.

Figure 6.2 shows the results of SNR at the beginning of the data block versus training block length for different fade rates. For this equalizer and a channel SNR of 15 dB, the average SNR_{opt} is about 12 dB. For a fixed channel and for low fade rates, a long block can produce a small degradation. In order to minimize overhead, however, some SNR must be tolerated. If, say a ΔSNR of 2 dB is allowed, then a training block length of 65 symbols is sufficient for fade rates up to 0.5 Hz. For higher fade rates, however, significant variation occurs within 65 symbols and a degradation of less than 2 dB cannot be attained.

Figure 6.3 shows the results for a fixed channel and different numbers of equalizer coefficients. In the figures, the notation DFE(n,m) is used to describe the equalizer. For example, DFE(10,4) denotes an equalizer with 10 $T/2$ spaced forward filter taps and 4 feedback filter taps. In the channel simulations, the multipath model always has three paths and each path gain has a variance of one. However, the multipath

Figure 6.2
 Δsnr at Beginning of Data
Block vs. Training Block Length For Different
Fade Rates, DFE(10,4)

Figure 6.3
 Δsnr at Beginning of Data
Block vs. Training Block Length For Different
Numbers of Equalizer Coefficients, Fixed Channel

spreads are chosen to accomodate the different equalizer lengths.

Table 2 shows the tap delays in the multipath model that are used for the equalizers in Figure 6.3.

Table 2

Equalizer Structure and Corresponding Multipath Tap Delays

DFE (6,3)	0, T, 2T
DFE (10,4)	0, 2T, 4T
DFE (14,6)	0, 3T, 6T

As expected, an increased number of coefficients requires a longer training block to obtain the same Δ SNR. When the optimum coefficients vary with time, more variation can occur within a longer block, causing greater degradation for the same fade rate. Thus, as the number of equalizer coefficients is increased, a block algorithm for estimation of the coefficients is limited to lower fade rates.

In the above simulations, the signal to noise ratio in the channel is 15 dB. However, other simulations were run with channel SNR's of 5 dB and 25 dB, with very little difference in the results.

6.2 Data Block Length

Once the coefficients have been estimated from the training block, they can be used to equalize the channel over the data block. In a time varying channel, the optimum coefficients will vary from the values they had at the beginning of the block. This leads to an excess mean

squared error that increases during the length of the block. Thus, the data block must be short enough to keep this excess mean squared error to within a tolerable level. However, to minimize overhead, it is desirable to have a data block that is as long as possible.

The mean squared error at the equalizer output, or equivalently the output SNR, is also a function of the signal to noise ratio in the channel. In the simulations, however, it is observed that the signal power in the channel remained practically constant over the data block lengths of interest. Due to the diversity effect, this is especially true for a multipath model with three paths, whereas a one path channel showed slightly faster variation of signal power. Because of this observation, it is assumed that the drop in output SNR over the data block is due mostly to the deviation of the optimum coefficients from their values at the beginning of the data block.

To determine the possible lengths for the data block as a function of fade rate, simulations are run in which the optimum coefficients are calculated periodically from the actual channel impulse response. The routine for doing this is listed in Appendix D. These coefficients are used to equalize the channel for the symbols that follow, and receiver decisions are used in the feedback filter of the DFE. The output SNR is calculated for each symbol in the block by using equation (6.1). The value for MSE in this equation is estimated by averaging the squared magnitudes of the errors for the past 20 symbols. An 'equalization interval' is defined as the number of symbols before the output SNR drops 3 dB below its initial value.

It was mentioned that receiver decisions $\bar{a}$ are used in the DFE. To see if this significantly affects the 'equalization interval', some simulations are run in which the actual transmitted symbols are used in the feedback filter. A two path multipath model and a DFE(8,4) are used, and the equalization intervals of 30 simulations are averaged. Also, the fade rate is 0.5 Hz and the channel SNR is 15 dB. In this case, the results are no different from those in which receiver decisions are used. Presumably, this is because error propagation does not occur until the output SNR has already dropped below a certain level.

In addition to using constant equalizer coefficients, an LMS algorithm with a normalized step size is used. The algorithm is periodically initialized with the optimum coefficients, and used in decision directed mode to try to track the changes in the optimum coefficients. This is done in an attempt to lengthen the equalization interval, allowing a longer data block and less overhead.

As described in Chapter 4, the normalized step size in the LMS algorithm makes use of the constants w and α . A w that is too small gives a noisy estimate of signal power, whereas a w that is too close to one results in slow tracking of changes in signal power. In addition, if α is too small the LMS algorithm will be too slow in tracking the equalizer coefficients. However, a larger α increases estimation noise and may cause instability. To determine α and w , some preliminary simulations were done at a fade rate of 0.3 Hz. Average equalization intervals were measured using different values of α and w , and it was found that $\alpha=0.2$ and $w=0.9$ gave the best results.

The simulations use the 3 path multipath model with delays of 0, $2T$ and $4T$ and the DFE with 10 forward filter and 4 feedback filter coefficients. Also, the channel SNR is 15 dB. For a given fade rate, a simulation is run in which the optimum coefficients are calculated 50 times, at intervals that depend on the fade rate. After each optimum coefficient calculation, the equalization interval is measured using constant coefficients and also using LMS. The 50 intervals for each method are then averaged. Figure 6.4 shows average equalization interval vs. fade rate for constant coefficients and for LMS. Fade rates of 0.2, 0.3 and 0.5 Hz are used. As expected, the average interval decreases with the fade rate. Also, the LMS algorithm, restarted with optimum coefficients, increases the average interval by about 50%. This improvement is due to the ability of the LMS algorithm, in some cases, to track variations in the optimum coefficients. However, note that even for the LMS algorithm, the indicated data block lengths result in an overhead of 25 to 38% when a training block of 65 symbols is used. Finally, in other simulations it was noticed that while the average equalization interval did not depend on the number of equalizer coefficients, it tended to decrease when more paths were used in the multipath model. This seems intuitively correct since a greater number of paths should increase the chance of optimum coefficient variation within a certain time.

Note that the optimum coefficients are used at the beginning of the interval. Thus, the output SNR in the equalization interval does not include the initial degradation that results from starting with estimated coefficients. Because of this, the above results focus on the effect of

Figure 6.4
Average Equalization Interval vs. Fade Rate

channel variation during the data block. If the initial degradation is included, the output SNR during the equalization interval will be that much lower. In this case a drop of 3 dB may not be tolerable. Thus, the above results are an upper bound on the possible data block lengths.

It was mentioned that the measured equalization intervals indicate the requirement of a large overhead. Moreover, only the average intervals were reported. For the average equalization interval to be a useful indication of possible data block lengths, most of the measured intervals should be close to the average. However, in the simulation it is found that this is not the case. In fact, a very large range of intervals is observed. Figure 6.5 shows equalization interval distributions for constant coefficients and LMS at two different fade rates. In all of the distributions there is a relatively large number of very short intervals. A short interval indicates a rapid variation of the optimum equalizer coefficients, giving a large drop in output SNR in a short period of time. Also, although the distribution for the LMS algorithm shows an increased number of long intervals, a significant number of short intervals remains. In effect, the LMS algorithm is effective in intervals in which the coefficients vary slowly, but it cannot do anything when a fast change occurs that results in a short interval. It can also be seen that this behaviour occurs for fade rates of 0.2 and 0.5 Hz, indicating that relatively fast variation of equalizer coefficients can occur even when the channel is varying more slowly.

Figure 6.5a)
Equalization interval Distributions. Fade Rate = 0.2

Figure 6.5b)
Equalization Interval Distributions. Fade Rate = 0.5

2

OF/DE

2

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

Thus, the value of the average equalization interval is misleading, for there is a large percentage of intervals that are much smaller than this. If the average is used as a data block length, then many blocks will be lost because of fast coefficient variation. However, from Figure 6.5 it can be seen that very short data blocks would have to be used to avoid this problem. When the training block is used this would result in a very large overhead.

In order to investigate this behaviour further, the values of optimum equalizer coefficients are computed every few symbol intervals. The multipath model has two paths with delays of 0 and T, and the fade rate is 0.5 Hz. Also, the equalizer has 4 forward filter coefficients and one feedback coefficient. These small numbers of coefficients are used to reduce the computation load. The real part of the main equalizer coefficient is shown in Figure 6.6, in which an abrupt change is noticed at about 1300 symbols. An equalization interval that begins during such a fast change would be quite short. Thus, the occurrence of fast changes in the optimum coefficients explains the presence of the short intervals.

The presence of abrupt changes is due to the nature of the solution for the equalizer coefficients. To see this, consider the simple example of the transmission of T-spaced impulses through a two path channel with impulse response:

$$h(nT) = a\delta(nT) + b\delta(nT-T) \quad (6.3)$$

where a and b are real. The T-spaced samples of the received signal are then given by:

Figure 6.6
Real Part of Equalizer Coefficient, 2 path
Channel, Fade Rate = 0.5Hz

$$r(nT) = \sum_k d_k h(nT-kT) + n(nT) \quad (6.4)$$

where $n(nT)$ is assumed to be white noise with a variance of σ_n^2 . Finally, assume a DFE with one feedback tap and 2 T-spaced forward filter taps. The optimum coefficient vector is given by

$$\underline{c}_{\text{opt}} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \frac{1}{(a^2 + \sigma_n^2)(a^2 + b^2 + \sigma_n^2) - a^2 b^2} \begin{bmatrix} b\sigma_n^2 \\ a(a^2 + \sigma_n^2) \\ -ab(a^2 + \sigma_n^2) \end{bmatrix} \quad (6.5)$$

and when both a and b are large compared to the noise, this is approximately $[0, 1/a, -b/a]$.

Now say that b remains large but a decreases, so that $c_2 = 1/a$ increases. However, when a is much smaller than σ_n , $\underline{c}_{\text{opt}}$ is approximately $[1/b, 0, -a/b]$. Note that to achieve this, a rapid change must occur in c_2 . This is verified by plotting c_2 as a function of a in Figure 6.7. Thus, when a becomes close to zero, even small changes in the channel can result in a large change in the equalizer coefficients. The amount that a varies depends on the rate of channel variation, so that the time rate of change of the equalizer coefficients still depends on the fade rate. The measurements of block length, however, show that for fade rates of 0.2 to 0.5 Hz, the changes are rapid enough and frequent enough to significantly impair block by block equalization.

Figure 6.7
Variation of c_2 with a .
 $b=1.0, \sigma_n^2=0.01$

6.3 Block By Block With Interpolation

In the previous section we have reported simulated behaviour based on ideal optimum equalizer coefficients (that would not be available in practice) to get some feel for data block lengths appropriate for a fading channel environment. To overcome the problem encountered in the use of constant coefficients or LMS over the data block, some way of tracking fast changes in the optimum equalizer coefficients is required. The previous section contemplated a system in which only the optimum coefficients at the beginning of the data block were used. If the optimum coefficients are changing rapidly, however, we have seen that this information becomes obsolete very quickly. What is needed is a knowledge of the trend in the optimum coefficient variation. This can be achieved at the expense of a delay in the receiver's decisions.

Consider the training and data blocks in Figure 6.8, where the estimated coefficients at the beginning and end of the data block are $\underline{c}_{t1}$ and $\underline{c}_{t2}$. Assume that the received signal samples of the data block are stored until the training sequences at each end of the block are used to calculate $\underline{c}_{t1}$ and $\underline{c}_{t2}$. Then, the signal samples are passed through the equalizer and the coefficients for each symbol interval are found by linearly interpolating between $\underline{c}_{t1}$ and $\underline{c}_{t2}$. Say the symbol intervals in the data block are numbered from 1 to B, where B is the data block length. To linearly interpolate between $\underline{c}_{t1}$ and $\underline{c}_{t2}$, the coefficients are updated at each symbol according to:

Figure 6.8
Training Blocks Used in Interpolation

$$\underline{c}(n+1) = \underline{c}(n) + \frac{(\underline{c}_{t2} - \underline{c}_{t1})}{B}, \quad n=0, \dots, B-1, \quad (6.6)$$

where $\underline{c}(0)$ is given the value of $\underline{c}_{t1}$. This update is done before the computation of each equalizer output.

Using the above method, the value of an interpolated coefficient varies linearly over the length of the data block. However, it is most likely that the trajectory of an optimum coefficient will not be a straight line, and this will result in an excess mean squared error. Nevertheless, the shorter the data block, the more closely the straight line will approximate the trajectory. Thus, the data block must be short enough to keep this excess mean squared error to within a tolerable level.

Some simulations were run in which the optimum coefficients were computed at each end of the block, and linear interpolation was used to update the coefficients. Again, the motive was to investigate the limits of performance of such a system. Using the same procedure as in the previous section, the output SNR was computed over the length of the block. Also as in the previous section, it was found that the signal power was practically constant for the block lengths of interest. It was noted that the output SNR was high at the ends of the block where the optimum coefficients were used, and that it often dipped at some point in the middle of the block. Additionally, it was noticed that a longer block resulted in a larger dip in the output SNR. This observation led to a measure of degradation as a function of the block length. This is the difference in output SNR, in dB, between the lowest SNR in the block and the SNR at one of the ends of the block. The end

with the lowest SNR is chosen in order to reduce the effects of variation in the minimum mean squared error. This measure of 'worst SNR difference' is used to estimate the range of possible block lengths, and to provide a comparison with the use of constant coefficients.

In addition to interpolation of coefficients over the block, a couple of variations of this approach are investigated. These involve augmenting the interpolation with an LMS algorithm. The idea behind this is that linear interpolation provides slope information about the variation of the optimum coefficients. Thus, no matter how fast the optimum coefficients are actually varying, the LMS algorithm need only track the difference between the optimum coefficient trajectory and the straight line approximation. Thus, it may be possible to extend the block length with only a small increase in computation. In applying the LMS algorithm, a normalized step size and decision direction are used.

The first approach assumes that the coefficient vector is the sum of two terms:

$$\underline{c}(n) = \underline{c}_i(n) + \underline{c}_\Delta(n) \quad (6.7)$$

where $\underline{c}_i(n)$ results from interpolation and $\underline{c}_\Delta(n)$ results from an LMS algorithm. The term $\underline{c}_\Delta(n)$ is updated as discussed above:

$$\underline{c}_\Delta(n+1) = \underline{c}_\Delta(n) + \frac{(\underline{c}_{t2} - \underline{c}_{t1})}{B} \quad (6.8)$$

The algorithm for $\underline{c}_\Delta(n)$ is derived by finding the gradient of the mean squared error with respect to $\underline{c}_\Delta(n)$, and using an estimate of the gradient as described in Chapter 4. This results in:

$$\underline{c}_2(n+1) = \underline{c}_2(n) + \Delta y_n e_n^* \quad (6.9)$$

where $e_n = d_n - \underline{c}^*(n)^T \underline{y}_n$. The update for the actual coefficient vector is found by adding (6.8) and (6.9) to get:

$$\underline{c}(n+1) = \underline{c}(n) + \frac{(\underline{c}_{t2} - \underline{c}_{t1})}{B} + \Delta y_n e_n^* \quad (6.10)$$

This algorithm is termed LMS-aided interpolation.

Because of estimation noise and lag, the coefficients calculated using LMS-aided interpolation will likely deviate from $\underline{c}_{t2}$ by the time the end of the block is reached. Thus, full advantage is not taken of the fact that these coefficients have already been estimated. To overcome this problem, another approach is used in augmenting interpolation with LMS. In this approach, the coefficients of the first symbol are calculated as in LMS-aided interpolation:

$$\underline{c}(1) = \underline{c}(0) + \frac{\underline{c}_{t2} - \underline{c}_{t1}}{B} + \Delta e_0^* \underline{y}_0 \quad (6.11)$$

However, at the second symbol, the interpolation is recalculated as if the block now starts at this symbol. Thus, instead of $\underline{c}_{t1}$, the interpolation takes place between $\underline{c}(1)$ and $\underline{c}_2$. This gives:

$$\underline{c}(2) = \underline{c}(1) + \frac{\underline{c}_{t2} - \underline{c}(1)}{B-1} + \Delta e_1^* \underline{y}_1 \quad (6.12)$$

Thus for the n 'th symbol, the algorithm, termed modified LMS-aided interpolation, becomes:

$$\underline{c}(n+1) = \underline{c}(n) + \frac{\underline{c}_{t2} - \underline{c}(n)}{B-n} + \Delta e_n^* \underline{y}_n, \quad n=0, \dots, B-1 \quad (6.13)$$

If the LMS algorithm moves the coefficients in the right direction, then this algorithm has the effect of interpolating over successively shorter blocks. Note that for $n=B-1$, $\underline{c}(B) = \underline{c}_{t2} + \Delta e_{B-1}^* \underline{y}_{B-1}$, regardless of the previous values of $\underline{c}(n)$. Hence, this algorithm pulls the coefficients toward $\underline{c}_{t2}$ as it approaches the end of the block.

To investigate the above algorithms, simulations are run to measure the worst SNR difference as a function of block length. In doing this, 100 blocks of a given length are simulated, and the values of worst SNR difference are averaged. In all of the simulations of this section, a 3 path channel is used with delays of 0, 2T and 4T, and a DFE(10,4) is used in the receiver. The first simulations compare LMS-aided interpolation with the modified algorithm for different values of α , the constant in the normalized step size. In all cases, it is found that the modified LMS-aided interpolation gives better results, so only this algorithm will be discussed in the remainder of the thesis. The next simulations compare worst SNR difference versus block length for constant coefficients and linear interpolation. The results are shown in Figure 6.9, and the improvement obtained by using interpolation is apparent.

Finally, linear interpolation is compared to modified LMS-aided interpolation. The simulated channel has a fade rate of 0.5 Hz, and the constant α has a value of 0.1. The results are shown in Figure 6.10. In this case the modified algorithm has a higher 'worst SNR difference' for shorter blocks, but a smaller value for longer blocks. This behavior is likely due to the introduction of estimation noise by the LMS

Figure 6.9
Comparison of Constant Coefficients with
Interpolation. Fade Rate = 0.3Hz
Channel snr = 15dB

Figure 6.10
Comparison of Interpolation with the
Modified Algorithm. Fade Rate = 0.5Hz
Channel snr = 15 dB

algorithm. At smaller block lengths, the excess mean squared error from estimation noise is greater than that due to the difference between the optimum and interpolated coefficients. At longer block lengths, the difference between optimum and interpolated coefficients is large enough that the improvement from the LMS adaptation is greater than the degradation due to its estimation noise. Thus, if a block length of 200 or more can be used when the fade rate is 0.5 Hz, the modified LMS-aided interpolation may offer some improvement. Similar simulations were run for a fade rate of 0.15 Hz, in which the modified algorithm gave a higher worst SNR difference for block lengths up to 500 symbols.

The last simulation results of block by block with interpolation present curves of bit error probability versus channel SNR. In this case, instead of using optimum coefficients at each end of the data block, training blocks are used to obtain coefficient estimates. The bit error probability is approximated by assuming that Gray coding has been used, so that the bit error rate is about half the symbol error rate. The symbol error rate is estimated by transmitting a known number of symbols and counting the number of errors. The first of these results, shown in Figure 6.11, compare linear interpolation with modified LMS-aided interpolation. As before, the multipath model has delays of 0, $2T$, $4T$, and the fade rate is 0.5 Hz. Also, a DFE(10,4) is used, and receiver decisions are used in the feedback filter and in the LMS adaptation. The training block length is 65 and the data block length is 200 symbols, giving an overhead of 25%. As can be seen, the modified algorithm does not improve the error rate, even though it had a smaller value of worst SNR difference. Thus, although the lowest output SNR in the block was increased by using LMS, the presence of estimation noise must have reduced the output SNR at other points in the block.

Figure 6.11
Error Rate for Modified LMS and Interpolation.
Fade Rate = 0.5, $\alpha = 0.1$, data block length = 200.

More important, however, is the fact that the bit error rate for this block length is quite high, so that in practice the block length must be smaller. Therefore, for the block lengths that give a useful error rate, the use of LMS does not offer an improvement in performance.

The next results compare the square root Kalman algorithm in decision directed mode to block by block with interpolation. Periodic retraining is used in the Kalman algorithm. This is done by sending a training sequence for a given number of data symbols. This simply provides known symbols for use in the adaptation, and no restarting of the algorithm is done. The training sequence length is the same as the training block length of the block by block method, and, for the same percentage overhead, the number of data symbols is equal to the data block length. In all the results that follow, the training sequence length is 65. Thus the percentage of overhead is controlled by the data block length. For comparison, the performance curve for the square root Kalman algorithm in ideal reference mode is also presented.

The results for a fade rate of 0.5 Hz are shown in Figure 6.12. It can be seen that at low channel SNR's, block by block with interpolation has a better performance than decision directed square root Kalman. This is due to the higher error rate at low SNR's and the fact that in block by block with interpolation, decision errors do not affect adaptation. However, it can also be seen that in the range of channel SNR's presented, the curves for block by block with interpolation level out to an irreducible bit error rate (BER). Since any adaptive algorithm is limited by its ability to estimate the optimum coefficients, even the Kalman algorithm in ideal reference mode will have an irreducible BER.

Figure 6.12
Error Rate Comparison of Kalman to Block by Block
with Interpolation. Fade Rate = 0.5 Hz.

Figure 6.12 indicates, however, that block by block with interpolation has a higher irreducible BER. This is due to the excess mean squared error that results from deviation of optimum coefficients from a straight line trajectory. A shorter data block decreases the irreducible BER at the expense of an increase in overhead. For a fade rate of 0.5 Hz, though, an overhead of as much as 40% still gives a probability of error that is greater than 10^{-3} . On the other hand, the decision directed Kalman algorithm achieves a much lower error rate as the channel SNR is increased. Thus, at this fade rate, block by block with interpolation is inferior to continuous decision-directed adaptive algorithms if the signal to noise ratio is high.

Figure 6.13 shows the results for a 0.15 Hz fade rate. For an overhead of 11%, the same type of behaviour is noticed as in the case of a 0.5 Hz fade rate. When the overhead is increased to 18%, however, block by block with interpolation shows better performance than decision directed Kalman for all channel SNR's presented. Thus, in this case, block by block with interpolation has an advantage, due to its lack of decision direction, for channel SNR's up to 25 or 30 dB. Although the curve shows sign of levelling out, the probability of error at this point may be tolerable.

Finally, it was mentioned earlier that if the data block is long enough, then the block by block approach will be computationally efficient. With interpolation, the computation per block is the number of operations required by the covariance method and Cholesky's algorithm, plus the N divisions required to find $(c_{t2} - c_{t1})/B$. The effective computation per

Figure 6.13
Error Rate Comparison of Kalman to Block by Block
with Interpolation. Fade Rate = 0.15Hz

symbol is this amount divided by $(M+3)$, the sum of the training and data block lengths, plus the N multipliers needed to compute the equalizer output. This gives:

$$\frac{(M-N_b) \left(\frac{N}{2}(N+1)+N \right) + (N^3+9N^2+2N)/6 + N}{(M+B)} + N \quad (6.14)$$

multiplies and divisions per symbol. Note that this implies another delay of $(M+B)$ to allow for computation.

For $M=65$, $N_b=4$, $N=14$, and $B=300$ (say, for a fade rate of 0.15 Hz), the computation per symbol for different algorithms is given in Table 3.

Table 3	
Multipliers and Dividers Per Symbol	
For $N=14$	
Block By Block	36
Square Root Kalman	385
Fast Kalman	500

In this case, the small computational load required by block by block and square root Kalman is due to the small number of coefficients. If N is doubled, which in turn doubles M , the computation loads becomes as shown in Table 4. Here the fast Kalman algorithm becomes more efficient than square root Kalman, but the block by block approach is still far more efficient than either of the Kalman algorithms.

Table 4
Multiplies and Divides Per Symbol
For $N=28$

Block By Block	163
Square Root Kalman	1358
Fast Kalman	948

Thus, for lower fade rates, block by block with interpolation does indeed offer a significant saving in computational complexity.

Alternatively, if the computational power is available, the coefficients can be found with a smaller delay.

CHAPTER 7

CONCLUSIONS7.1 Concluding Discussion

In the completion of this thesis, much has been learned about the feasibility of block by block equalization for the HF channel. First, it was found that the optimum equalizer coefficients are subject to abrupt changes. Thus, a block by block approach which uses constant coefficients during the data block requires a very large overhead. Furthermore, the use of an LMS algorithm does not allow a longer data block when such an abrupt change occurs. An approach in which coefficients are interpolated between the estimated values at each end of the block offers a great improvement over the use of constant coefficients. This is because knowledge of the slope of the coefficient variation is provided, alleviating the effect of a rapid change. The price paid for this is a delay in the receiver decisions. Again, the use of an LMS algorithm did not improve performance. The problem with the LMS algorithm in the simulations is that, as μ increases, estimation noise becomes too large before the adaptation speed becomes adequate. This is probably due to a large eigenvalue spread.

It was found that block by block with interpolation has an irreducible bit error rate. On the other hand, even in decision direction, the Kalman algorithm did not show signs of an irreducible BER for channel SNR's up

to 25 dB. This indicates that, at SNR's corresponding to an irreducible BER, the Kalman coefficients tracked the optimum setting much more closely than the interpolated ones. Thus, especially at higher fade rates, the problem of disruption of adaptation due to receiver errors is more than compensated by the ability to compute a coefficient estimate at every symbol interval.

At lower channel SNR's, block by block with interpolation has an advantage because of its lack of decision direction. Additionally, if the fade rate is low enough and the overhead is sufficient, then this advantage persists up to SNR's where the error rate is tolerable.

Finally, the data block lengths that can be used at lower fade rates allow a great reduction in the computational complexity. In effect, a block by block approach allows a tradeoff between computation complexity and tracking ability. In addition, the use of interpolation allows the recovery of some tracking ability at the expense of a delay.

7.2 Suggestions for Further Research

All the results in this thesis deal with a simulated HF channel. While the model that is used has been experimentally confirmed in [6], the performance of adaptive equalization should be tested on real HF channels. Some work has been done in [2] and [11], but a wider range of channels should be observed to determine the distribution of possible fade rates and channel SNR's. Finally, the block by block approach with interpolation could be investigated on a real channel and compared with the Kalman algorithm.

When using adaptive equalization on the HF channel, the actual fade rate may not be known. This is a problem because in any algorithm some parameters depend on the fade rate, whether it is Δ , λ , or the block length [11]. The performance of symbol by symbol algorithms is sensitive to Δ or λ because of the effect on estimation noise. The training block length in the block by block approach has a similar effect. Thus, some research could be done to determine the sensitivity of algorithm parameters to changes in the fade rate, especially when decision direction is used.

Finally, the requirement of periodic training for the Kalman algorithm has been discussed. It has been shown how the probability of error improves as the overhead is increased. However, an improvement in the BER can also be achieved by coding. Thus, some research could be done to determine the best proportions of training and coding overheads.

REFERENCES

- [1] N.M. Maslin, "High Data Rate Transmission Over HF Links", The Radio and Electronic Engineer, Vol.52, No.2, February 1982, pp.75-87.
- [2] R. Tront, "Performance of Kalman Decision-Feedback Equalization in HF Data Modems", M.Sc. Dissertation, University of British Columbia, December 1983.
- [3] P. Monsen, "Feedback Equalization for Fading Dispersive Channels", IEEE Trans. Information Theory, January 1971, pp.56-64.
- [4] R. Freeman, Telecommunication Transmission Handbook, 2nd Ed., John Wiley and Sons, New York, 1981.
- [5] J. Proakis, Digital Communications, McGraw-Hill, New York, 1983.
- [6] C. Watterson, J. Juroshek, W. Bensema, "Experimental Confirmation of an HF Channel Model", IEEE Trans. Communication Technology, Vol.COM-18, December 1970, pp.792-803.
- [7] P.A. Bello, B.D. Nelin, "The Influence of Fading Spectrum on the Binary Error Probabilities of Incoherent and Differentially Coherent Matched Filter Receivers", IRE Trans. on Communication Systems, Vol.CS-10, June 1962, pp.160-168.
- [8] P.A. Bello, B.O. Nelin, "The Effect of Frequency Selective Fading on the Binary Error Probabilities of Incoherent and Differentially Coherent Matched Filter Receivers", IEEE Trans. on Communication Systems, Vol.CS-11, June 1963, pp.170-186.
- [9] M.J. Di Toro, "Communication in Time-Frequency Spread Media Using Adaptive Equalization", Proceedings of the IEEE, Vol.56, No.10, October 1968, pp.1653-1679.
- [10] B. Widrow, J.M. McColl, M.G. Larimore, C.R. Johnson, Jr., "Stationary and Nonstationary Learning Characteristics of the LMS Adaptive Filter", Proceedings of the IEEE, Vol.64, No.8, August 1976, pp.1151-1162.
- [11] E. Eleftheriou, "Adaptive Filtering in a Time-Varying Environment With Applications to Equalization of Fading HF Channels", Ph.D. Dissertation, Carleton University, October 1985.
- [12] R. Gitlin, S. Weinstein, "Fractionally-Spaced Equalization: An Improved Digital Transversal Equalizer", Bell System Tech. J., Vol.60, No.2, February 1981, pp.275-296.

- [13] S. Qureshi, "Adaptive Equalization", Proceedings of the IEEE, Vol.73, No.9, September 1985, pp.1349-1387.
- [14] J. Salz, "Optimum Mean-Square Decision Feedback Equalization", Bell System Tech. J., Vol.52, No.8, October 1973, pp.1341-1373.
- [15] D.L. Duttweiler, J.E. Mazo, D.G. Messerschmitt, "An Upper Bound On the Error Probability in Decision-Feedback Equalization", IEEE Trans. Information Theory, Vol.IT-20, No.4, July 1974, pp.490-497.
- [16] D.D. Falconer, H.M. Hafez, A.U.H. Sheikh, E. Eleftheriou and M. Tobis, "A Comparative Study of Maximum Likelihood and Adaptive Equalization for High Rate High Frequency Data Communications", Department of Systems and Computer Engineering, Carleton University, Technical Report, April 1983.
- [17] M.L. Honig, D.G. Messerschmitt, Adaptive Filters, Kluwer Academic Publishers, Boston, 1984.
- [18] J.G. Proakis, J.H. Miller, "An Adaptive Receiver for Digital Signaling Through Channels With Intersymbol Interference", IEEE Trans. on Information Theory, Vol.IT-15, No.4, July 1969, pp.484-497.
- [19] G. Ungerboeck, "Theory on the Speed of Convergence in Adaptive Equalization for Digital Communication", IBM Journal of Research and Development, November 1972, pp.546-555.
- [20] R.D. Gitlin, S.B. Weinstein, "On The Required Tap-Weight Precision for Digitally Implemented, Adaptive, Mean-Squared Equalizers", Bell System Tech. J., Vol.58, No.2, February 1979, pp.301-321.
- [21] F.M. Hsu, A.A. Giordano, H.E. de Pedro, J.G. Proakis, "Adaptive Equalization Techniques for High-Speed Transmission on Fading Dispersive HF Channels", National Telecommunications Conference, December 1980, pp.58.1.1-58.1.7.
- [22] D. Godard, "Channel Equalization Using A Kalman Filter for Fast Data Transmission", IBM Journal of Research and Development, May 1974, pp.267-273.
- [23] D.D. Falconer, L. Ljung, "Application of Fast Kalman Estimation to Adaptive Equalization", IEEE Trans. on Communications, Vol.COM-26, No.10, October 1978, pp.1434-1446.
- [24] F.M. Hsu, "Square Root Kalman Filtering for High-Speed Data Received Over Fading Dispersive HF Channels", IEEE Trans. on Information Theory, Vol.IT-28, No.5, September 1982, pp.753-763.
- [25] F. Ling, J.G. Proakis, "Adaptive Lattice Decision-Feedback Equalizers - Their Performance and Application to Time-Variant Multipath Channels", IEEE Trans. on Communications, Vol.COM-33, No.4, April 1985, pp.348-356.

- [26] R.L. Burden, J.D. Faires, A.C. Reynolds, Numerical Analysis, 2nd Ed., PWS Publishers, Boston, 1981.
- [27] J.M. Cioffi, "The Block-Processing FTF Adaptive Algorithm", IEEE Trans. on Acoustics, Speech, and Signal Processing, Vol. ASSP-34, No.1, February 1986, pp.77-90.
- [28] A.V. Oppenheim, R.W. Schaffer, Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975.

APPENDIX A

IIR FILTER FOR MULTIPATH TAP GAIN GENERATION

Because of the $T/2$ sampling at the channel output, not all of the channel output samples are used. This can lead to a saving in computation since if a sample is not used, there is no need to compute it. The diagram of the multipath model and additive noise is redrawn in Figure A.1, where the location of the $T/2$ sampler has been moved. In this case the multipath and noise filter output is only computed at corresponding $T/2$ spaced sample times. As a result, the tap gains themselves need only be computed at $T/2$ intervals. Because tap gain generation involves Gaussian random number generation and filtering for each path, this leads to a substantial saving in computation.

The IIR filter design starts with the transfer function of the analog Butterworth filter, and uses the impulsive invariance method to derive the digital filter. The sampling interval that is assumed in the derivation is $T/2$, and the resulting z transform transfer function is:

$$H(z) = \frac{\sqrt{2}}{2 + f_r} \left[\frac{z \exp(-f_r T / \sqrt{2}) \sin(+f_r T / \sqrt{2})}{z^2 - 2z \exp(-f_r T / \sqrt{2}) \cos(-f_r T / \sqrt{2}) + \exp(-2f_r T / \sqrt{2})} \right] \quad (A.1)$$

where f_r is the desired fade rate in Hz.

Figure A.1
Modified Multipath Simulation Model

Finally, the noise that is input to the IIR filter has a variance that is equal to the desired variance of the tap gain. Thus, in order to preserve this variance after filtering, it is necessary to divide the input noise samples by the square root of the energy of the filter [28]. Given a general transfer function of the form:

$$H(z) = \frac{b_1 z}{z^2 + a_1 z + a_2} \quad (\text{A.2})$$

the energy can be found as follows. First factor the denominator to get

$$H(z) = \frac{b_1 z}{(z-r_1)(z-r_2)}$$

where

$$r_1, r_2 = \frac{-a_1 \pm \sqrt{a_1^2 - 4a_2}}{2} \quad (\text{A.3})$$

Then, using partial fraction expansion:

$$H(z) = \frac{K z}{z-r_1} - \frac{K z}{z-r_2}$$

where

$$K = \frac{b_1}{r_1 - r_2} \quad (\text{A.4})$$

The inverse transform of this is

$$h(n) = K r_1^n - K r_2^n, \quad n \geq 0, \quad (\text{A.5})$$

and finding the energy in the time domain gives

$$\sum_{n=-\infty}^{\infty} h(n)^2 = K^2 \left[\frac{1}{1-r_1^2} + \frac{1}{1-r_2^2} - 2 \operatorname{Re} \left(\frac{1}{1-r_1 r_2^*} \right) \right] \quad (\text{A.6})$$

APPENDIX B

COMPUTATION OF ADDITIVE NOISE VARIANCE

The average signal to noise ratio at the input to the receiver is:

$$\text{SNR} = 10 \log_{10} \frac{\sigma_s^2}{\sigma_n^2} \quad (\text{B.1})$$

where σ_n^2 is the variance of the noise samples and σ_s^2 is the average signal power. Thus, for a desired SNR, the required noise variance is

$$\sigma_n^2 = \frac{\sigma_s^2}{10^{\text{SNR}/10}} \quad (\text{B.2})$$

To find the average signal power, first express the multipath output as the convolution of the transmitted signal and the multipath model:

$$s(t) = \sum_{j=1}^N G_j(t) u(t-\tau_j) \quad (\text{B.3})$$

where N is the number of paths. Substituting this into the definition of signal power gives:

$$\sigma_s^2 = E[s^*(t)s(t)] = \sum_{i=1}^N \sum_{j=1}^N E[G_i^*(t)G_j(t)] E[u^*(t-\tau_i)u(t-\tau_j)] \quad (\text{B.4})$$

Then, assuming the tap gains to be independent and the transmitted signal stationary, this becomes:

$$c_s^2 = \sum_{j=1}^N c_{G_j}^2 \quad (B.5)$$

where $c_{G_j}^2$ is the variance of the j -th tap gain and c_u^2 is the signal power out of the pulse shaping filter. The variance of each tap gain is input to the program, whereas c_u^2 must be calculated from the coefficients of the pulse shaping filter.

To find c_u^2 from the pulse shaping filter coefficients, it is necessary to note that, from the point of view of the receiver, the pulse is effectively sampled at $T/2$ intervals. To see this, first express the samples of the multipath output as:

$$s(n \frac{T}{N_s}) = s((mN_s + i) \frac{T}{N_s}) \quad 0 \leq i \leq N_s - 1 \quad (B.6)$$

where N_s is the number of samples per symbol interval. Note that $T/2$ spaced sampling instants correspond to $i=0$ and $i=N_s/2$. Using this notation, (B.3) becomes

$$s((mN_s + i) \frac{T}{N_s}) = \sum_{j=1}^N G_j((mN_s + i) \frac{T}{N_s}) u((mN_s + i - z_j) \frac{T}{N_s}), \quad (B.7)$$

where the delay for the j -th path is $z_j T/N_s$. Next, recall that the transmitted signal is obtained by passing an impulse every N_s samples through the pulse shaping filter. Let $d(n \frac{T}{N_s})$ represent the input impulse train, so that $d(n \frac{T}{N_s})$ is nonzero only when n is a multiple of N_s . An expression for a transmitted signal sample is then:

$$u(n \frac{T}{N_s}) = \sum_{k=0}^{N_{lpf}-1} p_k d((n-k) \frac{T}{N_s}), \quad (B.8)$$

where p_k are the coefficients of the pulse shaping filter and N_{1pf} is the number of coefficients. Using the above notation, the transmitted signal in the j -th path of the multipath model is:

$$u((mN_s + i - z_j) \frac{T}{N_s}) = \sum_{k=0}^{N_{1pf}} p_k d((mN_s + i - k - z_j) \frac{T}{N_s}) \quad (B.9)$$

Now $d((mN_s + i - k - z_j) \frac{T}{N_s})$ is nonzero only when $i - k - z_j = -aN_s$, where a is some integer. For $i=0$ and $i=N_s/2$, values of k in the summation of (B.9) for which the corresponding term is nonzero are those that satisfy:

$$\begin{aligned} k &= aN_s - z_j \\ \text{and} \\ k &= aN_s - \frac{N_s}{2} - z_j \end{aligned} \quad (B.10)$$

These values correspond to $T/2$ spaced samples of the pulse shaping filter's impulse response. Therefore, effectively, the transmitted signal in any one path is the same as what would have been produced by a $T/2$ spaced transversal filter. The difference between the paths is the phase of the $T/2$ sampling, which depends on the path's delay.

This fact is needed in finding the signal power at the output of the pulse shaping filter. Consider the analog pulse spectrum $P(f)^2$. For uncorrelated data, the power spectrum of the analog transmitted signal is $\frac{1}{T} P(f)^2$. Since the raised cosine spectrum is bandlimited, the signal power is:

$$S_u^2 = \frac{1}{T} \int_{-1/T}^{1/T} P(f)^2 df \quad (B.11)$$

When the pulse is sampled at a rate of $2/T$, the spectrum becomes, in

range $-1/T \leq t \leq 1/T$,

$$\hat{p}(f) = \frac{2}{T} P(f) \quad (B.12)$$

Using this expression, the signal power can be expressed in terms of the spectrum of the sampled pulse as:

$$\begin{aligned} \sigma_u^2 &= \frac{1}{T} \int_{-1/T}^{1/T} \left[\frac{T}{2} \hat{p}(f) \right]^2 df \\ &= \frac{1}{2} \left[\frac{T}{2} \int_{-1/T}^{1/T} \hat{p}(f)^2 df \right] \end{aligned} \quad (B.13)$$

Then, using Parseval's theorem,

$$\begin{aligned} \sigma_u^2 &= \frac{1}{2} \sum_n \left| p\left(n \frac{T}{2}\right) \right|^2 \\ &= \frac{1}{2} \frac{2N_1 p_f}{N_s} \sum_{k=0}^{N_s} \left| p_{kN_s/2} \right|^2 \end{aligned} \quad (B.14)$$

where it is assumed that the FIR filter is long enough to approximate the pulse with negligible error. Finally, the expression for the average signal power at the multipath output is:

$$\sigma_s^2 = \frac{1}{2} \frac{2N_1 p_f}{N_s} \sum_{k=0}^{N_s} \left| p_{kN_s/2} \right|^2 \sum_{j=1}^N c_j^2 \quad (B.15)$$

This, used in (B.2), gives the noise variance for the desired SNR.

The variance σ_n^2 is used to generate the white noise samples at the input to the noise filter. To keep the same noise variance at the output of the filter, it is necessary to divide the input noise samples by the square root of the energy of the filter. This energy is simply the sum of the squared magnitudes of the FIR noise filter coefficients.

APPENDIX C

SUBROUTINE LISTINGS FOR INITIALIZATION
AND CHANNEL SIMULATION .

SUBROUTINE INIT(PARAM)

C This routine provides initialization of all parameters
 C used in the simulation. Global variables are those
 C passed in COMMON to other routines.
 C NAG library routine G05CCF is used.
 C Also, functions ENERG and RAISED are called.

C GLOBAL VARIABLES
 C Channel simulation

INTEGER NSAMP

C -number of samples per symbol interval
 C (must be even number)

DOUBLECOMPLEX DELINE(0:200)

C -(T/NSAMP) spaced delay line for multipath model

INTEGER MAXDEL

C -maximum number of delays used in DELINE

DOUBLECOMPLEX CHTAP(10)

C -tap gains of multipath model

INTEGER NPATH

C -number of paths in multipath model

INTEGER MPDEL(10)

C -corresponding delays, in number of samples, of
 C each path

DOUBLECOMPLEX AIIR(10,4)

C -delay lines for iir filters used in generating
 C tap gains

DOUBLEPRECISION COEF(3)

C -coefficients of iir filters

DOUBLEPRECISION MPGAIN(10)

C -rms gains of taps in multipath model

DOUBLEPRECISION SQENRG

C -square root of energy of iir filter, used to
 C normalize variance of white noise input

DOUBLECOMPLEX LPFLIN(0:200)

C -delay line for pulse shaping fir filter

DOUBLEPRECISION ALPF(0:200)

C -coefficients of pulse shaping low pass filter

INTEGER NLPF

C -maximum delay in LPFLIN
 C (must be divisible by NSAMP)

```

C      INTEGER TIMING
C          -position of first T/2 sampling time in symbol interval
C          (between 1 and NSAMP/2-1)

C      DOUBLEPRECISION ADGAIN(1)
C          -standard deviation of additive noise

C      DOUBLECOMPLEX NOISLN(0:200)
C          -delay line for fir filter that filters additive noise

C      DOUBLEPRECISION WLPF(0:200)
C          -coefficients of additive noise filter

C      DOUBLEPRECISION NCOR(0:30)
C          -correlation coefficients of additive noise

C      DOUBLEPRECISION WLPFEN
C          -energy of additive noise filter, used to normalize
C          variance of white noise input

C      Receiver simulation

C      DOUBLECOMPLEX FDTAP(30)
C          -delay line for forward filter of equalizer

C      DOUBLECOMPLEX BKTAP(15)
C          -delay line for feedback filter

C      DOUBLECOMPLEX REFER
C          -current received data symbol

C      DOUBLECOMPLEX FDCOEF(30)
C          -forward filter coefficients

C      DOUBLECOMPLEX BKCOEF(15)
C          -feedback filter coefficients

C      INTEGER NFDTAP
C          -number of forward filter coefficients

C      INTEGER NBKTAP
C          -number of feedback filter coefficients

C      DOUBLEPRECISION ALPHA
C          -adaptation constant

C      LOCAL VARIABLES

C      CHARACTER * 10 PARAM
C          -name of input file

C      DOUBLEPRECISION SNR
C          -desired average snr

C      DOUBLEPRECISION FACTOR

```



```

C      -snr, but not in dB, used to compute noise variance
      DOUBLEPRECISION SIGVAR
C      -signal power
      DOUBLEPRECISION LPFENG
C      -signal power at output of pulse shaping lpf.
      INTEGER LPFDEL
C      -signal delay due to lpf
      DOUBLEPRECISION CUTOFF, ROLLOFF
C      -cutoff frequency and rolloff factor of raised
C      cosine pulse shaping lpf
      DOUBLEPRECISION FS
C      -sampling rate
      DOUBLEPRECISION FS1
C      -(2/T), twice the symbol rate
      DOUBLEPRECISION FC
C      -desired fade rate, 3dB frequency of iir filters
      DOUBLEPRECISION X,Y,PI
C      -used in computing iir coefficients
      DOUBLEPRECISION TPENRG
C      -used in computing SQENRG
      DOUBLEPRECISION WCUT,WROLL
C      -cutoff frequency and rolloff_factor of noise lpf
      DOUBLEPRECISION LPBAND
C      -total low pass bandwidth of system, used to
C      compute WROLL
      COMMON /CHAN1/ DELINE, CHTAP,MAXDEL, NPATH, MPDEL,NSAMP
      COMMON /CHAN2/ AI1R, COEF, MPGAIN, SQENRG
      COMMON /CHAN3/ LPFLIN,ALPF,ADGAIN,TIMING,NLPF
      COMMON /CHAN4/ NOISLN,WLPF,NCOR,WLPFEN
      COMMON /RECV1/ FDTAP,BKTAP,REFER,FDCOEF,BKCOEF
      COMMON /RECV2/ NFDTAP,NBKTAP,ALPHA
C      Initialize random number generator in NAG
C      library to use a different seed each time program
C      is run.
      CALL G05CCF
C      Open file indicated by PARAM and read in
C      parameters for simulation. These describe the
C      multipath channel model including the initial
C      tap gain settings, the pulse shaping and
C      noise low pass filters, the snr, the sampling

```

```

C      and symbol rate, the timing, the DFE, and the
C      adaptation constant.
      OPEN(13,NAME=PARAM,USAGE='INPUT')

      READ(13,10) MAXDEL
      READ(13,10) NPATH
      READ(13,20) (MPDEL(I),I=1,NPATH)
      READ(13,30) FC,FS,NSAMP
      READ(13,40) (MPGAIN(I),I=1,NPATH)
      READ(13,45) (CHTAP(I),I=1,NPATH)
      READ(13,100) NFDTAP,NBKTAP
      READ(13,105) ALPHA
      READ(13,110) SNR,TIMING,LPBAND
      READ(13,115) NLPF,CUTOFF,ROLLOFF
C      FORMAT STATEMENTS
10      FORMAT(20X,I3)
20      FORMAT(20X,10(I3,1X))
30      FORMAT(20X,F6.3,1X,F7.1,1X,I2)
40      FORMAT(20X,10(F6.3,1X))
45      FORMAT(20X,20(F6.3,1X))
100     FORMAT(20X,I3,1X,I3)
105     FORMAT(20X,F7.4)
110     FORMAT(20X,F6.2,1X,I3,1X,F7.1)
115     FORMAT(20X,I3,1X,F7.1,1X,F5.3)
*
C      Using the desired fade rate and a sample
C      rate of (2/T), compute the 2nd order
C      Butterworth iir filter coefficients.
      FS1=FS*2.0/FLOAT(NSAMP)
      PI=4.0*DATAN(1.0D0)
      X=DSQRT(2.0D0)*PI*FC/FS1
      Y=DEXP(-X)
      COEF(1)=Y*DSIN(X)/(FS1*X)
      COEF(2)=2.0*Y*DCOS(X)
      COEF(3)=-Y**2

C      Compute energy of impulse response of iir filter.
      TPENRG=ENERG(COEF)
      SQENRG=DSQRT(TPENRG)

C      Initialize all filter delay lines.
      DO 50 I=0,MAXDEL
          DELINE(I)=(0.0,0.0)
50      CONTINUE
      DO 70 I=1,NPATH
          AIIR(I,1)=(0.0,0.0)
          AIIR(I,2)=(0.0,0.0)
          AIIR(I,3)=CHTAP(I)
          AIIR(I,4)=CHTAP(I)
70      CONTINUE
      DO 160 I=0,NLPF
          LPFLIN(I)=(0.0,0.0)
          NOISLV(I)=(0.0,0.0)
160     CONTINUE

```

```

C      Initialize Equalizer taps and coefficients.
DO 80 I=1,NFDTAP
    FDTAP(I)=(0.0,0.0)
    FDCOE(I)=(0.0,0.0)
80  CONTINUE
    REFER=(0.0,0.0)
DO 90 I=1,NBKTAP
    BKTAP(I)=(0.0,0.0)
    BKCOE(I)=(0.0,0.0)
90  CONTINUE

C      Compute pulse shaping and noise lpf coefficients.
DO 95 I=0,200
    ALPF(I)=0.0
    WLPF(I)=0.0
95  CONTINUE
    WCUT=FS/NSAMP
    WROLL=(LPBAND-WCUT)/WCUT
    LPFDEL=NLPF/2
DO 170 I=0,NLPF
    ALPF(I)=RAISED((I-LPFDEL)/FS,ROLLOFF,CUTOFF)
    WLPF(I)=RAISED((I-LPFDEL)/FS,WROLL,WCUT)
170 CONTINUE

C      Compute signal power at output of lpf.
    LPFENG=0.0
DO 171 I=0,2*NLPF/NSAMP
    LPFENG=LPFENG+ALPF((NSAMP*I)/2)**2
171 CONTINUE
    LPFENG=LPFENG/2.0

C      Compute average signal power at output of multipath.
    SIGVAR=0.0
DO 150 I=1,NPATH
    SIGVAR=SIGVAR + MPGAIN(I)**2
150 CONTINUE
    SIGVAR=SIGVAR*LPFENG

C      Compute noise variance.
    FACTOR=10**((SNR/10.0))
    ADGAIN(1)=DSQRT(SIGVAR/FACTOR)

C      Compute correlation coefficients of additive noise.
DO 200 M=0,NFDTAP-1
    NCOR(M)=0.0
    DO 210 I=0,NLPF-M-NSAMP/2
        NCOR(M)=NCOR(M)+WLPF(I)*WLPF(I+M+NSAMP/2)
210 CONTINUE
200 CONTINUE
    WLPFEN=NCOR(0)
    NCOR(0)=ADGAIN(1)**2
DO 220 M=1,NFDTAP-1
    NCOR(M)=NCOR(0)*NCOR(M)/WLPFEN
220 CONTINUE

```

```
RETURN
END
```

```
C      Function to compute energy of iir filter.
      DOUBLEPRECISION FUNCTION ENERG(COEF)
```

```
      DOUBLEPRECISION COEF(3),TERM1,TERM2
      DOUBLECOMPLEX B1,A1,A2,DISC,R1,R2,K
```

```
      B1=DCMPLX(COEF(1),0.000)
      A1=DCMPLX(-COEF(2),0.000)
      A2=DCMPLX(-COEF(3),0.000)
      DISC=CDSQRT((A1**2)-(4.0*A2))
      R1=(-A1+DISC)/2.0
      R2=(-A1-DISC)/2.0
      K=B1/(R1-R2)
      TERM1=1.0/(1.0-CDABS(R1)**2) + 1.0/(1.0-CDABS(R2)**2)
      TERM2=2.0*DREAL(1.0/(1.0-R1*DCONJG(R2)))
      ENERG=(CDABS(K)**2)*(TERM1-TERM2)
      RETURN
      END
```

```
C      Function to calculate raised cosine impulse response.
```

```
      DOUBLEPRECISION FUNCTION RAISED(T,A,W)
```

```
      DOUBLEPRECISION T,A,W,PI,TEMP,TEMP2,LEVEL,EPS
```

```
      EPS=1.0E-8
      PI=4.0*DATAN(1.000)
      IF(DABS(T).LT.EPS) THEN
          TEMP=1.0
      ELSE
          TEMP=DSIN(2.0*PI*T*W)/(2.0*PI*T*W)
      ENDIF
      LEVEL=1.0/(4.0*A*W)
      IF ((DABS(T).LT.LEVEL+EPS).AND.(DABS(T).GT.LEVEL-EPS)) THEN
          TEMP2=PI/4.0
      ELSE
          TEMP2=DCOS(2.0*PI*A*T*W)/(1-(4.0*A*T*W)**2)
      ENDIF
      RAISED=TEMP*TEMP2
      IF(DABS(RAISED).LT.EPS) RAISED=0.0
      RETURN
      END
```

SUBROUTINE HFSIM(DATA,CHOUT)

```

C   This routine simulates the fading multipath channel.
C   It is called once per symbol interval, so that NSAMP
C   samples are processed with each call. The value
C   of the data symbol is input, is passed as an
C   impulse through the pulse shaping filter, and the pulse
C   is sent through the multipath model. Noise is added
C   to the multipath output. The final output of the routine
C   is two T/2 spaced samples for input to the equalizer.

C   Routine WGNV is called, which in turn uses
C   random number generators from the NAG library.

C   GLOBAL VARIABLES (see INIT)

DOUBLECOMPLEX DELINE(0:200),CHTAP(10),AIIR(10,4)
DOUBLECOMPLEX LPFLIN(0:200),NOISLN(0:200)
INTEGER MAXDEL,NPATH,MPDEL,NSAMP,NLPF,TIMING
DOUBLEPRECISION COEF(3),MPGAIN(10),SQENRG,ALPF(0:200)
DOUBLEPRECISION ADGAIN(1),WLPF(0:200),NCOR(0:30),WLPFEN

C   LOCAL VARIABLES

DOUBLECOMPLEX DATA
C   -data symbol input
DOUBLECOMPLEX CHOUT(2)
C   -two T/2 spaced output samples
DOUBLECOMPLEX MPOUT
C   -multipath output sample
DOUBLECOMPLEX NOISE(10)
C   -vector of white noise samples to be used
C   in generating the multipath tap gains
DOUBLECOMPLEX ADNOIS(1)
C   -white noise sample to be used in generating
C   additive noise
DOUBLECOMPLEX LPFOUT
C   -pulse shaping filter output sample
DOUBLECOMPLEX WLPOUT
C   -noise filter output sample

COMMON /CHAN1/ DELINE,CHTAP,MAXDEL,NPATH,MPDEL,NSAMP
COMMON /CHAN2/ AIIR, COEF, MPGAIN, SQENRG
COMMON /CHAN3/ LPFLIN,ALPF,ADGAIN, TIMING,NLPF
COMMON /CHAN4/ NOISLN,WLPF,NCOR,WLPFEN

C   Simulate for NSAMP samples.
DO 100 K=1,NSAMP

C       Send data, impulse through pulse shaping filter
C       and white noise sample through noise filter.

C       Shift samples in delay lines.
DO 55 J=0,NLPF-1

```

```

        LPFLIN(NLPF-1)= LPFLIN(NLPF-1-1)
        NOISLN(NLPF-1)=NOISLN(NLPF-1-1)
55      CONTINUE

C      Input data impulse.
      IF (K.EQ.1) THEN
        LPFLIN(0)=DATA
      ELSE
        LPFLIN(0)=(0.000,0.000)
      ENDIF

C      Input white noise to noise filter.
      CALL WGNV(ADNOIS,1,ADGAIN)
      NOISLN(0)=ADNOIS(1)/DSQRT(WLPFEN)

C      Form pulse shaping filter, output.
      LPFOUT=(0.000,0.000)
      DO 60 I=0,NLPF
        LPFOUT=LPFOUT + LPFLIN(I)*ALPF(I)
60      CONTINUE

C      Send pulse sample through multipath.

C      Shift multipath delay line and input pulse sample.
      DO 65 I=0,MAXDEL-1
        DELINE(MAXDEL-1)=DELIN(MAXDEL-1-1)
65      CONTINUE
      DELINE(0)=LPFOUT

C      Only, form multipath and noise output if the current
C      sample corresponds to one of the T/2 spaced sampling
C      instants.
      IF((K.EQ.TIMING) .OR. (K.EQ.TIMING+(NSAMP/2))) THEN

C          Get tap gains from filtering noise.

C          Generate white gaussian noise vector.
          CALL WGNV(NOISE, NPATH, MPGAIN)

C          Filter noise vector with bank of nir filters.
          DO 20 I=1,NPATH
            AIIR(I,4)=AIIR(I,3)
            AIIR(I,3)=CHTAP(I)
            AIIR(I,2)=AIIR(I,1)
            AIIR(I,1)=NOISE(I)/SQENRG
            CHTAP(I)=(0.0,0.0)
            DO 10 J=1,3
              CHTAP(I)=CHTAP(I) + AIIR(I,J+1)*COEF(J)
10          CONTINUE
20          CONTINUE

C      Form multipath output sample.
      *PCOUT=(0.0,0.0)
      DO 70 I=1,NPATH

```

```

      MPOUT=MPOUT + DELINE(MPDEL(I))*CHTAP(I)
70      CONTINUE
      C      Form additive noise sample.
      WLPOUT=(0.0,0.0)
      DO 160 I=0,NLPP
          WLPOUT=WLPOUT+NOISLN(I)*WLPPF(I)
160      CONTINUE
      MPOUT=MPOUT+WLPOUT
      C      Assign T/2 spaced samples to output vector.
      IF (K.EQ.TIMING) THEN
          CHOUT(1)=MPOUT
      ELSE IF (K.EQ.TIMING+(NSAMP/2)) THEN
          CHOUT(2)=MPOUT
      ENDIF
      ENDIF
100      CONTINUE
      RETURN
      END

```

SUBROUTINE WGNV(NOISE,NPATH,MPGAIN)

```

C      Subroutine returns a vector NOISE which contains
C      complex gaussian random variables with zero mean
C      and standard deviation MPGAIN.
C
C      NAG library routine G05DDF is used.
C      Real and imaginary parts are divided by root 2
C      so that the complex number has a variance of MPGAIN**2.

```

```

      INTEGER NPATH
      DOUBLECOMPLEX NOISE(NPATH)
      DOUBLEPRECISION MPGAIN(NPATH), NR, NI, ROOT2
      ROOT2=DSQRT(2.000)
      DO 110 I=1,NPATH
          NR=G05DDF(0.000,MPGAIN(I))/ROOT2
          NI=G05DDF(0.000,MPGAIN(I))/ROOT2
          NOISE(I) = DCMPLX(NR,NI)
110      CONTINUE
      RETURN
      END

```

APPENDIX D

SUBROUTINE LISTINGS FOR COEFFICIENT ESTIMATION
AND OPTIMUM COEFFICIENT CALCULATION


```

SUBROUTINE CORPEL(D,X,BL,C,M1)

C   This routine forms an estimate of the equalizer coefficients
C   using the covariance method. The autocorrelation
C   matrix and cross correlation vector are estimated, and
C   the matrix is inverted using Cholesky's method.
C
C   To invert the matrix, NAG library routine F01BPF is used.
C   It inverts a positive definite, Hermetian matrix using
C   Cholesky decomposition. The lower triangle of the matrix
C   is input, and the upper triangle of the inverted matrix
C   is returned.

C   GLOBAL VARIABLES

      INTEGER NFDTAP,NBKTAP
      DOUBLEPRECISION ALPHA
      DOUBLECOMPLEX FDTAP(30),BKTAP(15),REFER,FDCOEF(30),BKCOEF(15)

C   LOCAL VARIABLES

      INTEGER BL
C         -number of symbols in the training block length
      INTEGER M
C         -total number of equalizer coefficients
      INTEGER M1
C         -number of used signal samples occurring before
C         the block
      DOUBLECOMPLEX D(BL)
C         -block of data symbols
      DOUBLECOMPLEX X(-30:2000)
C         -block of received signal samples
      DOUBLECOMPLEX R(46,45)
C         -lower triangle contains estimated autocorrelation
C         matrix, upper triangle contains returns inverted
C         matrix.
      DOUBLECOMPLEX P(45)
C         -estimated cross correlation vector.
      DOUBLECOMPLEX A(45,45)
C         -complete inverted matrix
      DOUBLECOMPLEX C(45)
C         -estimated coefficients
      DOUBLECOMPLEX V(45)
C         -dummy array used by inversion routine
      INTEGER IA
C         -maximum dimension of array, used by inversion routine
      INTEGER IFAIL
C         -error flag used by inversion routine

      COMMON /RECV1/ FDTAP,BKTAP,REFER,FDCOEF,BKCOEF
      COMMON /RECV2/ NFDTAP,NBKTAP,ALPHA

      M=NFDTAP+NBKTAP
      IA=46

```

```

C      Initialize all arrays.
DO 10 I=1,46
    DO 20 J=1,45
        R(I,J)=(0.0,0.0)
20    CONTINUE
10    CONTINUE
DO 15 I=1,45
    DO 17 J=1,45
        A(I,J)=(0.0,0.0)
17    CONTINUE
    P(I)=(0.0,0.0)
    C(I)=(0.0,0.0)
15    CONTINUE

C      Form summations in estimating the autocorrelation
C      matrix and cross correlation vector. Since the matrix
C      is Hermitian, only the lower triangle is formed.

DO 25 N=NBKTAP+1,BL

C      Form lower triangle of matrix partition R11, and
C      first NFDTAP elements of P.
DO 30 I=1,NFDTAP
    DO 40 J=1,I
        R(I+1,J)=R(I+1,J)+X(2*N+1-I)*DCONJG(X(2*N+1-J))
40    CONTINUE
    P(I)=P(I)+X(2*N+1-I)*DCONJG(D(N))
30    CONTINUE

C      Form partition R21 and lower triangle of R22, and
C      remainder of P.
DO 50 I=NFDTAP+1,M
    II=I-NFDTAP
    DO 60 J=1,NFDTAP
        P(I+1,J)=R(I+1,J)+D(N-II)*DCONJG(X(2*N+1-J))
60    CONTINUE
    DO 70 J=NFDTAP+1,I
        JJ=J-NFDTAP
        R(I+1,J)=R(I+1,J)+D(N-II)*DCONJG(D(N-JJ))
70    CONTINUE
    P(I)=P(I)+D(N-II)*DCONJG(D(N))
50    CONTINUE

25    CONTINUE

C      To ensure positive definiteness, set imaginary parts of
C      diagonal elements to zero.
DO 75 I=1,M
    R(I+1,I)=DCMPLX(DREAL(R(I+1,I)),0.000)
75    CONTINUE

C      Call inversion routine.

```

```
IFAIL=0
CALL FO1BPF(M,R,IA,V,IFAIL)
```

```

C      Copy upper triangle of R to upper triangle of A,
C      and conjugate of upper triangle of R to lower
C      triangle of A.
DO 80 I=1,M
    DO 90 J=1,I-1
        A(I,J)=DCONJG(R(J,I))
90      CONTINUE
    DO 100 J=I,M
        A(I,J)=R(I,J)
100     CONTINUE
80      CONTINUE

C      Multiply A by P to get equalizer coefficients.
DO 110 I=1,M
    DO 120 K=1,M
        C(I)=C(I)+A(I,K)*P(K)
120     CONTINUE
110     CONTINUE
```

```

RETURN
END
```

SUBROUTINE OPT(SYMBOL,C)

```

C      This routine computes the optimum equalizer coefficients
C      for a certain instant in time.
C      First, the channel impulse response is found by convolving
C      the pulse shaping filter with the multipath model.
C      Then, the impulse response and noise correlation
C      are used to calculate the exact
C      autocorrelation matrix and cross correlation vector.
C      Finally, the positive definite, Hermetian matrix is
C      inverted using Cholesky's method by calling the NAG
C      library routine F01BPF. This routine accepts the lower
C      triangle of the autocorrelation matrix and returns
C      the upper triangle of the inverse. This is inverse
C      is then multiplied by the cross correlation vector to
C      obtain the optimum coefficients.

C      GLOBAL VARIABLES

DOUBLECOMPLEX DELINE(0:200),CHTAP(10),LPFLIN(0:200)
DOUBLEPRECISION ALPF(0:200),ADGAIN(1),ALPHA
INTEGER MAXDEL,NPATH,MPDEL(10),NSAMP,TIMING,NLPF
INTEGER NFDTAP,NBKTAP
DOUBLECOMPLEX FDTAP(30),BKTAP(15),REFER,FDCOE(30),BKCOEF(30)
DOUBLECOMPLEX NOISLN(0:200)
DOUBLEPRECISION WLPF(0:200),NCOR(0:30),WLPFEN

C      LOCAL VARIABLES

INTEGER SYMBOL
C      -delay of data through channel and forward filter
INTEGER M
C      -total number of coefficients
DOUBLECOMPLEX R(46,45)
C      -lower triangle of Hermetian autocorrelation matrix,
C      and the upper triangle of the inverse matrix
DOUBLECOMPLEX P(45)
C      -cross correlation vector
DOUBLECOMPLEX H(0:200)
C      -impulse response of channel
INTEGER NH
C      -number of samples spanned by impulse response
INTEGER NL
C      -number of symbol intervals spanned by impulse response
INTEGER IH,IH1,IH2,L,K
C      -indices to array of impulse response values, used in
C      computing R
INTEGER I,J
C      -indices to R
DOUBLECOMPLEX A(45,45)
C      -complete inverse matrix
DOUBLECOMPLEX C(45)
C      -optimum equalizer coefficients
DOUBLECOMPLEX V(45)

```

```

C      -dummy array used by inversion routine
INTEGER IA
C      -maximum dimension of matrix, used by inversion routine
INTEGER IFAIL
C      -error flag used by inversion routine

COMMON /CHAN1/ DELINE,CHTAP,MAXDEL,NPATH,MPDEL,NSAMP
COMMON /CHAN3/ LPFLIN,ALPF,ADGAIN,TIMING,NLPP
COMMON /CHAN4/ NOISLN,WLPP,NCOR,WLPFEN
COMMON /RECV1/ FDTAP,BKTAP,REFER,FDCOEF,BKCOEF
COMMON /RECV2/ NFDTAP,NBKTAP,ALPHA

M=NFDTAP+NBKTAP
IA=46

C      Initialize all arrays.
DO 10 I=1,46
    DO 20 J=1,45
        R(I,J)=(0.0,0.0)
20    CONTINUE
10    CONTINUE
    DO 30 I=1,45
        P(I)=(0.0,0.0)
        C(I)=(0.0,0.0)
30    CONTINUE

    DO 40 I=0,200
        H(I)=(0.0,0.0)
40    CONTINUE

C      Compute length of impulse response in samples,
C      and in symbols.
NH=NLPP + MPDEL(NPATH)
NL=NH/NSAMP

C      Convolve pulse shaping filter with multipath.
DO 50 I=0,NH
    DO 60 J=1,NPATH
        IF(I.GE.MPDEL(J)) THEN
            H(I)=H(I)+CHTAP(J)*ALPF(I-MPDEL(J))
        ENDIF
60    CONTINUE
50    CONTINUE

C      Compute autocorrelation matrix and cross correlation
C      vector. I and J are indicies to the matrix, and different
C      samples of the impulse response are used depending on
C      whether I or J is even or odd.

DO 70 I=1,NFDTAP
    DO 80 J=1,I
        K=(I-1)/2
        L=(J-1)/2

```

```

      IF (MOD(I,2).NE.0) THEN
        IF (MOD(J,2).NE.0) THEN
          DO 90 IH=0,NL
            IH1=NSAMP*IH
            IH2=NSAMP*(IH+K-L)
            R(I+1,J)=R(I+1,J)+H(IH1)*D*CONJG(H(IH2))
          90 CONTINUE
            R(I+1,J)=R(I+1,J)+NCOR(2*(K-L))
        ELSE
          DO 100 IH=0,NL
            IH1=NSAMP*IH
            IH2=NSAMP*(IH+K-L)-NSAMP/2
            R(I+1,J)=R(I+1,J)+H(IH1)*D*CONJG(H(IH2))
          100 CONTINUE
            R(I+1,J)=R(I+1,J)+NCOR(2*(K-L)-1)
        ENDIF
        P(I)=H(NSAMP*(SYMDEL-K))
      ELSE
        IF (MOD(J,2).NE.0) THEN
          DO 110 IH=0,NL
            IH1=NSAMP*IH-NSAMP/2
            IH2=NSAMP*(IH+K-L)
            R(I+1,J)=R(I+1,J)+H(IH1)*D*CONJG(H(IH2))
          110 CONTINUE
            R(I+1,J)=R(I+1,J)+NCOR(2*(K-L)+1)
        ELSE
          DO 120 IH=0,NL
            IH1=NSAMP*IH-NSAMP/2
            IH2=NSAMP*(IH+K-L)-NSAMP/2
            R(I+1,J)=R(I+1,J)+H(IH1)*D*CONJG(H(IH2))
          120 CONTINUE
            R(I+1,J)=R(I+1,J)+NCOR(2*(K-L))
        ENDIF
        P(I)=H(NSAMP*(SYMDEL-K)-NSAMP/2)
      ENDIF
    80 CONTINUE
    70 CONTINUE

    DO 130 I=NFDOTAP+1,M
      K=I-NFDOTAP
      DO 140 J=1,NFOTAP
        L=(J-1)/2
        IF (MOD(J,2).NE.0) THEN
          IH=NSAMP*(SYMDEL+K-L)
          R(I+1,J)=D*CONJG(H(IH))
        ELSE
          IH=NSAMP*(SYMDEL+K-L)-NSAMP/2
          R(I+1,J)=D*CONJG(H(IH))
        ENDIF
      140 CONTINUE
      DO 150 J=NFDOTAP+1,M
        IF (I.EQ.J) R(I+1,J)=(1.0,0.0)

```

```

150         CONTINUE
130     CONTINUE

C         To ensure positive definiteness, set imaginary parts
C         of diagonals to zero.
DO 160 I=1,M
    R(I+1,I)=DCMPLX(DREAL(R(I+1,I)),0.000)
160     CONTINUE

IFAIL=0
CALL F01BPF(M,R,IA,V,IFAIL)

C         Form complete inverse matrix in A.
DO 270 I=1,M
    DO 280 J=1,I-1
        A(I,J)=DCONJG(R(J,I))
280     CONTINUE
    DO 290 J=1,M
        A(I,J)=R(I,J)
290     CONTINUE
270     CONTINUE

C         Multiply inverse by P to get coefficients.
DO 300 I=1,M
    DO 310 J=1,M
        C(I)=C(I)+A(I,J)*P(J)
310     CONTINUE
300     CONTINUE

RETURN
END

```

APPENDIX E

LIST OF MAIN PROGRAM FOR SIMULATION OF BLOCK BY
BLOCK WITH INTERPOLATION


```

C      This main program calls the routines for initialization
C      and channel simulation. It also calls the routine for
C      estimation of equalizer coefficients, and the routine
C      for interpolation of coefficients and equalization of
C      the channel over the data block.
C      In this program many blocks are transmitted through
C      the simulated channel and received by the equalizer.
C      Using a delayed version of the transmitted data stream,
C      the number of errors is determined. Additionally,
C      the distribution of the number of blocks versus
C      the number of errors per block is found.

C      Subroutines used: INIT,DATASYM,HFSIM,SHIFT,CORREL,
C                        LINEQ

C      GLOBAL VARIABLES

      DOUBLECOMPLEX DELINE(0:200),CHTAP(10),AIIR(10,4)
      @DOUBLECOMPLEX LPFLIN(0:200),NOISLN(0:200)
      DOUBLECOMPLEX FDTAP(30),BK TAP(15),REFER,FDCOE(30),BKCOEF(15)
      DOUBLEPRECISION COEF(3),MPGAIN(10),SQENRG,ALPF(0:200)
      DOUBLEPRECISION ADGAIN(1),NCOR(0:30),WLPFEN,ALPHA,WLPF(0:200)
      INTEGER MAXDEL,NPATH,MPDEL(10),NSAMP,NFDTAP,NBK TAP
      INTEGER TIMING,NLPF

C      LOCAL VARIABLES

      INTEGER SYMDEL
C      -symbol delay through pulse shaping filter and
C      forward filter of equalizer
      INTEGER IN,INBK,IX
C      -counters for loops and arrays
      INTEGER NBK
C      -number of blocks transmitted
      INTEGER TBL,DBL
C      -training block length and data block length
      INTEGER M1
C      -number of symbol intervals from previous block
C      to be included in covariance method
      INTEGER ERPTOT
C      -total number of errors
      INTEGER ERRCNT
C      -number of errors in a block
      INTEGER IMPCNT
C      -used to print error distribution
      INTEGER ERDIST(0:1000)
C      -error distribution, each element is the number of
C      blocks with a number of errors equal to the index
C      of the array
      DOUBLECOMPLEX DATA
C      -data symbol
      DOUBLECOMPLEX CHOUT(2)
C      -T/2 spaced output samples of channel
      DOUBLECOMPLEX REF(1000)

```

```

C      -training block of transmitted data symbols
DOUBLECOMPLEX X(-30:2000)
C      -stored signal samples for training block
DOUBLECOMPLEX DATREF(1000)
C      -data block of transmitted data symbols
DOUBLECOMPLEX DATX(2000)
C      -stored signal samples for data block
DOUBLECOMPLEX ESTIM(1000)
C      -block of data decisions
DOUBLECOMPLEX DATBUF(50)
C      -buffer for delaying transmitted data
DOUBLECOMPLEX C1(45),C2(45)
C      -estimated coeffs at beginning and end of block
CHARACTER * 10 PARAM
C      -name of input file

COMMON /CHAN1/ DELINE,CHTAP,MAXDEL,NPATH,MPDEL,NSAMP
COMMON /CHAN2/ AIIR, COEF, MPGAIN, SQENRG
COMMON /CHAN3/ LPFLIN,ALPF, ADGAIN, TIMING,NLPP
COMMON /CHAN4/ NOI$LN,WLPP,NCOR,WLPFEN
COMMON /RECV1/ FDTAP,BKTAP,REFER,FDCOEF,BKCOEF
COMMON /RECV2/ NFDTAP,NBKTAP,ALPHA

C      *
C      Read name of input file, number of blocks, and block
C      lengths. Initialize arrays and parameters.
READ*,PARAM
READ*,NBK,TBL,DBL;
WRITE(99,*) 'FILE: ',PARAM
WRITE(99,*) 'NBK,TBL,DBL: ',NBK,TBL,DBL

ERRTOT=0
DO 3 I=0,1000
    ERDIST(I)=0
3  CONTINUE

DO 5 I=1,50
    DATBUF(I)=(0.0,0.0)
5  CONTINUE

CALL INIT(PARAM)
SYMDEL= NLPP/(NSAMP*2) + NFDTAP/2 -1
M1=MAX(0,2*((NFDTAP/2)-NBKTAP-1)-1)

C      Run simulation to fill up delay lines in channel
C      and equalizer. Also, create signal samples that
C      occur before first training block for use by
C      covariance method.
IX=-M1
DO 10 IN=1,4*SYMDEL

C      This part creates a random data symbol and gets
C      two T/2 samples from the channel simulator.
C      Also, the data buffer is shifted and the new

```

```

C      symbol entered. This is done to match the
C      delay of the reference data to that of the
C      channel and equalizer.
      CALL DATASYM(DATA)
      CALL HFSIM(DATA,CHOUT)
      DO 8 I=SYMDEL+1,2,-1
        DATBUF(I)=DATBUF(I-1)
9      CONTINUE
      DATBUF(1)=DATA

C      This shifts the channel output samples into the
C      equalizer, and shifts the feedback filter.
C      REFER is the current symbol being decided and
C      will be shifted into the feedback filter next
C      interval.
      CALL SHIFT(CHOUT(1),CHOUT(2))
      REFER=DATBUF(SYMDEL+1)

C      This stores signal samples occurring before
C      the first training block, but may be needed
C      in the first coefficient estimation.
      IF(IN.GT.4*SYMDEL-(M1+1)/2) THEN
        X(IX)=CHOUT(1)
        IX=IX+1
        X(IX)=CHOUT(2)
        IX=IX+1
      ENDIF
10     CONTINUE

C      Before the main loop is entered, the coefficients
C      at the beginning of the first data block are estimated.
C      These, with those at the end of the block, will be
C      used in interpolation.

C      First, the transmission of a training block is simulated,
C      and the samples are used in CORREL to estimate the
C      coefficients.
      DO 20 IN=1,TBL
        CALL DATASYM(DATA)
        CALL HFSIM(DATA,CHOUT)
        DO 15 I=SYMDEL+1,2,-1
          DATBUF(I)=DATBUF(I-1)
15       CONTINUE
          DATBUF(1)=DATA
          CALL SHIFT(CHOUT(1),CHOUT(2))
          REFER=DATBUF(SYMDEL+1)
          REF(IN)=DATBUF(SYMDEL+1)
          X(2*IN-1)=CHOUT(1)
          X(2*IN)=CHOUT(2)
20      CONTINUE
      CALL CORREL(REF,X,TBL,C1,M1)

C      This is the main loop for transmission of blocks.
C      Since the first coefficients have been estimated, C1,

```

C It starts with the transmission of a data block.
 C At the end of the block the coefficients C2 are estimated
 C and C1 and C2 are used in LINEQ to equalize the channel
 C with coefficients that are linearly interpolated between
 C C1 and C2. After this, the coefficients in C2 are moved
 C into C1 to prepare for the next block.

C The array ESTIM is compared to DATREF to determine the
 C number of errors in the block, and the total number of
 C errors is accumulated. The number of errors in the data
 C block is used as an index to increment an element in
 C ERDIST.

DO 100 INBK=1,NBK

C Data block.

DO 30 IN=1,DBL

CALL DATASYM(DATA)

CALL HFSIM(DATA,CHOUT)

DO 25 I=SYMDEL+1,2,-1

DATBUF(I)=DATBUF(I-1)

25

CONTINUE

DATBUF(1)=DATA

DATREF(IN)=DATBUF(SYMDEL+1)

DATX(2*IN-1)=CHOUT(1)

DATX(2*IN)=CHOUT(2)

30

CONTINUE

C Store samples previous to next training block
 C that may be needed in covariance method.

DO 31 I=-M1,0

X(I)=DATX(2*NBK+1)

31

CONTINUE

C Training block.

DO 40 IN=1,TBL

CALL DATASYM(DATA)

CALL HFSIM(DATA,CHOUT)

DO 35 I=SYMDEL+1,2,-1

DATBUF(I)=DATBUF(I-1)

35

CONTINUE

DATBUF(1)=DATA

REF(IN)=DATBUF(SYMDEL+1)

X(2*IN-1)=CHOUT(1)

X(2*IN)=CHOUT(2)

40

CONTINUE

CALL CORREL(REF,X,TBL,C2,M1)

CALL LINEQ(DATREF,DATX,DBL,ESTIM,C1,C2)

ERRCNT=0

DO 45 I=1,DBL

IF(ESTIM(I).NE.DATREF(I)) THEN

SUBROUTINE SHIFT(CHOUT1,CHOUT2)

```

CH(1)=CHOUT1
CH(2)=CHOUT2
DO 10 K=1,2
    DO 20 J=1,NFDTAP-1
        FDTAP(NFDTAP-J+1)=FDTAP(NFDTAP-J)
20    CONTINUE
        FDTAP(1)=CH(K)
10    CONTINUE
    DO 30 J=1,NBKTAP
        BKTAP(NBKTAP-J+1)=BKTAP(NBKTAP-J)
30    CONTINUE
    BKTAP(1)=REFER
    RETURN
END

```

SUBROUTINE LINEQ(DATA,X,BL,ESTIM,C1,C2)

```

C      This routine accepts as input the data block of received
C      samples and reference symbols, as well as the estimated
C      coefficients for each end of the block.
C      At each symbol interval within the block, equalizer
C      coefficients are calculated by linearly interpolating
C      between the coefficients at the ends of the block.
C      These are then used to form the equalizer output, and
C      the decision is stored in ESTIM.

C      Routine DECIDE is used.

C      GLOBAL VARIABLES
DOUBLECOMPLEX FDTAP(30),FDCOEF(30),REFER,BKTAP(15),BKCOEF(15)
DOUBLEPRECISION ALPHA
INTEGER NFDTAP,NBKTAP

C      LOCAL VARIABLES

INTEGER BL
C      -data block length
INTEGER IB
C      -counter
DOUBLECOMPLEX X(2*BL)
C      -block of received signal samples
DOUBLECOMPLEX DATA(BL)
C      -block of reference data symbols
DOUBLECOMPLEX STEP(45)
C      -increment applied to coefficients at each symbol
C      interval to achieve interpolation
DOUBLECOMPLEX ESTIM(BL)
C      -block of receiver decisions
DOUBLECOMPLEX EQOUT
C      -equalizer output
DOUBLECOMPLEX C1(45),C2(45)

```

```

C      -coefficients at each end of block .

COMMON /RECV1/ FDTAP,BKTAP,REFER,FDCOEFF,BKCOEF
COMMON /RECV2/ NFDTAP,NBKTAP,ALPHA

C      Calculate increment to coefficient vector to
C      achieve interpolation.
DO 150 J=1,NFDTAP
    FDCOEFF(J)=C1(J)
    STEP(J)=(C2(J)-C1(J))/BL
150 CONTINUE
DO 160 J=1,NBKTAP
    JJ=J+NFDTAP
    BKCOEF(J)=C1(JJ)
    STEP(JJ)=(C2(JJ)-C1(JJ))/BL
160 CONTINUE

C      For each symbol in the data block, increment coefficient
C      vector by STEP, and form equalizer output. Then call
C      DECIDE to get the decision and store it in ESTIM.
C      Update REFER with the decision so that decisions are
C      used in feedback filter.
DO 70 IB=1,BL

    CALL SHIFT(X(2*IB-1),X(2*IB))

    EQOUT=(0.0,0.0)
    DO 110 J=1,NFDTAP
        FDCOEFF(J)=FDCOEFF(J)+STEP(J)
        EQOUT=EQOUT+FDTAP(J)*DCONJG(FDCOEFF(J))
110 CONTINUE
    DO 120 J=1,NBKTAP
        BKCOEF(J)=BKCOEF(J)+STEP(J+NFDTAP)
        EQOUT=EQOUT+BKTAP(J)*DCONJG(BKCOEF(J))
120 CONTINUE

    CALL DECIDE(EQOUT,ESTIM(IB))
    REFER=ESTIM(IB)

70 CONTINUE

RETURN
END

SUBROUTINE DATASYM(DATA)

C      Routine to generate a random complex number
C      where the symbols  $\exp(j*k*\pi/4)$ ,  $k=1,3,5,7$ 
C      are equally likely.

C      Uses NAG library routine G05CAF
C      which returns a double precision number uniformly
C      distributed over (0,1).

```

```
DOUBLECOMPLEX DATA
DOUBLEPRECISION PARAM,X
DOUBLEPRECISION DUMMY
```

```
PARAM = 1. / DSQRT(2.000)
X=G05CAF(DUMMY)
IF (X .LT. 0.25) THEN
  DATA=DCMP(X(PARAM,PARAM)
ELSE IF (X .LT. 0.5) THEN
  DATA=DCMPLX(-PARAM,PARAM)
ELSE IF (X .LT. 0.75) THEN
  DATA=DCMPLX(-PARAM,-PARAM)
ELSE IF (X .LT. 1.0) THEN
  DATA=DCMPLX(PARAM,-PARAM)
END IF
RETURN
END
```

```
SUBROUTINE DECIDE(IN,OUT)
```

```
C Subroutine to quantize a received sample to the
C nearest transmitted level.
```

```
DOUBLECOMPLEX IN,OUT
DOUBLEPRECISION PARAM,OUTR,OUTI
```

```
-PARAM=1./DSQRT(2.000)
IF (DREAL(IN) .GE. 0.0) THEN
  OUTR=PARAM
ELSE IF (DREAL(IN) .LT. 0.0) THEN
  OUTR=-PARAM
ENDIF
IF (DIMAG(IN) .GE. 0.0) THEN
  OUTI=PARAM
ELSE IF (DIMAG(IN) .LT. 0.0) THEN
  OUTI=-PARAM
ENDIF
OUT=DCMPLX(OUTR,OUTI)
RETURN
END
```


END

0|9|0|8|8|7

FIN