THE UNIVERSITY OF ALBERTA

EVALUATION OF DIGITAL SUBSCRIBER LOOP CAPABILITIES FOR BIT RATE ADAPTIVE TRANSMISSION

by

YOUNG-FEI SHEN

©

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL ENGINEERING

EDMONTON, ALBERTA

SPRING 1990

NOTICE

AVIS

ISBN 0-315-60187-6

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR:	YOUNG-FEI SHEN
TITLE OF THESIS:	EVALUATION OF DIGITAL SUBSCRIBER LOOP CAPABILITIES FOR BIT RATE ADAPTIVE TRANSMISSION
DEGREE:	MASTER OF SCIENCE
YEAR THIS DEGREE GRANTED:	SPRING 1990

(SIGNED) ...Shen Young Fei...........

PERMANENT ADDRESS:

Flat A, 17th Floor
Beverley Heights, Belair Garden
Shatin, Hong Kong

DATE: ..Jan..30.....1990

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled EVALUATION OF DIGITAL SUBSCRIBER LOOP CAPABILITIES FOR BIT RATE ADAPTIVE TRANSMISSION submitted by YOUNG-FEI SHEN in partial fulfilment of the requirements for the degree of MASTER of SCIENCE.

Supervisor

Keith A. Stromsmoe

Date: 2 Jan. 1990

ABSTRACT

This thesis investigates the possible improvements in the utilization of subscriber loop plant due to the deployment of adaptive bit rate transceivers. This is achieved through computer simulation of the multi-rate transceiver working in the presence of residual echoes, near-end crosstalk, and coloured Gaussian noise on a set of 35 loops ranging from 0-10.7 km. The results show that it is possible to achieve bit rates up to 1.5 Mbit/s over loops up to 1.5 km long and with bit rates reduced to 160 kbit/s a reach of 10.7 km is achievable. Depending on the subscriber population, the potential capacity of the rate-adaptive transmission is 500-700% greater than the capacity available for the ISDN basic access.

ACKNOWLEDGEMENTS

I would like to thank my supervisor, Dr. Witold Krzymien, for his guidance and many fruitful discussions, and Dr. Wayne Grover for the initial idea of the project. In addition, I would like to extend my sincere thanks to the following:

- the members of the examining committee.
- the staff of Alberta Telecommunications Research Centre.
- Alberta Telecommunications Research Centre for the award of the ATRC Telecommunications Scholarship.
- University of Alberta for the award of the Province of Alberta Graduate Scholarship.
- Mr. Tom Bilodeau for his fruitful discussions.

Last but not least, my family and friends for their constant support and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ANSI	American National Standards Institute
AWG	American wire gauge
BAI	Basic Access Interface
BER	Bit error rate
CCITT	International Telegraph and Telephone Consultative Committee
CO	Central office
DFE	Decision feedback equalizer
EC	Echo canceller
FEXT	Far-end crosstalk
FFT	Fast Fourier transform
FSE	Fractionally spaced equalizer
IFFT	Inverse fast Fourier transform
ISDN	Integrated Services Digital Network
ISI	Intersymbol interference
LMS	Least mean-squares
LSI	Large scale integration
LT	Line termination
MR	Multiple response
MSE	Mean-squared error
NEXT	Near-end crosstalk
NT	Network termination
PAI	Primary Access Interface
PAM	Pulse amplitude modulation
PBX	Private branch exchange
PCM	Pulse code modulation

PIC	Polyethelene insulated cable
PSD	Power spectral density
QSNR	Quantiser signal to noise ratio
RA-DSL	Rate-adaptive digital subscriber loop
RLS	Recursive least squares
SG	Stochastic gradient
SNR	Signal to noise ratio
SUB	Subscriber
TCM	Time-compression multiplexing
ZF	Zero-forcing

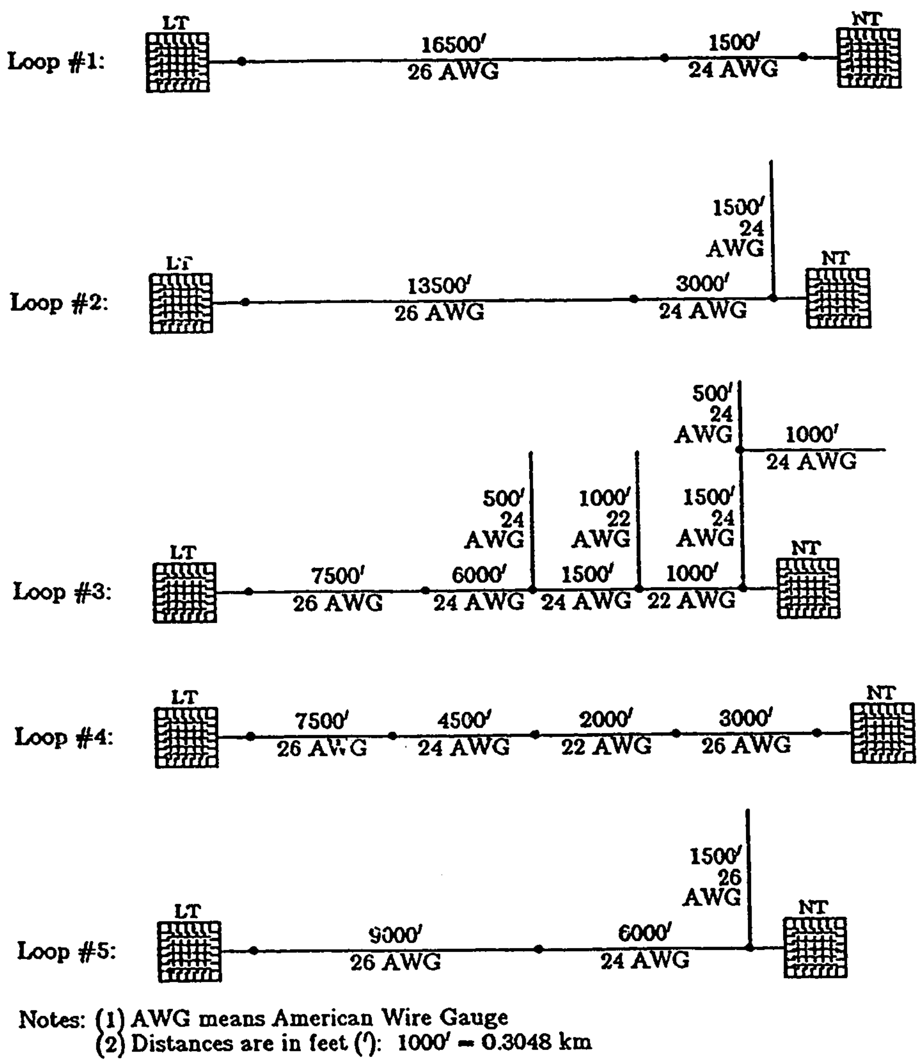
Chapter 1

INTRODUCTION

1.1 Overview of the Integrated Services Digital Network concepts

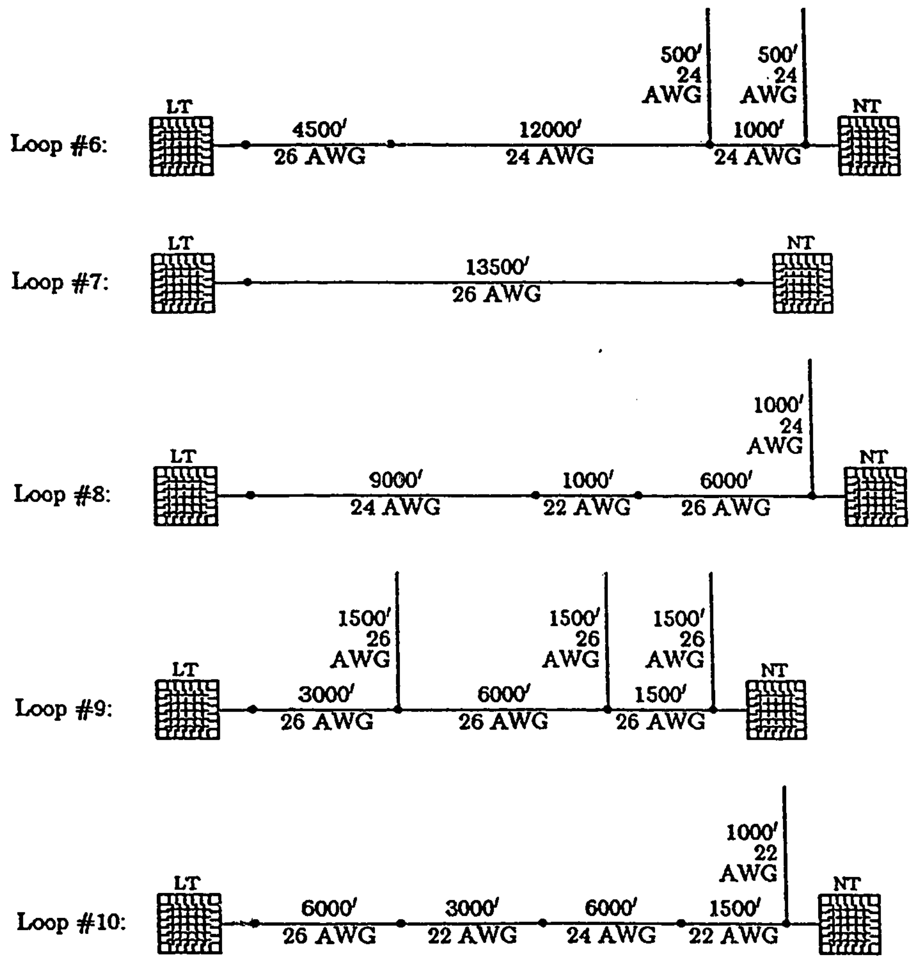
The telephone network is made up of the core network and the loop plant. The core network includes all the switching offices, toll trunks... etc, and the loop plant includes the wireline transmission paths between the subscribers and the central offices. The transmission over the subscriber loops is analog, though most newly installed telephone switches use digital transmission within a central office. When digital data are to be transmitted over the loop, a modem is needed to convert the digital data to analog signal for transmission and vice versa. Data communication now accounts for more than 10% of the network traffic in North America, and it is increasing at the rate of 30% a year. Data transmission through modems is inefficient. During the past few years, telephone companies throughout the world have been working towards an all digital network — Integrated Services Digital Network (ISDN). The principle of the ISDN is to provide full duplex digital services (voice or non-voice) to all subscribers using the existing subscriber loop plant.

The ISDN is intended to provide 99% coverage of the North-American non-loaded loop population, comprising loops of length up to 5.5 km. The bit error rate requirement is 10^{-7}, and a set of 15 loop configurations is defined by the American National Standards Institute (ANSI) for testing the transceiver performance [1]. These loops are shown in Fig. 1.1. The reason for ISDN coverage of non-loaded loop population is understandable. When phone lines are primarily used for voice



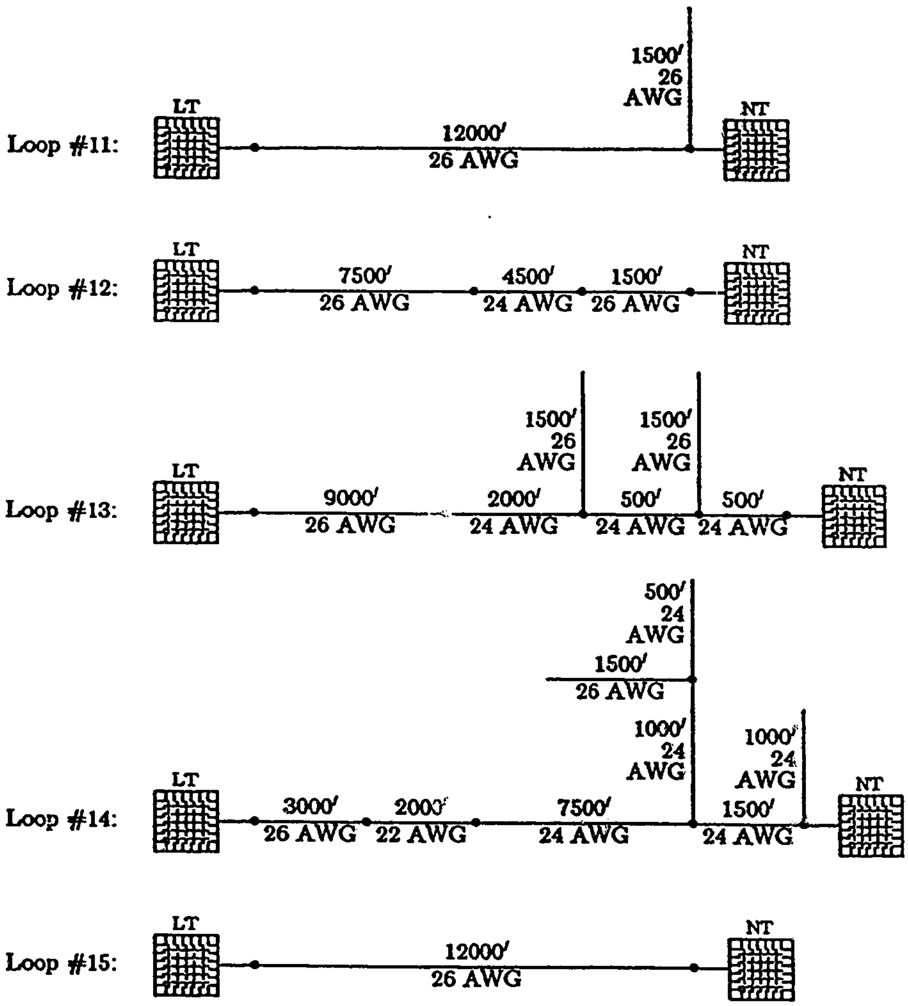
(a) Loops #1 - #5

Fig. 1.1 ANSI standard loops for testing received signal performance



Notes: (1) AWG means American Wire Gauge
(2) Distances are in feet (′): 1000′ = 0.3048 km

(b) Loops #6 - #10



Notes: (1) AWG means American Wire Gauge
(2) Distances are in feet (′): 1000′ = 0.3048 km

(c) Loops #11 - #15

communications, loading coils are usually added to long or defective loops to enhance their response at higher frequencies (above 1.5 kHz). High bit rate transmission is not possible on these loaded loops because of the dramatic increase of their attenuation above 4 kHz. In order to prevent the expensive process of loop conditioning, ISDN coverage is limited to non-loaded loops which are shorter than 5.5 km.

Two major physical layer interfaces for the ISDN are specified by the International Telegraph and Telephone Consultative Committee (CCITT) standards: the basic access interface (BAI) and the primary access interface (PAI) [1]. The basic access interface provides two 64 kbit/s full duplex B channels and one 16 kbit/s D channel. Each B channel can be used as a pulse code modulated (PCM) voice channel or as a 64 kbit/s data channel. The 16 kbit/s D channel will be used for network control and for low-speed data transmission. Another 16 kbit/s of overhead is added for framing and maintenance, which makes the transmission rate equal to 160 kbit/s. The PAI supports 23B+D in North America and Japan, and 30B+D in other countries. The B channels and the D channel for the PAI are both 64 kbit/s.

The reference points at which the standard interfaces are defined are shown in the reference configuration in Fig. 1.2. The subscriber loop connects the subscribers to the central office, and the line interface is named network termination (NT) at the subscriber end and line termination (LT) at the central office end. These two interface points are commonly known as the U interface. There is a set of electrical characteristics that are specified at the U interface [1], and they are difficult to implement because of the environmental conditions and duplex transmission on a single pair of wires. This

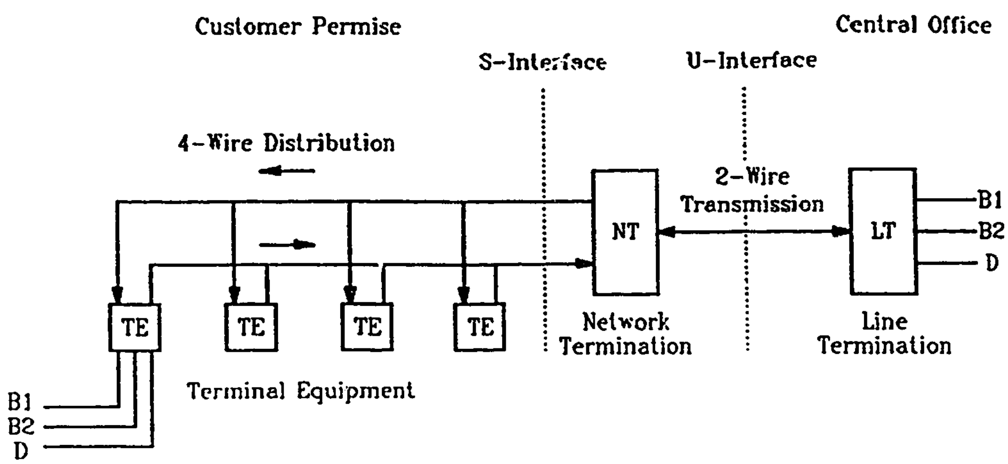


Fig. 1.2 The standard interfaces of the ISDN digital subscriber loop

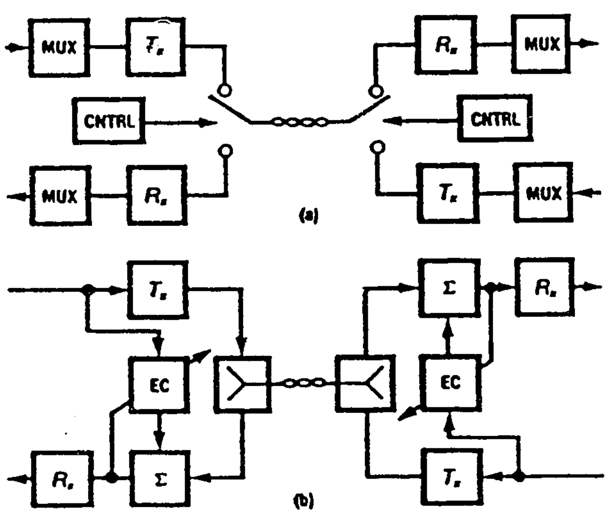


Fig. 1.3 The block diagram of (a) time-compression muliplexing (TCM)
(b) echo canceller full duplex transmission method (EC)

thesis deals with the transmission between the subscribers and the central offices; therefore, only the U interface will be discussed.

There are two practical methods for a single twisted wire pair full duplex digital transmission: the burst synchronized time compression multiplexing (TCM) and the echo cancellation method (EC). The block diagrams for the two transmission methods are shown in Fig. 1.3. In the TCM system, transmission in the two directions is separated in time, and blocks of bits are sent alternately in each direction. Due to the propagation delay of the line and guard time for the line to settle down, the transmission rate in each direction must be approximately 2.3 times higher than the resulting duplex rate. This technique is attractive because of the reduced complexity of the transceiver design. It is widely used in short range transmission for private branch exchange (PBX) applications, but it shows shortcomings in long subscriber loops due to higher attenuation and noise at a higher transmission rate. In the EC system, transmission rate does not have to increase as transmission in both directions occurs simultaneously. This is achieved by using a hybrid circuit and an echo canceller at each transceiver to separate the two directions of transmission (hybrid circuit and echo canceller will be discussed in chapter 2). This approach increases hardware complexity. However, the fast development of integrated-circuit technology has rapidly reduced the cost of extra hardware, and large scale integration (LSI) of the EC transceiver has been demonstrated to be feasible. The EC method has now been accepted by the ANSI to be used on the ISDN loops.

1.2 RA-DSL transmission method

The channel capacity estimates of subscriber lines indicate that the capacity varies considerably with distance, and for short loops significantly exceeds 1 Mbit/s [2]. Consequently, ubiquitous deployment of basic access ISDN at its fixed rate of 160 kbit/s will imply considerable under-utilization of the subscriber loop plant. The recently proposed high bit rate digital subscriber access at 800 kbit/s over loops within the carrier serving area [3] will improve the situation, but still will not utilize the transmission potential of individual subscriber loops to their fullest extent. The proposed rate-adaptive digital subscriber loop (RA-DSL) transmission method may be a solution to the problem [4].

The concept of the RA-DSL transmission method is as follows: instead of limiting the transmission rate over a class of loops due to a small population of low capacity loops, we can adaptively maximize the transmission rate of individual loops depending on their capacity. This approach can provide a significant increase in the utilization of the existing copper access network, while enabling some new ancillary applications such as the long reach 1B+D service. The RA-DSL method was impractical prior to the onset of the ISDN because there was no workable way to utilize the extra capacity due to the non-channelized, in-band signalling protocols. The RA-DSL method is implemented through the use of a multi-rate transceiver and a control protocol for bit rate maximization. A prototype with a simplified transceiver (without echo cancellation and using simplified linear adaptive equalization) and the rate adaptation protocol has been implemented in hardware and demonstrated [5,6].

1.3 Objective and organization of the thesis

The objective of this thesis is to investigate the possible improvements in the utilization of subscriber loop plant due to the deployment of adaptive bit rate transceivers. This is achieved through computer simulation of the multi-rate transceiver working in a realistic subscriber loop environment. Instead of implementing the control protocol for bit-rate maximization, the maximization process is performed manually, because joint simulation of the control protocol and transceiver signal processing is computationally infeasible. A group of 35 loops ranging from 0-10.7 km will be made up using the Bell system loop statistics data [7] to represent the whole loop plant in North America. The performance of these loops is measured by the bit error rate, quantiser signal to noise ratio and the eye opening. Finally, quantitative analysis will be performed to estimate the utilization gain of the RA-DSL transmission method.

In chapter 2, an overview of the North-American loop plant is presented. The disturbances inherent in the loop plant that degrade the transmission performance will follow.

In chapter 3, methods of improving the transmission performance are introduced. The use of adaptive equalization utilizing the least mean-square algorithm is emphasized.

In chapter 4, the structure and algorithms of the simulation programs are discussed.

In chapter 5, the simulation results using two different transceiver structures on some selected loops are discussed.

Quantitative analysis to estimate the utilization gain is also included.

In chapter 6, a summary of the entire research project is presented. Some suggestions on future research work are made.

CHAPTER 2

BACKGROUND AND SYSTEM OVERVIEW

This chapter will describe the general characteristics of the digital subscriber loop, and the disturbances inherent in the transmission environment. In order to study the feasibility of the RA-DSL transmission method in North America, we must rely on the statistics of the North American loop plant. As a result, the characteristics and statistics of the subscriber loops will be presented. On the other hand, an understanding of the disturbances present in the loops will provide hints to combat them, thus they will also be discussed.

2.1 Subscriber loop characteristics

Bell Canada performed a loop survey in 1981 to study the physical and electrical characteristics of the local network [7]. The loops are mainly made up of 19, 22, 24, and 26 AWG (American Wire Gauge) wires (0.912 mm, 0.645 mm, 0.511 mm and 0.404 mm respectively). Accordingly, the amount of attenuation and dispersion is more serious in the 26 gauge wire than in the 19 gauge wire. Fig. 2.1 shows the gauge distribution as a function of distance from the central office. The 26 AWG wire dominates at short distances and is gradually replaced by the 24 AWG, the 22 AWG and 19 AWG wire as the length of the loop increases. This is to reduce the loop attenuation while addressing duct congestion near the central office. As a design rule, the DC resistance of the loops is kept below 1300 ohms. The average loop length is 3.102 km, and only 0.7% of the unloaded loops are in excess of 5.5 km.

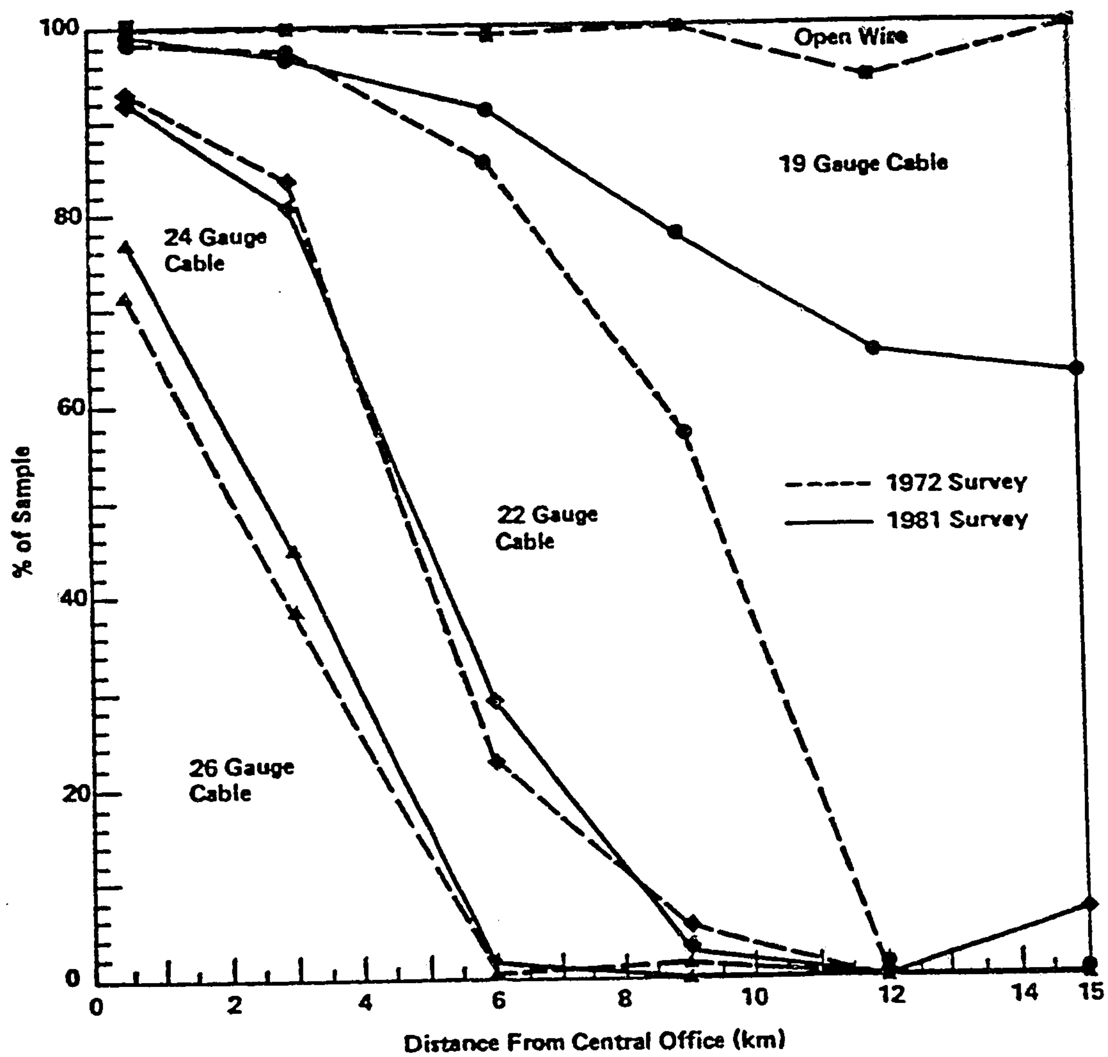


Fig. 2.1 Gauge distribution with distance from central office

A bridged tap is simply an open-circuit wire pair connected at a point along the main line and electrically in parallel to it. These bridged taps allow the telephone network to reach geographically dispersed locations economically, and increase the fill in the loop plant (fill is the ratio of the number of working pairs to the total number of pairs installed). The average length of a bridged tap is 0.286 km, and the total length of bridged taps on a single unloaded loop is less than 1.83 km in 99.4% cases.

An actual loop may consist of wires of different gauges and lengths, and bridged taps branched at different locations; but basically the frequency response of the loop is low pass in nature. For high data rate transmission, the attenuation at high frequencies limits the maximum span of transmission. Bridged taps will not only cause additional attenuation due to loading, but they will also cause a reflected pulse that is delayed relative to the main pulse and causes postcursor intersymbol interference. When the length of a bridged tap approaches a quarter wavelength, the main line will see it as a short-circuit and the amount of attenuation will increase sharply. This will be a problem at high data rate transmission as many bridged taps may become quarter wavelength ones.

2.2 Disturbances affecting transmission

2.2.1 Crosstalk

Crosstalk is defined as the disturbance created in one communication circuit by the signals in other communication circuits. In telephony, the metallic wires are bundled into a larger cable, and crosstalk is created by the capacitive coupling of the signal on one wire pair to another. There are two kinds of crosstalk, near-end crosstalk (NEXT) and far-end crosstalk (FEXT). Both kinds of crosstalk are illustrated in Fig. 2.2. Near-end crosstalk represents a crosstalk of a local transmitter into a local receiver, and its power transfer function is modelled by [8]:

$$|H_{NEXT}(f)|^2 = K_{NEXT} * f^{1.5} \qquad (2.1)$$

where K_{NEXT} is the near-end crosstalk coupling constant.
Far-end crosstalk represents a crosstalk of a local transmitter into a remote receiver and its power transfer function is modelled by [8]:

$$|H_{FEXT}(f)|^2 = K_{FEXT} * f^2 * L \qquad (2.2)$$

where K_{FEXT} is the far-end crosstalk coupling constant, and L is the length of the coupling path.

Both NEXT and FEXT experience less attenuation as frequencies increase. Therefore, it is important to limit the power in the high frequencies of the transmitted signal spectrum. This can be accomplished by a proper choice of line code. FEXT is shown to be less significant (chapter 4) comparing to NEXT, and NEXT is the limiting factor for the EC transmission method in the digital subscriber loop.

2.2.2 Echo

In the EC system, transmission in both directions happens simultaneously, and a hybrid transformer is needed to separate the two directions of transmission. Fig. 2.3 shows the structure of an electronic hybrid with its balance network. If the balance impedance of the hybrid is matched to the line, none of the transmitted signal will

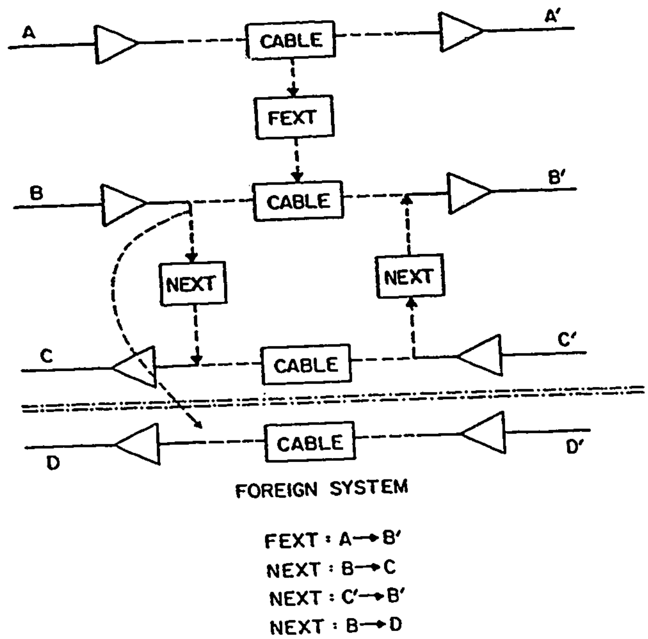


Fig. 2.2 Near-end crosstalk and far-end crosstalk

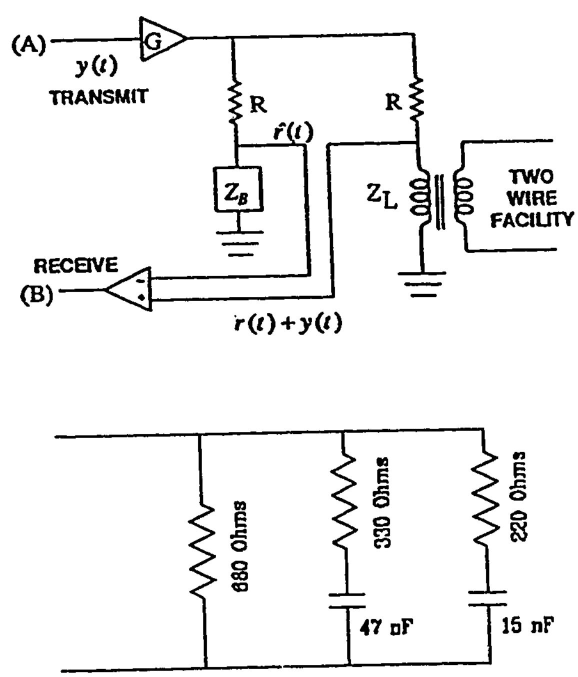


Fig. 2.3 An electronic hybrid with the balance network

be coupled to the receive port of the hybrid, and half of the transmitted power will be coupled to the line. This situation rarely happens because the hybrid circuit is designed with the assumption that neither the length, gauges, nor the presence of bridged taps are known, thus the balance network is selected as a compromise match to all possible loop configurations. The impedance mismatch causes echo as the transmit signal leaks into the receive port of the hybrid. The performance of the hybrid is measured by the echo return loss, and it is infinitely large when the line and the balance impedance are matched.

$$\text{Echo Return Loss (dB)} = 20\log_{10}\left|\frac{(Z_L+R)(Z_B+R)}{R*(Z_L-Z_B)}\right| \qquad (2.3)$$

R is the load impedance, Z_B is the balance impedance, while Z_L is the line input impedance. A typical hybrid transformer will provide 10 dB echo cancellation. In order to provide a 25 dB signal-to-echo noise ratio on a loop with 40 dB attenuation, another 55 dB echo cancellation is required. The extra echo cancellation required is provided by the echo canceller.

A block diagram of a linear echo canceller is shown in Fig. 2.4. An echo canceller can be implemented as an adaptive transversal filter that learns the response of the echo path. Since the symbols transmitted at the local transceiver are known, an accurate replica of the echo can be formed. This replica is subtracted from the received signal to yield an echo-free signal. The tap coefficients are adjusted by the least mean-square (LMS) algorithm, which can be expressed as:

$$\vec{C}_{(n+1)} = \vec{C}_n + \mu e_n \vec{Y}_n \qquad (2.4)$$

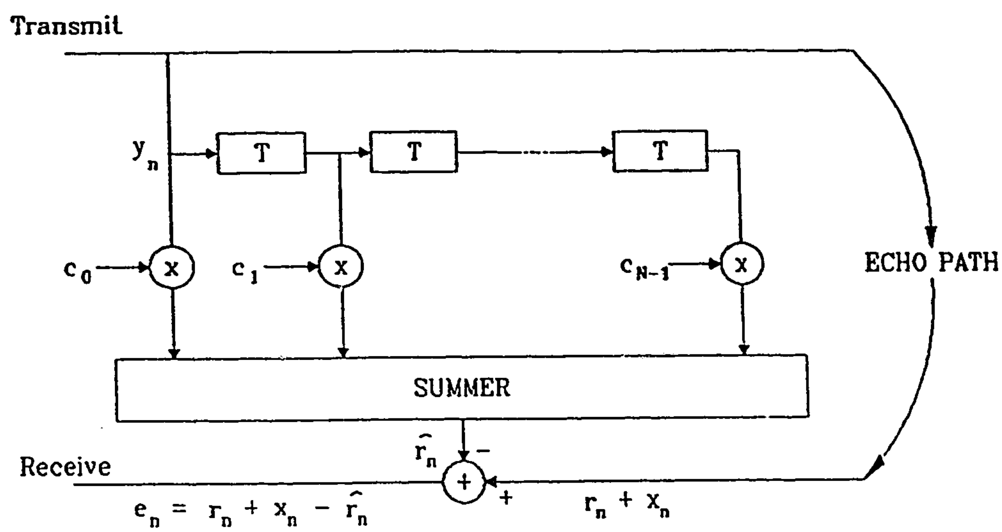


Fig. 2.4 A linear transversal filter echo canceller

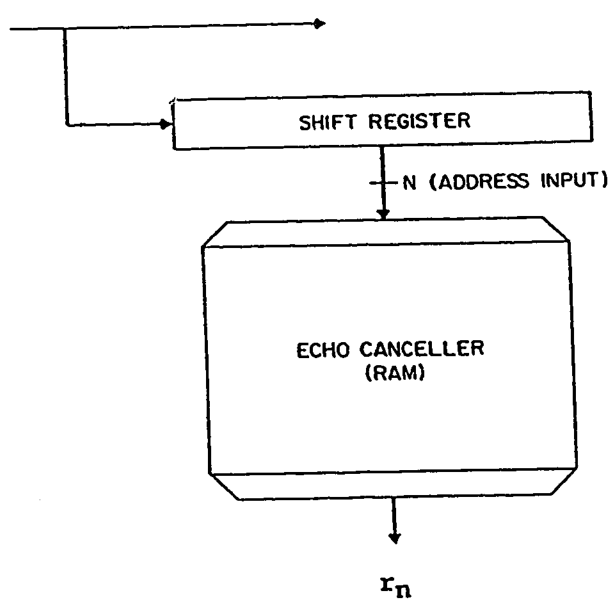


Fig. 2.5 Memory-based non-linear echo canceller

where μ is the adaptation step size, e_n is the residual signal which is composed of the cancellation error, the far-end signal and other uncorrelated noise, and $\vec{C}_n$ and $\vec{Y}_n$ are the tap coefficient vector and data vector at time n respectively. The LMS algorithm will be discussed in detail in a later section.

The cancellation can be done in analog domain by converting the echo canceller output to analog form, or in the digital domain by converting the hybrid output to digital form prior to cancellation. The problem of a linear echo canceller is that a 20 bit word length may be required to provide 55 dB cancellation due to non-linearity in the data-converter, transmitted pulse asymmetry, and timing jitter. A non-linear echo canceller may be the solution of the problem.

A non-linear echo canceller has to be used if the echo path has non-linearity that compromises the performance of a linear echo canceller. One implementation of a non-linear echo canceller based on the table-look up method is shown in Fig. 2.5. The echo replica of this echo canceller can be expressed as [10]:

$$r_n = f(y_n, y_{n-1}, \ldots, y_{n-N+1}) = f(\vec{Y}_n) \qquad (2.5)$$

where f(.) is a arbitrary nonlinear function and $\vec{Y}_n$ is the data vector. If the length of the echo canceller impulse response spans N baud intervals, and the symbols transmitted have M levels, a memory of size M^N words is needed to store the echo replica of all possible data patterns. The adaptation process of a non-linear echo canceller can be described by using the following equation:

$$f(\vec{x}, n+1) = f(\vec{x}, n) + \mu e_n \quad \text{for } \vec{x} = \vec{Y}_n \qquad (2.6)$$

where x is the address of the memory location, μ is the adaptation step size, e_n is the residual echo, and n and n+1 are time instants. The current near-end transmitted data vector $\vec{Y}_n$ is used as the address input to the memory. The echo replica for that particular transmitted data pattern in the memory location is updated, while the content at other memory locations will not change. When the transceiver at one end is being trained, the far-end signal received from the transmitter at the other end will slow down or diverge the adaptation process. To avoid this problem, during training of one transceiver, the transmitter at the other end of the line is turned off.

The advantages of using a non-linear echo canceller are:

1) It relaxes the echo path linearity requirement.
2) The echo replica is found by looking up in the memory instead of performing convolution.

The disadvantages of using a non-linear echo canceller are:

1) A non-linear echo canceller needs a much larger memory size. For M=2 and N=16, 2^{16} or 64k words are needed instead of 16 words in a linear echo canceller.
2) The rate of convergence slows down exponentially as N increases.

The best way is to implement an echo canceller which combines both types of cancellation such that the echo canceller will not get too complicated and the problem of non-linearity can be reduced. In [9], an echo canceller based on both types of cancellation methods is described. It achieves echo cancellation in excess of 70 dB.

2.2.3 Impulse noise

Impulse noise consists of infrequent high amplitude voltage bursts. In [11], the author suggested that impulse noise originates from the central office because field tests conducted by the New Brunswick Telephone Company (NBTel) shows that impulse noise on the subscriber side is smaller for longer loops. Impulse noise is usually measured by counting the number of events that are above certain threshold per unit time. In [12], impulse noise measurements performed in West Germany were reported. A considerable variation in the density of impulse bursts within 24 hours was found. The bursts occurred most frequently at the periods between 9 to 11 a.m. and 4 to 9 p.m.. Their amplitude probability distribution was approximated by

$$Q(u) = (u/u_0)^{-2} \; ; \qquad u_0 = 5 \text{ mV} \qquad (2.7)$$

where Q(u) was independent of the observation period and the bandwidth of the receiver. The pulse interval distribution could be expressed as the sum of exponential functions. In addition, spectral analysis of the impulse noise showed that the amplitude is noticeably greater for the spectral content below 40 kHz.

The sensitivity of a system to impulse noise depends on the line coding and amplitude of the transmitted symbols. It is desirable to have large amplitude for a given transmitted power, and a small bandwidth for a given bit rate. Simulation study was done in the same paper [12] to determine the transmission performance under impulse noise. It was found that the German (MMS43 code) and the US (2B1Q code) U-interface standards result in a similar performance despite the greater bandwidth requirement of the MMS43 code.

2.2.4 Intersymbol interference

The baseband pulse amplitude modulation (PAM) signal can be represented by the expression:

$$S(t) = \sum_{m=-\infty}^{\infty} A_m g(t-mT) \qquad (2.8)$$

where A_m is the transmitted symbol, and $g(t)$ is the transmitted pulse shape. If an ideal bandlimited channel has bandwidth $f_0/2$ and the transmitted pulse shape has the spectrum,

$$B(f) = \begin{cases} 1/f_0 \ ; & |f| \le f_0/2 \\ 0 \ ; & |f| > f_0/2 \end{cases} \qquad (2.9)$$

then the pulse shape is given by $g(t) = \sin(\pi f_0 t)/\pi f_0 t$. The pulse has zero crossings at all multiples of $1/f_0$ except at $t = 0$ where it is unity. If we set the symbol interval $T = 1/f_0$ and sample the signal at times $t = mT$ for integers m, then $S(mT)$ will equal to A_m. The neighbouring symbols will not interfere with one another at the sampling instant, and there will be no intersymbol interference (ISI). According to the first Nyquist criterion $f_0/2$ is the minimum bandwidth required for ISI free transmission given $T = 1/f_0$. This ideal bandlimited pulse which requires minimum bandwidth is physically unrealizable, but pulse shapes with excess bandwidth and exhibiting no ISI (Nyquist pulses) can be found. These pulse shapes must satisfy the first Nyquist criterion stated below.

$$\sum_{n=-\infty}^{\infty} P(f-nf_0) = 1 \tag{2.10}$$

One family of pulse shapes that satisfies the first Nyquist criterion is shown in Fig. 2.6.

$$H(f) = \begin{cases} 1 & |f| \le (1-\alpha)/2T \\ \left\{ 0.5 \left[1-\sin\left(\frac{\pi T}{\alpha} \left(|f| - \frac{1}{2T} \right) \right) \right] \right\} & (1-\alpha)/2T < |f| \le (1+\alpha)/2T \\ 0 & (1+\alpha)/2T < |f| \end{cases}$$

$$0 \le \alpha \le 1 \tag{2.11}$$

They are known as the raised-cosine pulses, because they can be represented as cosines raised by their peak amplitude. As α varies from 0 to 1, the bandwidth required changes from $f_0/2$ to f_0, and the pulse tail decays more rapidly as α is increased. The amount of excess bandwidth that should be introduced in a system is dependent on several factors. For a smaller excess bandwidth, the system will be more sensitive to the timing jitter and the timing phase. On the other hand, more noise will penetrate into the system with a larger excess bandwidth. For example, the NEXT noise power increases rapidly as the frequency increases. The optimal value of α will be different for different systems.

The digital subscriber loop will introduce frequency dependent attenuation and delay to the transmitted signal, so that the received signal will have ISI even if Nyquist pulses are sent. The ISI introduced

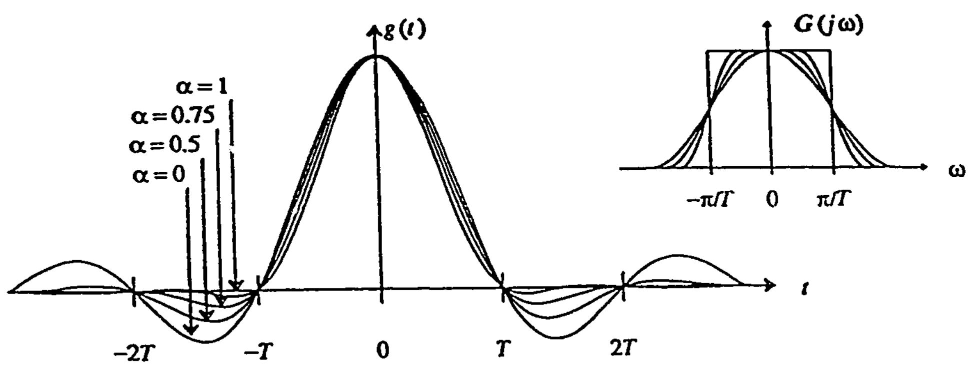


Fig. 2.6 The raised-cosine pulse with different values of roll-off factor

can be removed by a good equalization scheme. Some of the equalizer structures are discussed in the following chapter.

CHAPTER 3

METHCDS OF IMPROVING TRANSMISSION PERFORMANCE

The impairments that influence digital subscriber loop transmission were discussed in the previous chapter. In order to eliminate or reduce these impairments, there are several methods that were proved useful. One way is to use an adaptive equalizer to remove the intersymbol interference. In this chapter, different structures of adaptive equalizers with different degrees of complexity are discussed. Finally, the least mean-squared adaptation algorithm is introduced, which is commonly used to adapt the equalizer to the channel.

3.1 Fixed Equalizer

In the absence of bridged taps, the cable loss in decibels per unit length at high frequencies is roughly proportional to the square root of frequency, independent of the gauge or gauge mix. For instance, the cable frequency response can be approximated by the following expression [12]:

$$C(f) = C(0)e^{-bL\sqrt{f}} \qquad (3.1)$$

where L is the length of the cable, b is a positive constant that depends on the cable properties, and C(0) is the cable dc response. An equalizer which is designed to compensate for the length and $\sqrt{f}$ loss of the cable is known as the $\sqrt{f}$ equalizer. At frequencies below 10 kHz, the mismatch between the load and the characteristic impedance causes the attenuation to be asymptotic to its dc value. At frequencies above 200 kHz, the dielectric loss causes deviation from $\sqrt{f}$ characteristic.

Moreover, bridged taps introduce frequency-selective dips in the cable response, so the $\sqrt{f}$ equalization scheme is not useful for subscriber loops with widely varying transmission characteristics.

Although the $\sqrt{f}$ equalization is ineffective to be used on the digital subscriber loop, another type of fixed equalizer, known as the multiple response (MR) preequalizer, is found useful. The rejection of the dc component by the transformer at the line/network interface causes the echo pulse to have a long tail. This will increase the complexity of the echo canceller as more taps or more memory are required. The multiple response preequalizer with transfer function $P(z) = 1-z^{-1}$ can shorten the pulse tail through dc restoration. Fig. 3.1 shows the impulse response of a 5.28 km 26 AWG loop with and without the MR preequalizer. It is found that the length of the pulse tail is reduced substantially, but the equalized pulse response suffers a loss of signal amplitude which will degrade the noise immunity of the system. As a result, there is a tradeoff between the transceiver complexity and the system performance.

3.2 Linear discrete-time equalizers

3.2.1 T-spaced equalizer

One simple way to eliminate or reduce the ISI is to apply discrete-time linear equalization. A discrete-time linear receiver consists of a matched filter, baud-rate sampler and a linear equalizer, as shown in Fig. 3.2. The output of the matched filter is sampled at the baud rate and the samples are fed to the linear equalizer. [illegible] spaced sampling causes the channel frequency response to become [illegible], and the folded channel spectral response can be expressed as:

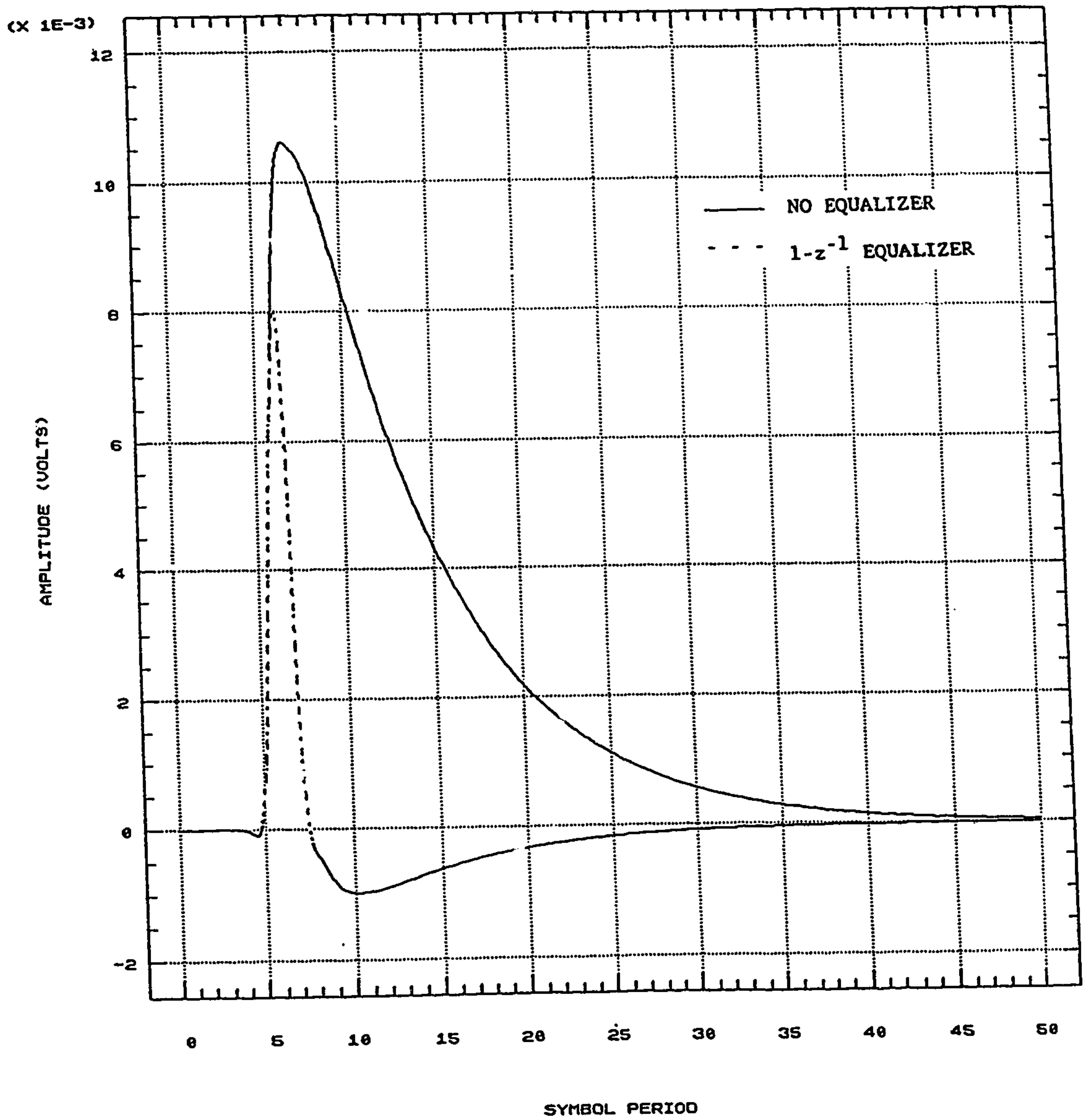


Fig. 3.1 Impulse response of a 5.28 km 26 AWG loop

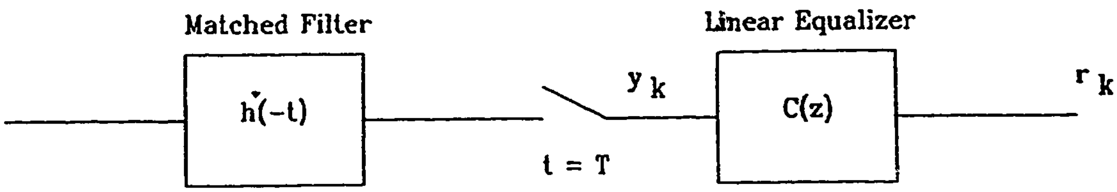


Fig. 3.2 T-spaced linear equalizer

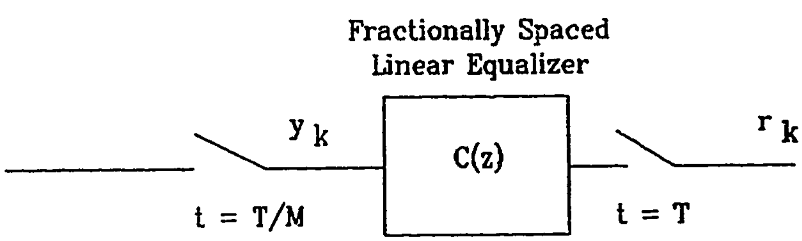


Fig. 3.3 Fractionally spaced linear equalizer

$$H'(f) = \sum_{n=-\infty}^{\infty} H(f-n/T) \qquad (3.2)$$

where H(f) is the channel frequency response, and H'(f) is the folded channel response. The principle of linear equalization is to provide gain at those frequencies where the folded channel has losses. Ideally, all the ISI can be removed if the linear equalizer transfer function has an inverse characteristics of the channel response. However, removing all ISI is not desirable because the noise power may be amplified excessively.

The linear equalizer can be expressed as a discrete time filter with the following transfer function:

$$C(z) = \sum_{n=0}^{N} c_n z^{-n} \qquad (3.3)$$

where N+1 is the number of equalizer taps, and the tap coefficients c_n are T-spaced samples of the equalizer impulse response. Therefore, this linear equalizer is sometimes called a T-spaced equalizer. The current and past values of the sampled received signal y(t-nT) are linearly weighted by the tap coefficients c_n, and summed to produce the equalizer output. The T-spaced equalizer output can be expressed as:

$$r_k = \sum_{n=0}^{N} c_n y(t_0+kT-nT) \qquad (3.4)$$

where t_0 is the timing phase. Because the signal spectrum is already aliased at the input to the T-spaced equalizer, the T-spaced equalizer is not able to compensate for phase and amplitude distortion simultaneously. The problem is solved by using another type of linear equalizer structure known as the fractionally spaced equalizer, which is discussed in the next section.

3.2.2 Fractionally spaced equalizer

The symbol-spaced discrete-time linear receiver discussed in the last section contains a matched filter at the front end. In practice, incomplete knowledge of the channel makes it impossible to realize the matched filter. For instance, the line configuration of the digital subscriber loop varies for each loop. A different equalizer structure called a fractionally spaced equalizer (FSE) is shown in Fig. 3.3. The FSE can be thought of as a T-spaced equalizer that operates at M times the baud rate (M is an integer), with the output sampled at the baud rate.

$$r_k = \sum_{n=0}^{N} c_n y(t_0+kT-nT/M) \qquad (3.5)$$

A properly designed FSE is equivalent to the matched filter plus the T-spaced equalizer, but the performance of the FSE is not affected by the sampling phase. The symbol-rate sampling at the input to a T-spaced equalizer causes aliasing. When the phases of the overlapping region are 180° apart, they add destructively and result in signal cancellation. The variable delay in the signal path causes variation in the sampling phase, which will strongly influence the effects of aliasing. The mean-squared error achieved by the T-spaced equalizer is a function of the sampling phase, and for signals with severe band-edge delay distortion, the performance is severely degraded. On the other hand, the higher sampling rate at the input of the FSE causes no aliasing, therefore; the FSE can compensate for the sampling phase as well as asymmetry in the channel amplitude or delay characteristics simultaneously. It has been shown that FSE with the same number of taps (half the time span) performs as well as the T-spaced equalizer, and noticeably better for channels with severe band-edge delay distortion [13].

3.3 Non-linear discrete-time equalizers

3.3.1 Decision-feedback equalizer

The linear equalizer discussed in the previous section is optimum in pure phase distortion; however, it does not perform satisfactorily in the presence of severe amplitude distortion. The performance can be improved by introducing nonlinearities in the equalizer. The decision-feedback equalizer (DFE) shown in Fig. 3.4 is one example of this approach. The DFE consists of a forward filter and a feedback filter. The forward filter can be a T-spaced or a fractionally spaced equalizer, and the feedback filter is a T-spaced equalizer. Samples of the received signal will feed through the forward filter, and the output of the feedback filter is subtracted from it. Decisions are made by the slicer on the equalized signal, and the detected symbols will feed through the feedback filter. The idea is that if the values of the past symbols are known correctly, then the ISI contributed by these symbols can be cancelled exactly by subtracting past symbol values with appropriate weighting from the equalizer output. Only the past symbols are known, and therefore the DFE can eliminate only the postcursor ISI.

The advantage of a DFE is that it can compensate for amplitude

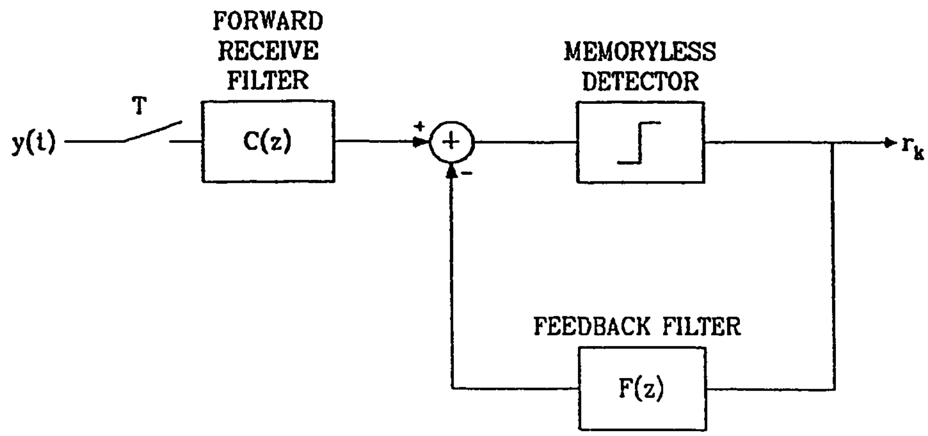


Fig. 3.4 Decision-feedback equalizer

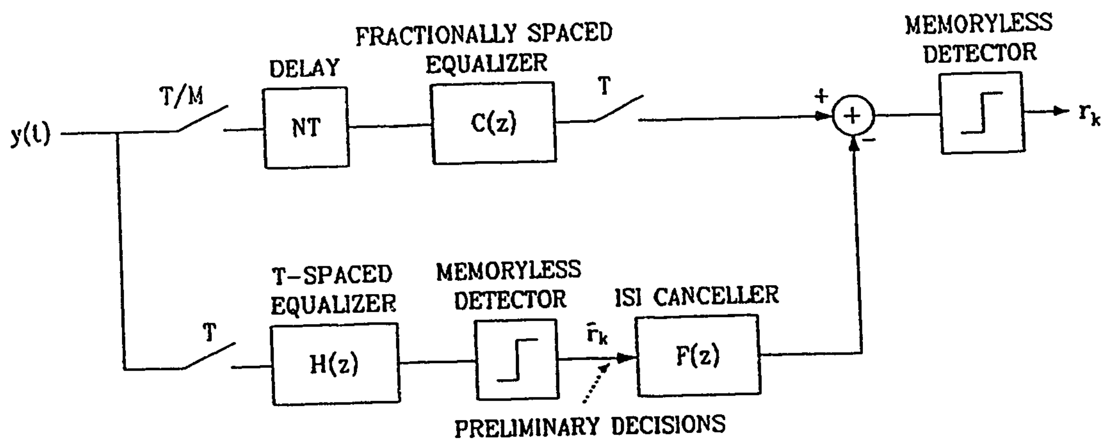


Fig. 3.5 Decision-aided ISI canceller

distortion without as much noise enhancement as a linear equalizer. It is because the cancellation of the postcursor ISI using the feedback filter does not enhance noise, since any noise present is eliminated by the slicer, and there is no noise at the output of the feedback filter. The forward linear equalizer has to remove only the precursor ISI, and therefore the noise enhancement will not be as much as when all the ISI is removed. This explanation assumes that all past symbols are detected correctly. When an incorrect decision is made, the DFE output will reflect this error during the next few symbols as the incorrect decision traverses the feedback filter. Thus it is more probable that more incorrect decisions will be made following the first one, which is error propagation. The problem of error propagation in DFE is examined in many papers [14], and it is shown that the benefit of reduced noise enhancement usually far outweighs the effect of error propagation.

3.3.2 Decision-aided ISI canceller

The concept of decision feedback of past data symbols to cancel ISI can be extended to include future data symbols. If all past and future symbols are assumed to be known at the receiver, all ISI can be cancelled at the receiver without any noise enhancement. A decision-aided ISI canceller in the receiver is proposed to reduce intersymbol interference without much noise penalty [15,16]. The structure is shown in Fig. 3.5, and it consists of a T-spaced linear equalizer, an ISI canceller and a fractionally spaced equalizer. The T-spaced equalizer will provide the preliminary decisions to the ISI canceller to produce an estimate of the actual interference imposed on the symbols. This estimate of the ISI is then subtracted from the output of the fractionally spaced equalizer. Theoretically, an infinite length ISI canceller with all the preliminary decisions being correct, can remove all the intersymbol interference. The ISI canceller approaches the zero-ISI matched filter performance, and the mean-squared error will reach the minimum value. The performance of an ISI canceller with a 62 T/2 taps matching filter and a 31 taps T-spaced transversal filter can be 3 dB better than a 64 T/2 taps linear equalizer on some selected channels [15].

3.4 Adaptive equalization

The channel characteristics for each loop are different due to different line configurations. It would be tedious or impossible if an equalizer had to be designed for each loop. An adaptive equalizer which is able to change its tap coefficients to adapt to the channel will be useful since the same equalizer can be used on all loops. The adaptation algorithm can be optimized using different criteria, and three design criteria, which are known as the zero-forcing (ZF), the mean-squared error (MSE) and the minimum probability of error (Min P_e) are used in the design of equalizers. Although the minimum probability of error is the most desired criterion, the Min P_e equalizer is very complicated and seldom used in practice. The formulation of the Min P_e equalizer can be found in [17]. The ZF equalizer and the MSE equalizer are favoured for their simplicity, and they will be discussed in the following sections.

3.4.1 Zero-forcing equalizer

Suppose the discrete time baseband channel can be described by its transfer function H(z). Ideally, the ISI introduced by the channel can be completely removed if a equalizer with the transfer function $C(z) = H^{-1}(z)$ is implemented (an inverse filter), such that the composite system response is given by $H(z)H^{-1}(z) = 1$. This ZF equalizer results in an equalized channel which satisfies the Nyquist criterion for no ISI. The problem is that the noise power may increase tremendously (noise enhancement) if large gain is required to compensate for the channel losses. It can be shown that with additive Gaussian noise of power spectral density (PSD) N_0, the total noise power at the equalizer output is given by [18]:

$$\sigma^2(\text{LE-ZF}) = \frac{N_0 T}{2\pi} \int_{-\pi/T}^{\pi/T} H^{-1}(e^{j\omega T})\,d\omega \tag{3.6}$$

At the extreme, σ^2(LE-ZF) may be infinitely large, if the folded spectrum of the channel has zeroes over a range of frequencies.

3.4.2 Mean-squares error equalizer

The ZF linear equalizer completely removed the ISI. From the point of view of minimizing the probability of error, it may be advantageous to allow some residual ISI in order to reduce the amount of noise enhancement. The MSE equalizer minimizes the mean-squares error between the equalizer output and the transmitted data symbols, and the noise enhancement is always smaller for a MSE equalizer than for a ZF equalizer. If the successive data symbols are uncorrelated with variance $\sigma_A{}^2$, and the PSD of the additive Gaussian noise is N_0, it can be shown that the MSE equalizer transfer function is given by [18]:

$$C(z) = \frac{1}{H(z) + N_0/\sigma_A^2} \tag{3.7}$$

and the total noise power at the equalizer output is given by:

$$\sigma^2(\text{LE-MSE}) = \frac{N_0 T}{2\pi} \int_{-\pi/T}^{\pi/T} C(e^{j\omega T}) d\omega \tag{3.8}$$

Comparing equation (3.6) and (3.8), we found that $\sigma^2(\text{LE-MSE}) \leq \sigma^2(\text{LE-ZF})$ because of an extra positive term in the denominator that makes the integrand smaller. Because of the greater noise enhancement and possible instability of the ZF equalizer, MSE equalizer is preferred in most applications. The following section will introduce the adaptation algorithm which is used to implement an MSE equalizer.

3.5 Adaptation algorithm

3.5.1 The least mean-squares (LMS) adaptation algorithm

A linear transversal filter with N taps is shown in Fig. 3.6. Throughout this section, capital letters are used to represent a matrix or a vector, lower case for scalars, and superscript T and -1 to represent transposition and inversion respectively. The output signal r(n) at time n can be expressed as the weighted sum of the input signal y(n):

$$r(n) = \sum_{k=1}^{N} c_k(n) y(n-k+1) = C^T(n) Y(n) \tag{3.9}$$

where $Y(n) = [y(n), y(n-1), \ldots, y(n-N+1)]^T$ is the input signal vector and $C^T(n) = [c_1(n), c_2(n), \ldots, c_N(n)]$ is the tap weight vector.

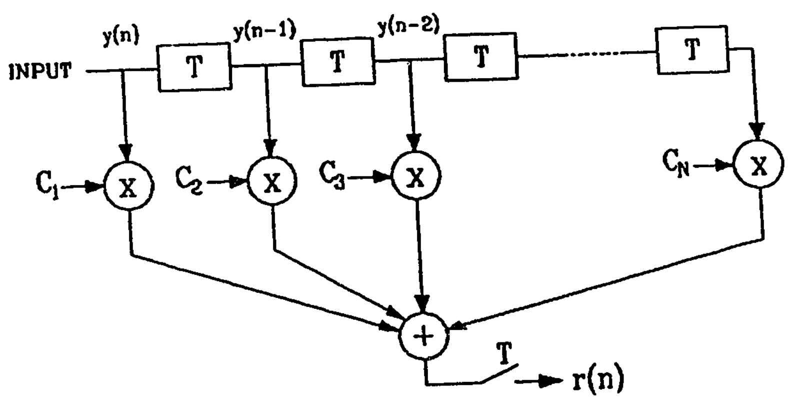


Fig. 3.6 Linear transversal filter

The error signal e(n) is obtained by subtracting the output signal r(n) from the desired signal d(n):

$$e(n) = d(n) - r(n) = d(n) - C^T(n)Y(n) \qquad (3.10)$$

and the mean-squared error J(n) at time n is given by:

$$\begin{aligned} J(n) &= E\{e^2(n)\} \\ &= E\{d^2(n)\} - 2E\{d(n)Y^T(n)\}C(n) + C^T(n)E\{Y(n)Y^T(n)\}C(n) \\ &= \sigma_d^2 - 2P^T(n)C(n) + C^T(n)R(n)C(n) \qquad (3.11) \end{aligned}$$

where $\sigma_d^2 = E\{d^2(n)\}$ is the variance of the desired response

$P(n) = E\{Y(n)d(n)\}$ is the cross-correlation matrix between the input vector and the desired response

$R(n) = E\{Y(n)Y^T(n)\}$ is the auto-correlation matrix of the input vector

In order to minimize the MSE, we differentiate J(n) with respect to the weighting vector C(n), and set the gradient to zero. The tap weight vector C^* thus obtained is optimum in the sense of mean-squared error. After some manipulations, the results are given by:

$$\begin{aligned} \nabla(n) &= \frac{\partial J(n)}{\partial C(n)} \\ &= -2P + 2RC = 0 \\ \therefore \qquad C^* &= R^{-1}P \qquad (3.12) \end{aligned}$$

The optimum tap weight vector C^* can therefore be calculated by solving N simultaneous equations. This brute force approach is not practical because the auto-correlation matrix R is not necessarily invertible; even if R is invertible, the computation is too intensive if

N is large.

The mean-squared error $J(n)$, which is known as the error performance function, is a quadratic function of the tap weight vector. The global minimum of the error performance function J_{min} can be obtained using the steepest gradient descent method. This method makes use of the fact that if the tap weight vector is updated successively in the direction of the negative of the gradient vector, it will eventually lead to the optimum tap weight vector C^* with minimum mean-squared error J_{min}. In other words, the tap weight vector is updated recursively as follows:

$$C(n+1) = C(n) + \mu[\ -\nabla(n)] \tag{3.13}$$

where μ is a small positive constant known as the step size parameter. If we define $\delta(n) = C(n) - C^*$ as the tap weight error vector, then the mean-squared error after n iterations is given by:

$$J(n) = J_{min} + \delta(n)R(n)\delta(n)^T \tag{3.14}$$

It can be shown that as $n \to \infty$, then $\delta(n) \to 0$ and $J(n) \to J_{min}$ if the step size parameter satisfies the following relationship.

$$0 < \mu < 1/\lambda_k\ , \qquad k = 1,\ldots,N \tag{3.15}$$

where λ_k are the eigenvalues of the auto-correlation matrix R.

If the gradient vector $\nabla(n)$ is known exactly, and μ is suitably chosen, the optimum tap weight vector will eventually be obtained. In reality, the gradient vector $\nabla(n)$ is not known and the alternative is to estimate the true gradient from the gradient of the instantaneous squared error. The gradient of the instantaneous squared error is given by:

$$\begin{aligned}\nabla[e^2(n)] &= 2e(n)\nabla[e(n)] \\ &= 2e(n)\nabla[d(n) - C^T Y(n)] \\ &= -2e(n)Y(n) \qquad (3.16)\end{aligned}$$

If we replace the true gradient by this estimated gradient, the new gradient search algorithm is given by:

$$C(n+1) = C(n) + 2\mu e(n)Y(n) \qquad (3.17)$$

The result is the least mean-squares (LMS) or the stochastic gradient (SG) algorithm. The behaviour of the LMS algorithm is dependent on three factors: the step size parameter μ, the number of taps N, and the eigenvalues of the auto-correlation matrix R. The mathematical details are derived in most adaptive filter theory texts [19,20], so the following properties of the LMS algorithm will be listed without proof.

(1) The use of a noisy gradient to iterate the optimum tap weight vector results in a MSE that is greater than J_{min}. The extra noise term is called the excess mean-squared error J_{ex}. The steady state MSE is given by the following expression.

$$J(\infty) \simeq J_{min} + \mu J_{min} \sum_{k=1}^{N} \lambda_k \qquad (3.18)$$

(2) For a small value of μ, the adaptation is slow, but the residual MSE after adaptation is smaller because of the large amount of data used by the algorithm to estimate the gradient

vector. On the other hand, when μ is large, the adaptation is relatively fast at the expense of a larger residual MSE.

(3) The convergence properties of the average MSE E[J(n)] depend on the number of taps N. The necessary and sufficient condition for E[J(n)] to be convergent is as follows:

$$0 < \mu < \frac{2}{\sum_{k=1}^{N} \lambda_k} \qquad (3.19)$$

The LMS algorithm is convergent in the MSE when this condition is satisfied. On the other hand, the necessary and sufficient condition for the tap weight vector E[C(n)] to be convergent is given by the expression:

$$0 < \mu < \frac{2}{\lambda_{max}} \qquad (3.20)$$

where λ_{max} is the largest eigenvalue of R. The LMS algorithm is convergent when this condition is satisfied. Since λ_{max} must be smaller than the sum of all eigenvalues, the convergence condition for E[C(n)] is automatically met if the step size parameter μ is chosen to meet the convergence condition for E[J(n)].

(4) When the eigenvalues of the autocorrelation matrix R are widely spread, the residual MSE is primarily determined by the largest eigenvalue. The convergence time for E[C(n)] is determined by the smallest eigenvalue while the convergence time for E[J(n)] is less dependent on the spread of

eigenvalues of R. In general, the LMS algorithm will take longer to converge if the auto-correlation matrix R has widely spread eigenvalues (ill conditioned).

This will conclude the discussion on the LMS algorithm. Although there are other more sophisticated and faster convergence adaptation algorithms such as the recursive least squares (RLS) and the fast Kalman algorithm, they are rarely used because of their computational complexity and possible instability. In a stationary environment such as the digital subscriber loop, the echo canceller and the DFE have to be trained only once since the channel characteristics will not change or only vary slowly due to temperature changes. So we can afford the price of adapting the echo canceller and DFE slowly to the channel in exchange for a simpler transceiver design. The LMS algorithm is also favoured in a non-stationary environment such as the radio channel where the channel characteristics are changing rapidly. The reason is that with the system operating at a high symbol rate, the hardware can not handle the computational requirements of other adaptation algorithms. An alternative is to choose a bigger step size μ so that changes in the channel characteristics can still be tracked, but the price to pay is a bigger MSE.

3.5.2 Updating DFE taps using LMS algorithm

In the previous section, the LMS algorithm was introduced, and in this section, a DFE is adapted using this algorithm. It is assumed that the forward filter has N1 non-causal taps and N2 causal taps, and the feedback filter has N3 causal taps. The structure of this DFE is shown

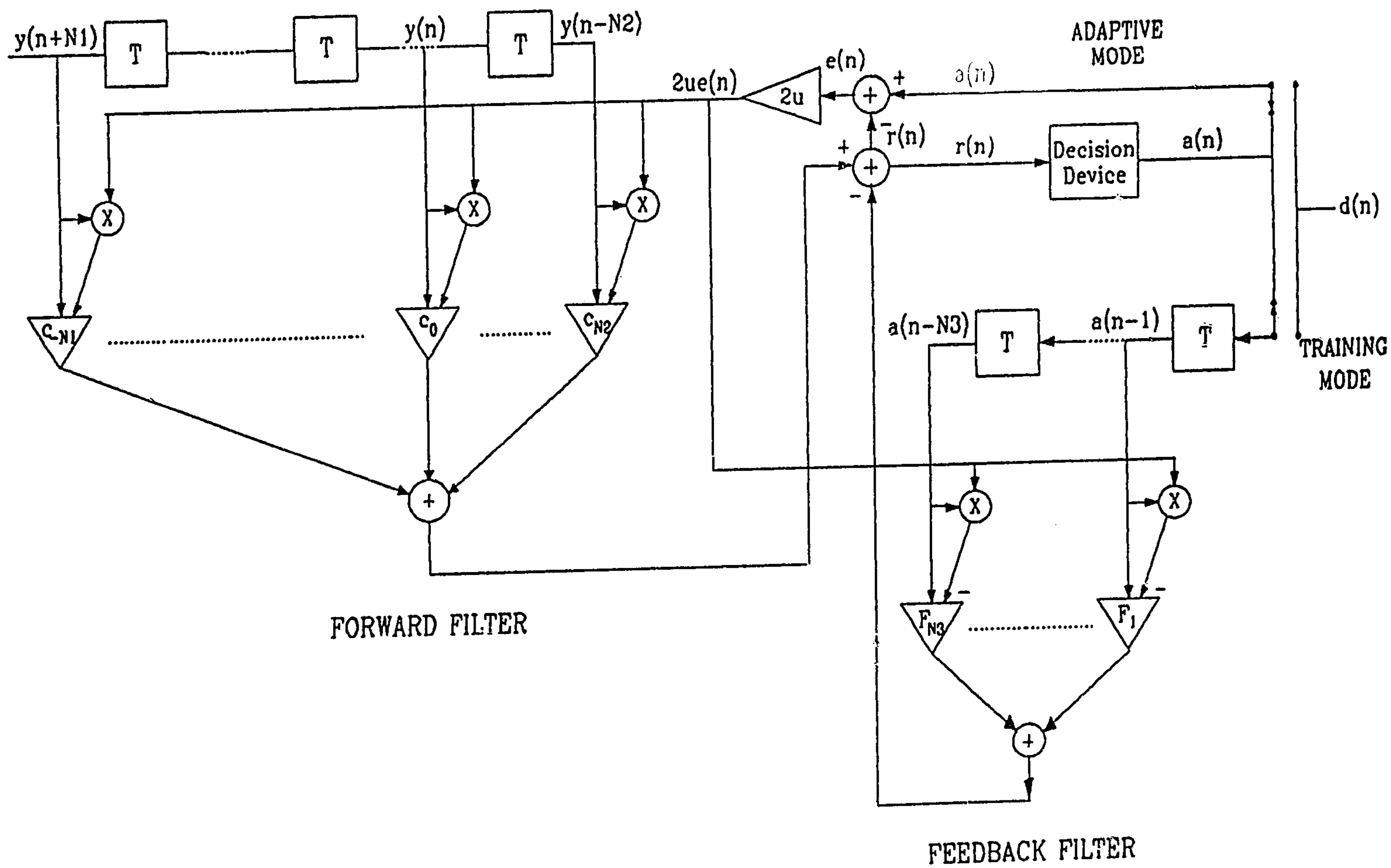


Fig. 3.7 Adaptive decision-feedback equalizer

in Fig. 3.7. The output $r(n)$ that feeds into the decision device can be expressed as:

$$r(n) = \sum_{k=-N1}^{N2} c_k(n)y(n-k) - \sum_{k=1}^{N3} f_k(n)a(n-k) \qquad (3.21)$$

where $c_k(n)$ and $y(n-k)$ are the tap coefficients and the input to the forward filter, $f_k(n)$ and $a(n-k)$ are the tap coefficients and the input to the feedback filter. The above equation can be expressed in vector form using the following substitution:

$$V_C(n) = [c_{-N1}(n),\ldots,c_0(n),\ldots,c_{N2}(n)] \qquad (3.22)$$

$$V_F(n) = [f_1(n),\ldots,f_{N3}(n)] \qquad (3.23)$$

$$V_Y(n) = [y(n+N1),\ldots,y(n),\ldots,y(n-N2)] \qquad (3.24)$$

$$V_A(n) = [a(n-1),\ldots,a(n-N3)] \qquad (3.25)$$

then $r(n)$ can be expressed as

$$r(n) = V_C(n)V_Y{}^T(n) - V_F(n)V_A{}^T(n) \qquad (3.26)$$

Equation (3.21) can be further simplified if the two tap weight vectors are combined into one single tap weight vector, and the two input vectors are combined into a single input vector. For instance, let

$$W(n) = (V_C(n),-V_F(n)) \text{ and } X(n) = (V_Y(n),V_A(n)) \qquad (3.27)$$

then $r(n)$ can now be simplified to

$$r(n) = W(n)X^T(n) \qquad (3.28)$$

The error signal e(n) is obtained by subtracting r(n) from the desired response d(n).

$$e(n) = d(n) - r(n) = d(n) - W(n)X^T(n) \qquad (3.29)$$

This equation is similar to equation (3.10), and we can use the previously derived results substituting this equation into the LMS algorithm.

$$W(n+1) = W(n) + 2\mu e(n)X(n) \qquad (3.30)$$

The forward equalizer and the feedback equalizer are adapted respectively as shown:

$$c_k(n+1) = c_k(n) + 2\mu e(n)y(n-k) \qquad k = -N1,\ldots,0,\ldots,N2 \qquad (3.31)$$

$$f_k(n+1) = f_k(n) - 2\mu e(n)a(n-k) \qquad k = 1,\ldots,N3 \qquad (3.32)$$

where k denotes the position of the taps, n and n+1 are the time instants. Equations (3.31) and (3.32) can be used together or alone depending on the equalizer structure. For a T-spaced or a fractionally spaced linear equalizer, equation (3.31) is used. For a DFE with no forward filter, N1 and N2 are set to zero, and the forward filter now acts as an automatic gain control device. In the most general case, N1, N2 and N3 are arbitrary integers, and both forward and feedback equalizers are adapted jointly using (3.31) and (3.32).

CHAPTER 4

SIMULATION OF THE TRANSMISSION SYSTEM

In this chapter, the simulation of the transmission system will be discussed. In order to simulate the rate-adaptive transmission system, we have to model a multi-rate transceiver with the control protocol working under a noisy loop environment. As explained in chapter 1, the control protocol for bit-rate maximization will not be implemented, and the maximization process is done manually. The simulation model can now be divided into two parts: the subscriber loop model and the transceiver model. In order to study the RA-DSL transmission method in the subscriber loop, we must be able to simulate the electrical characteristics of loops of any configurations. The solution is to use transmission (ABCD) matrices to represent the loops, the details of which are discussed in section 4.1. The transceiver structure and the methods of simulating different types of disturbances are discussed in section 4.2-4.3. The training of the adaptive equalizer will be discussed in section 4.4, and finally methods of calculating the eye diagram, and quantiser signal to noise ratio are given in section 4.5.

4.1 Subscriber loop model

4.1.1 Calculation of Z_0 and γ

The loop, which is made up by the 19, 22, 24, and 26 AWG polyethelene insulated cable (PIC), constitutes the most variable component in the system. The cable as a transmission line can be modelled using the lumped-parameter circuit approach. Fig. 4.1 shows the equivalent circuit of a transmission line segment of infinitesimal

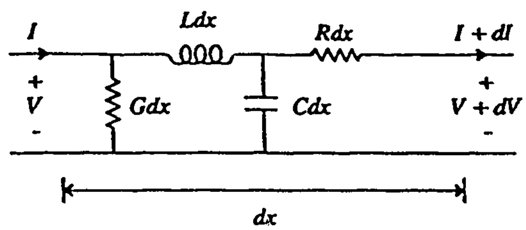


Fig. 4.1 Lumped-parameter model for a short section of transmission line

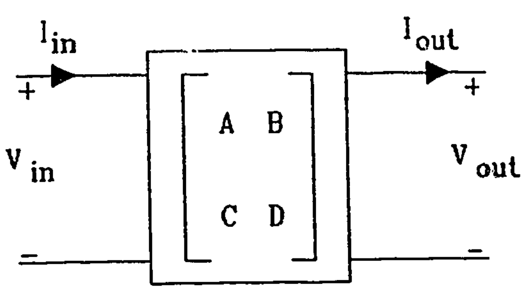


Fig. 4.2 The ABCD matrix

length dx. The model is represented by four parameters, namely the series resistance R, series inductance L, shunt conductance G and shunt capacitance C per unit length, which are called the primary constants of the transmission line. The primary constants of actual cables depend on factors such as the geometry and material used in the insulation. The measured values of the primary constants for the 19, 22, 24, and 26 AWG polyethelene insulated cable are available from 1 Hz to 5 MHz [1]. These data are given in imperial units, and conversion to metric units is done when necessary. It was found that the resistance is proportional to the square root of frequencies at high frequencies due to skin effect, the inductance is a slowly decreasing function of frequency ranging from about 0.62 mH/km at low frequencies to about 70% of that value at high frequencies, the conductance is negligibly small at low frequencies, and the capacitance is independent of frequency with a value of 0.0515 μF/km. From the primary constants, the secondary parameters such as the characteristic impedance (Z_0) and the propagation constant (γ) can be calculated as follows:

$$Z_0(\omega) = \left[\frac{R(\omega)+j\omega L(\omega)}{G(\omega)+j\omega C(\omega)} \right]^{0.5} \tag{4.1}$$

$$\gamma(\omega) = [(R(\omega)+j\omega L(\omega))(G(\omega)+j\omega C(\omega)]^{0.5} \tag{4.2}$$

The FORTRAN program interpol.for in Appendix D was developed to calculate the Z_0 and γ for the 19, 22, 24, and 26 AWG. The RLGC data for the frequency range of 1 Hz to 5 MHz were available at 0, 70 and 120°F (-17.8, 21.1 and 48.9°C respectively). The cubic spline algorithm, which uses a cubic polynomial as the interpolating function, was used to interpolate any intermediate RLG values that are not available in the data. Throughout this section, the first and second derivatives of some general function y (RLG parameters) are represented by y' and y'' respectively, and N is the number of input values of y. The second derivatives are defined by a set of N-2 linear equations in the N unknowns y_i'', $i=1...,N$. A unique solution is obtained by specifying the second derivatives at the boundaries, y_1'' and y_N''. Alternatively, y_1'' and y_N'' can be calculated from specified values of the first derivatives y_1' and y_N'. The values of y_1'' and y_N'' are crucial when values are extrapolated from the data, but their effects are negligible for interpolation. In the calculation, they are simply set to zero (natural cubic spline) since the maximum frequency of interest is 768 kHz (the required bandwidth for the transmission of 24 B channels using 2B1Q code). After y_i'' are calculated, y at any desired frequencies can be found by using the formulae shown below.

$$y(f) = Ay_i + By_{i+1} + Cy_i'' + Dy_{i+1}'' \tag{4.3}$$

where:

$$A = \frac{f_{i+1} - f}{f_{i+1} - f_i} \qquad B = 1 - A$$

$$C = \frac{1}{6}(A^3 - A)(f_{i+1} - f_i)^2 \qquad D = \frac{1}{6}(B^3 - B)(f_{i+1} - f_i)^2 \tag{4.4}$$

The values of y and y'' at f_i and f_{i+1}, which straddle the frequency of interest f, are used to calculate the value y(f). After the RLGC parameters are interpolated at all desired frequencies, Z_0 and γ are then calculated using equations (4.1) and (4.2).

4.1.2 Calculation of ABCD matrices

The transmission (ABCD) matrix relates the input voltage and current with the output voltage and current, and the graphical representation is shown in Fig. 4.2.

$$\begin{bmatrix} V_{in} \\ I_{in} \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} V_{out} \\ I_{out} \end{bmatrix} \tag{4.5}$$

It can be shown that the elements ABCD for a homogeneous line are given by:

$$A = \mathrm{Cosh}(\gamma L) \qquad B = Z_0 \mathrm{Sinh}(\gamma L)$$
$$C = \frac{\mathrm{Sinh}(\gamma L)}{Z_0} \qquad D = \mathrm{Cosh}(\gamma L) \tag{4.6}$$

where L is the length of the cable segment, sinh and cosh are hyperbolic sine and cosine respectively. Z_0 and γ are calculated from the RLGG parameters, and the details have been discussed in the previous section. Every subscriber loop is made up by sections of homogenous line, and therefore any configuration of loops can be modelled in the frequency domain by ABCD matrices. Loops with and without bridged taps will be considered separately.

(a) Mixed gauges without bridged tap

Fig. 4.3 shows a subscriber loop with mixed gauges and no bridged tap. The ABCD matrices of each homogeneous cable segment at all necessary frequencies are first calculated, and then multiplied together

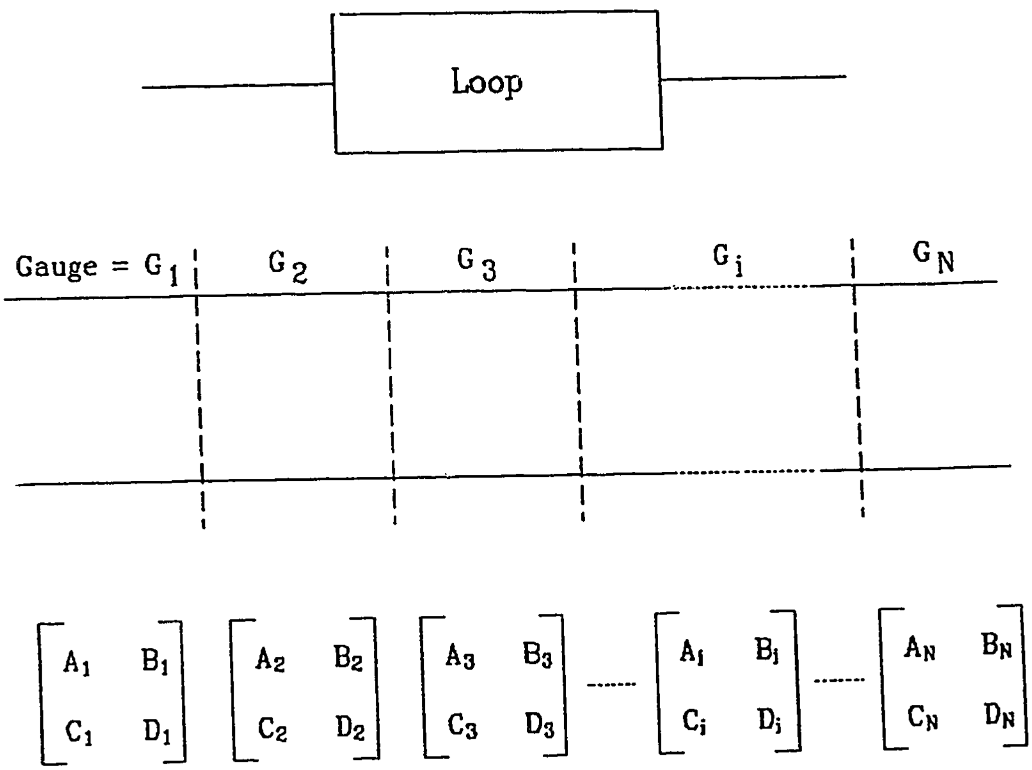


Fig. 4.3 Calculation of ABCD matrices for loop without bridged tap

to calculate the composite ABCD matrices of the loop. The formulation can be expressed as shown [21]:

$$\begin{bmatrix} A_{composite} & B_{composite} \\ C_{composite} & D_{composite} \end{bmatrix} = \prod_{i=1}^{N} \begin{bmatrix} A_i & B_i \\ C_i & D_i \end{bmatrix} \qquad (4.7)$$

(b) Mixed gauges with bridged taps

Fig. 4.4 shows a subscriber loop with mixed gauges and bridged taps. Bridged tap is seen as a stub from the main line and therefore the trick is to transform the bridged tap into a series element. The ABCD matrix for a bridged tap is derived in Appendix A while the result is:

$$\begin{bmatrix} A_{bridge} & B_{bridge} \\ C_{bridge} & D_{bridge} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ C/A & 1 \end{bmatrix} \qquad (4.8)$$

The parameters A and C are given by $Cosh(\gamma L)$ and $Sinh(\gamma L)/Z_0$ respectively if the bridged tap is a homogenous cable, and they are replaced by $A_{composite}$ and $C_{composite}$ if the bridged tap has mixed gauges. After the bridged tap is transformed into the main line as a series element, the calculation is the same as for a line without bridged tap.

The FORTRAN program dtxline.for in Appendix D was written to calculate the ABCD matrices of the subscriber loop including bridged tap on a bridged tap. The configuration of the line is expressed as a binary tree. The bridged taps represent the leaves, and the main line is the

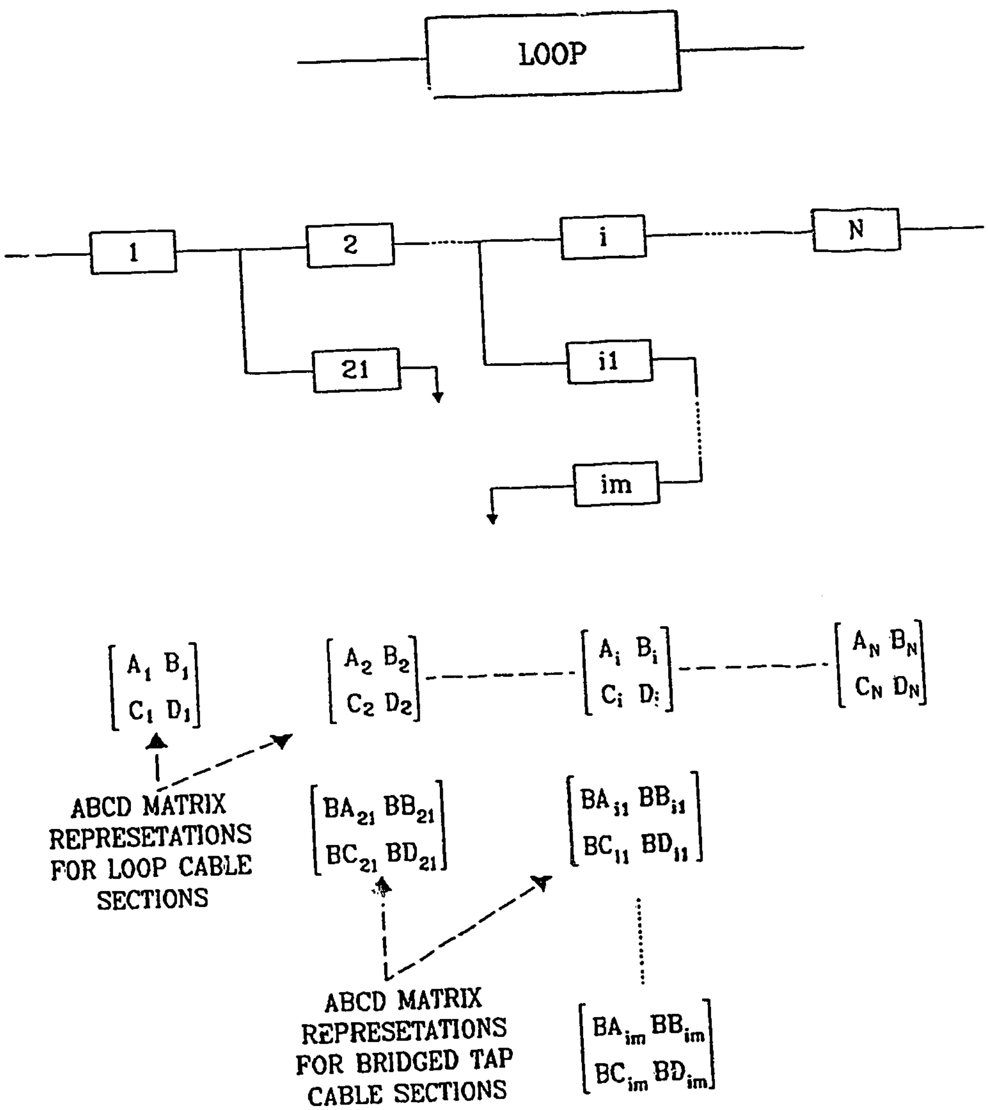


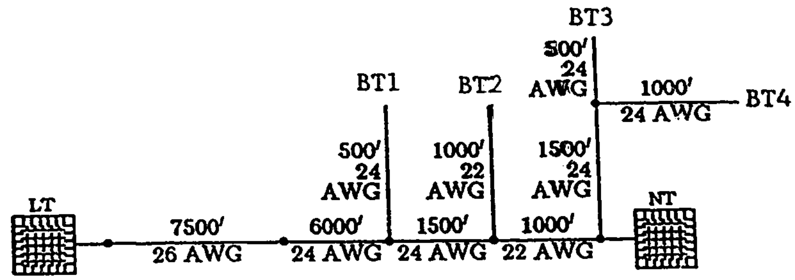
Fig. 4.4 Calculation of ABCD matrices for loop with bridged taps

parent. Bridged tap becomes the parent, if another bridged tap branches from it. The configuration of ANSI ISDN standard loop #3 [1] and its input file for the program are shown in Fig. 4.5. There are 4 columns in the input file: the first column is the control word, the second column is the cable gauge, the third column is the temperature of the cable, and the fourth column is the length of the cable. The control word can be 0, 1, or 2 depending on the loop configuration. Three different situations are explained below:

(1) On the main line, control word '1' represents a section of homogeneous cable, and the gauge, temperature and length of the cable will be typed in the remaining three columns. Control word '2' represents a bridged tap branched out from the main line and it causes the control transfer to the bridged tap. Control word '0' represents end of the main line, and the program will be terminated.

(2) On a bridged tap, control word '1' represents a section of homogeneous cable, and the gauge, temperature and length of the cable will be typed in the remaining three columns. Control word '2' represents a leaf bridged tap branched out from the parent bridged tap and the control will be transferred to the leaf bridged tap. Control word '0' represents the end of the bridged tap, and the control will be transferred to main line.

Loop #3:



1	26	70	7500		
1	24	70	6000		
2	0	0	0		
1	24	70	500	BT1	
0	0	0	0		
1	24	70	1500		
2	0	0	0		
1	22	70	1000	BT2	
0	0	0	0		
1	22	70	1000		
2	0	0	0		
1	24	70	1500		
2	0	0	0		
1	24	70	1000	BT4	BT3
0	0	0	0		
1	24	70	500		
0	0	0	0		
0	0	0	0		

Fig. 4.5 Line configuration and input file of ANSI standard loop #3

(3) On a leaf bridged tap, control word '1' represents a section of homogeneous cable, and the gauge, temperature and length of the cable will be typed in the remaining three columns. Control word '0' represents end of the leaf bridged tap, and the control will be transferred back to the parent bridged tap.

The program can only handle cases up to the case with bridged tap on a bridged tap unless more levels of subroutines are used. On the other hand, other high level languages such as 'C' can ideally handle an infinite number of levels of bridged tap using recursion. This will not be a problem because subscriber loops rarely have more than 2 levels of bridged taps (bridged tap on a bridged tap), and the built-in complex number handling capability is an asset in FORTRAN. The program will first read in the line configuration, then the ABCD matrices of each homogeneous cable section are calculated and multiplied together to obtain the composite ABCD matrices. Whenever bridged tap is encountered, the ABCD matrices of the bridged tap are calculated and multiplied by the ABCD matrices of the main line. This process goes on until the line is terminated. The flow chart of the program is shown in Fig. 4.6 to illustrate the mechanism. The ABCD matrices for the other direction of transmission can be obtained by interchanging the $A_{composite}$ and $D_{composite}$ terms.

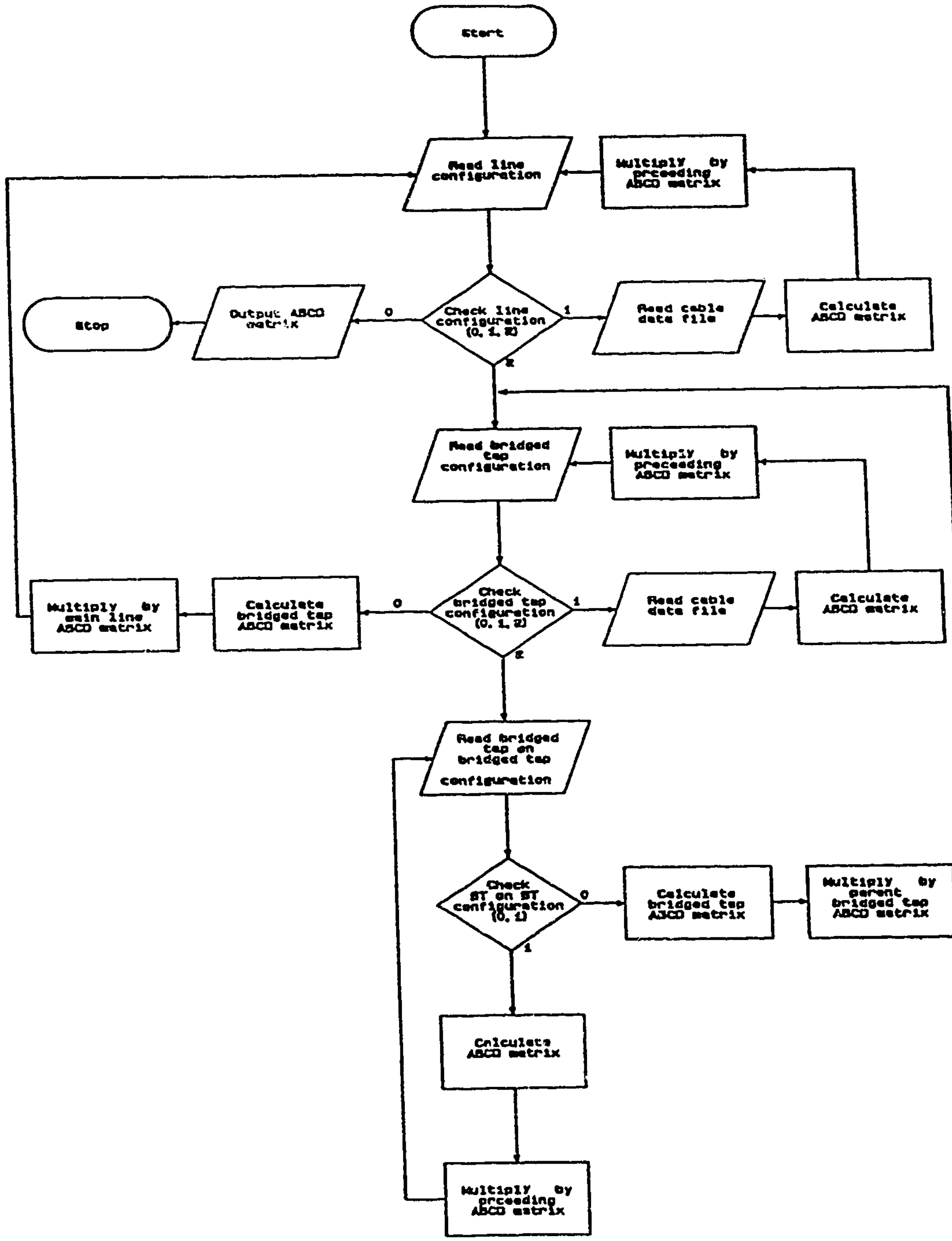


Fig. 4.6 Flow chart for the calculation of ABCD matrices

4.2 Transceiver structure

A conventional structure of a duplex subscriber loop transceiver, such as used in ISDN basic access terminals utilizing the echo cancellation method of achieving duplex communication, is assumed in the simulation study. A simplified block diagram of this transceiver is shown in Fig. 4.7. The encoder maps the incoming binary data into the transmitted symbols. The choice of line code will affect the power spectral density of the transmitted signal, and eventually the amount of crosstalk induced and required equalization. The 2B1Q (2 binary, 1 quaternary) code, which is a 4 level pulse amplitude modulation (PAM) code without redundancy, has been accepted by the ANSI as the standard line code for the basic access interface [1]. The mapping of the two binary bits into the quaternary symbols is specified as shown.

First Bit	Second Bit	Quaternary Symbol
1	0	+3
1	1	+1
0	1	-1
0	0	-3

Gray coding is used in the mapping such that the two neighbouring symbols have only one bit difference. When a symbol error occurs, it will only cause one bit error in most circumstances. Therefore the bit error probability is only half as large as the symbol error probability assuming distance 1 symbol errors are the case, as is reasonable in practice.

The line driver will amplify the input signal to a proper level so that the transmitted power at the line or network interface will have an average value of 13.5 dBm required by the ISDN standard. The square-root

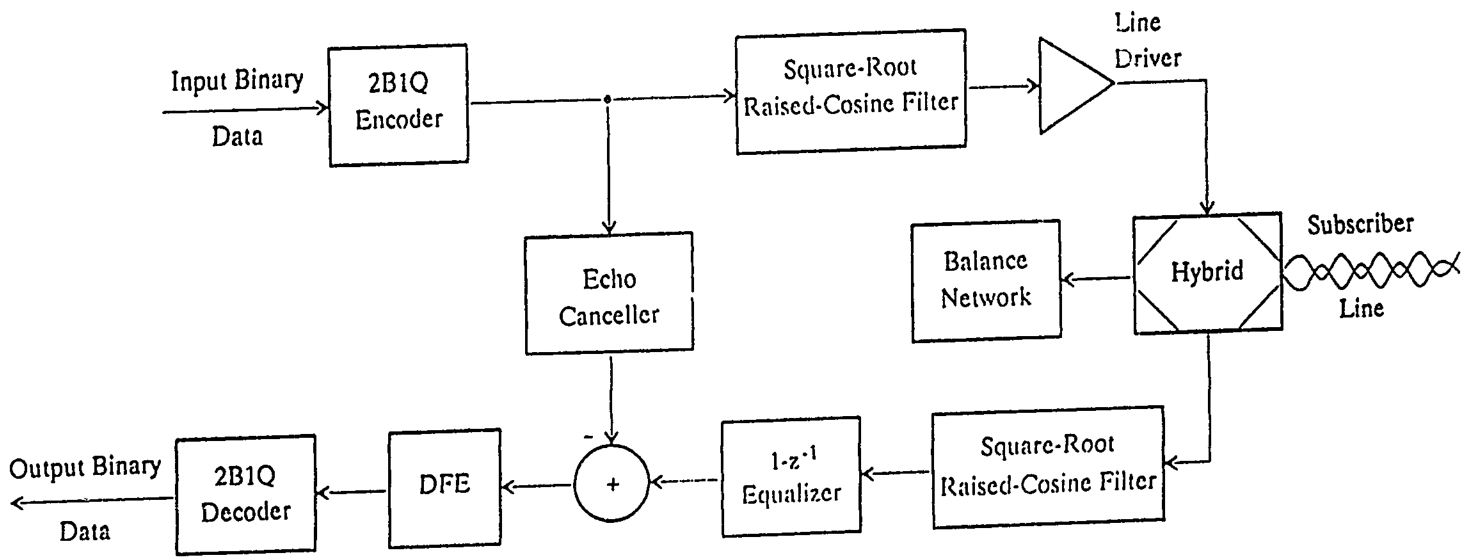


Fig. 4.7 Block diagram of the transceiver assumed in the simulation

raised cosine filter is used as the transmit and receive filter, and its transfer function is shown below:

$$H(f) = \begin{cases} 1 & |f| \leq (1-\alpha)/2T \\ \left(0.5 \left(1-\sin\left(\frac{\pi T}{\alpha} \left(|f| - \frac{1}{2T} \right) \right) \right) \right)^{0.5} & (1-\alpha)/2T < |f| \leq (1+\alpha)/2T \\ 0 & (1+\alpha)/2T < |f| \end{cases} \quad (4.9)$$

T is the symbol period and $0 \leq \alpha \leq 1$ is the roll-off factor. The transmit filter shapes the spectrum of the transmitted signal, and bandlimits it to reduce the amount of crosstalk power. The receive filter removes the excessive noise in the received signal, and at the same time acts as an anti-aliasing filter. The bandwidth of the system is controlled by the receive filter which has a bandwidth of $(1+\alpha)/2T$. Moreover, the transmit and receive filter together will provide spectral shaping which satisfies the first Nyquist's criterion of zero intersymbol interference. The value of α controls the amount of excess bandwidth, and there is an optimum value of α which reduces the amount of noise power and facilitates the operation of the timing recovery circuit.

The electronic hybrid circuit with its 5 element balance network as shown in Fig. 2.3 is used in this simulation study. The local echo is further suppressed by an echo canceller which is assumed to provide 40 and 65 dB reduction of the echo level in two independent studies. An ideal 1:1 transformer is used at the interface to couple the signal from the line to the hybrid. The rejection of the dc component by the transformer causes the system impulse response to have a long tail.

The multiple response (MR) fixed preequalizer with transfer function $H(z) = 1-z^{-1}$ is introduced in the receiver path. It will shorten the length of the system impulse response through dc restoration, and the number of taps required by the echo canceller and the DFE will be reduced [22]. The additional distortion introduced by this preequalizer can be removed completely by the DFE which follows. An adaptive decision-feedback equalizer is used in the transceiver. The LMS adaptation algorithm is used to train the DFE. The performance of the DFE is better than of the linear equalizer, and its simplicity of implementation makes it a proper choice for the equalizer.

4.3 Transceiver simulation model

Using the transceiver structure as mentioned in the last section, the models of the signal and noise can be built. The simulation structure is shown in Fig 4.8. There are four paths in the diagram which simulate the signal, echo, near-end crosstalk and coloured Gaussian noise. The far-end crosstalk and the impulse noise are not simulated, and the reasons will be explained in the following subsection. The signal and different noise samples are generated separately, and are then added and scaled to a proper level by the automatic gain control device, and finally passed through an adaptive DFE. The details of the modelling will be discussed in the following sections.

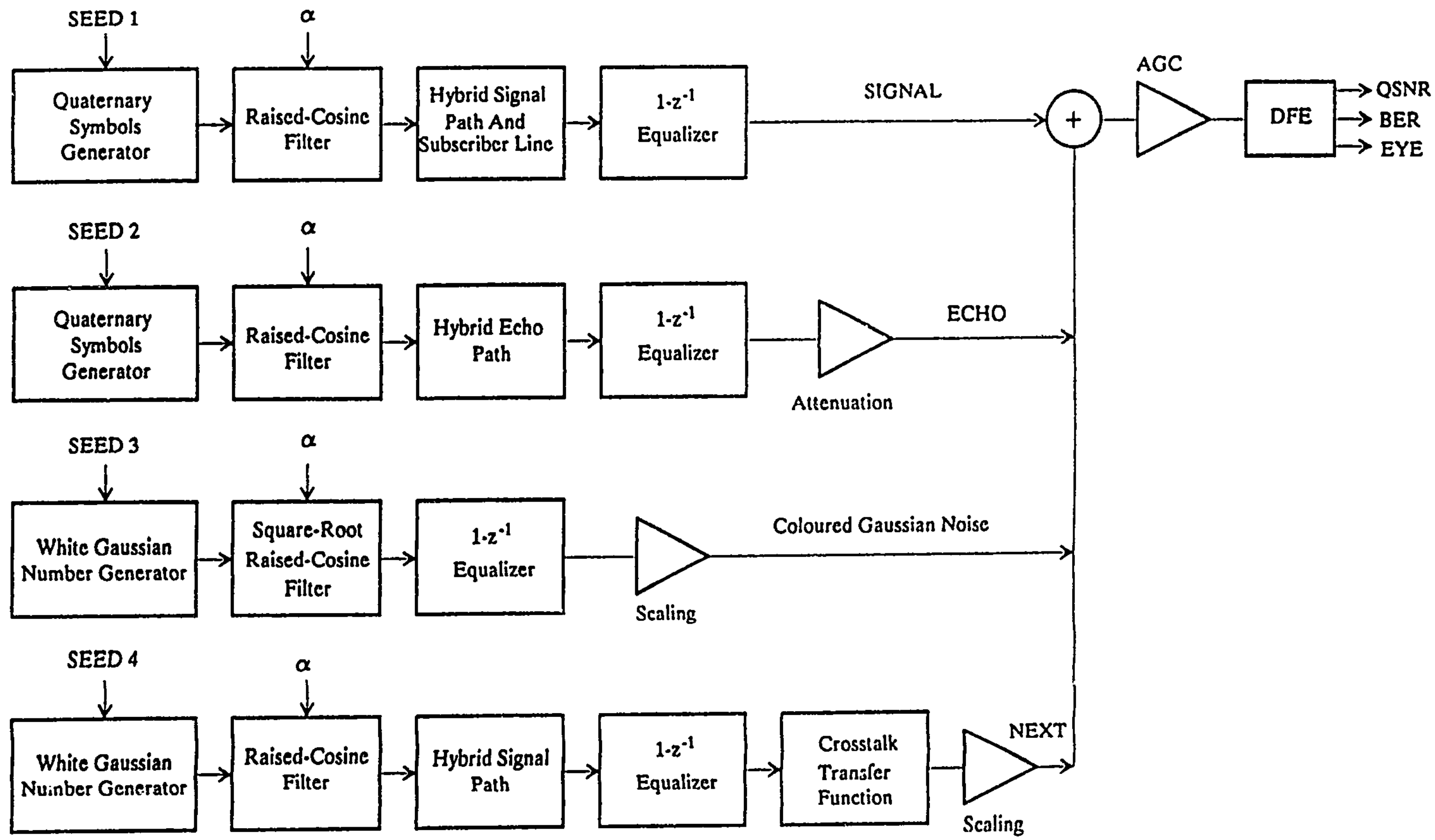


Fig. 4.8 Simulated signal processing structure for one direction of transmission

4.3.1 Far-end crosstalk and impulse noise

The disturbances that will affect the transmission performance are discussed in chapter 2. In the simulation, the far-end crosstalk and impulse noise are neglected and the reasons are explained below.

The power transfer function of NEXT and FEXT can be expressed as

$$NEXT\ (dB) = 10\log_{10}(K_{NEXT}) + 15\log_{10}(f)$$
$$FEXT\ (dB) = 10\log_{10}(K_{FEXT}) + 20\log_{10}(f) + 10\log_{10}(L) \qquad (4.10)$$

where K_{NEXT} and K_{FEXT} are the near-end and far-end crosstalk coupling constants respectively, f is the frequency, and L is the length of the coupling path. The NEXT coupling loss decreases with frequency by 4.5 dB/octave, while the FEXT decreases by 6 dB/octave. As a result, the frequency range in which a system operates will determine whether NEXT or FEXT is dominant. For basic rate access, FEXT will be negligible because the transmission rate is low. The RA-DSL transmission method will maximize the capacity of each loop; however, we can show that FEXT is still insignificant on short loops where the transmission rate is high.

Fig. 4.9 and 4.10 show the distribution of pair-to-pair NEXT and FEXT coupling loss respectively [23]. The foam skin insulated (FS) cable has electrical characteristics similar to the polyethelene insulated cable (PIC), and we will refer the FS cable in the figures. From the figures, the 99% NEXT loss is 80 dB and the 99% FEXT loss is 101 dB at 160 kHz. Using equation 4.10, we can calculate the 99% NEXT loss to be 84.5 dB and the 99% FEXT loss to be 107 dB at 80 kHz (bandwidth required for basic rate access using 2B1Q code). The difference will be 22.5 dB,

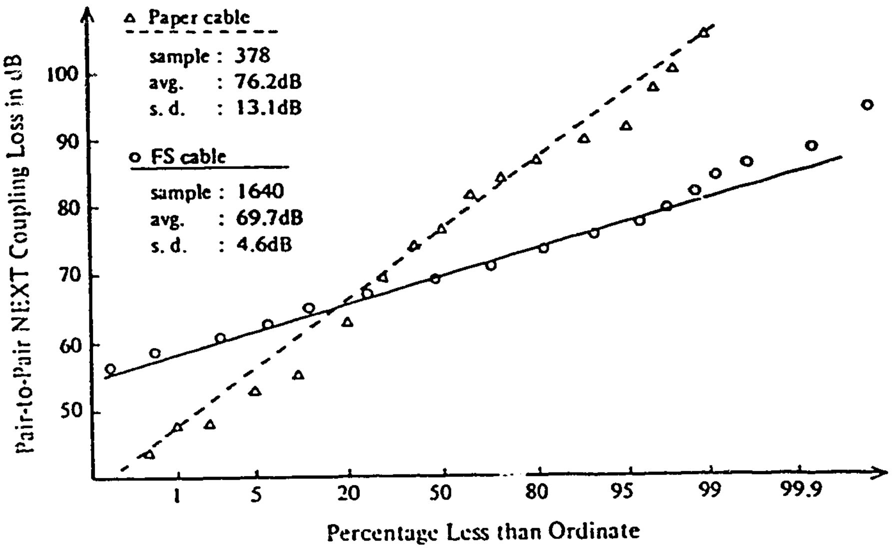


Fig. 4.9 Distribution of NEXT coupling loss (pair-to-pair, within 100-pair unit at 160 kHz)

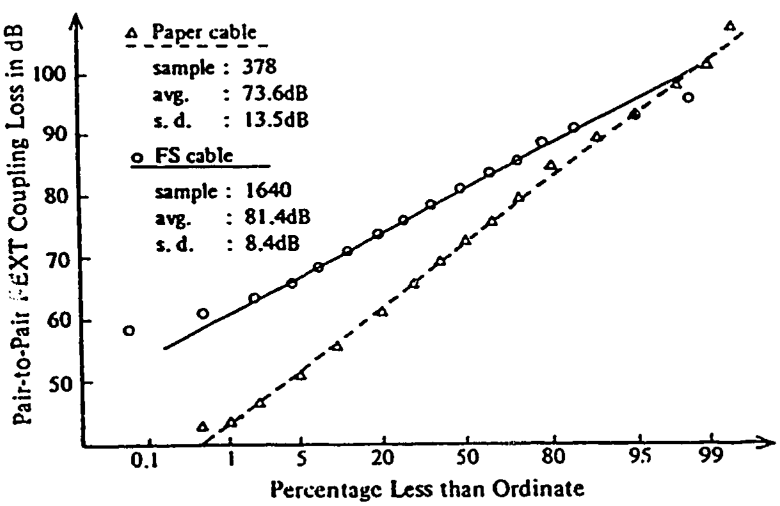


Fig. 4.10 Distribution of FEXT coupling loss (pair-to-pair, within 100-pair unit at 160 kHz and 1 km)

and thus FEXT is negligible comparing to NEXT at basic access rate. Similarly, the 99% NEXT loss can be calculated to be 69.8 dB and the 99% FEXT loss to be 87.4 dB at 768 kHz (maximum bandwidth required for RA-DSL transmission). The difference will now be 17.6 dB, which is still big enough to neglect the FEXT. Therefore, for the frequency range of interest, FEXT is negligible in the presence of NEXT.

The above calculation assumed that the FEXT coupling length is 1 km. For shorter coupling length, the FEXT coupling loss will be bigger and thus FEXT is insignificant. For longer coupling length, the FEXT coupling loss decreases. However, the transmission rate that is possible on the loop will also be smaller, FEXT will still be smaller than NEXT in spite of the increase in coupling length. For instance, we can compare the NEXT and FEXT coupling loss at 80 kHz (basic rate access) on a loop with coupling length of 10 km. The 99% NEXT loss can be calculated to be 84.5 dB and the 99% FEXT loss to be 97 dB. The difference is 12.5 dB, and NEXT is still marginally larger than FEXT in this conservative case. For the last case, we will compare the NEXT and FEXT coupling loss at 768 kHz on a loop with coupling length of 3 km. The 99% NEXT loss can be calculated to be 69.8 dB and the 99% FEXT loss to be 82.6 dB. The difference is 12.8 dB, and NEXT is still marginally larger than FEXT. From the above analysis, we can say that the FEXT is insignificant compared to NEXT, and thus it can be omitted in the simulation.

Impulse noise is one important factor which could limit the reach of the digital subscriber loop. However, there is little known about the characteristics of impulse noise. When ISDN was introduced, most of the literature discussed the problem of crosstalk, and impulse noise was simply omitted. Because of the lack of available measured data on impulse noise, it is impossible to deduce a model for the simulation of impulse noise. As a result, the impulse noise is not included in the simulation. If more is known about the characteristics of impulse noise in the future, it will be worthwhile to include the impulse noise model in this simulation software.

4.3.2 Signal model

When the signal is transmitted from the central office to the subscriber (or vice versa), it will pass through the square-root raised cosine transmitter filter, the line driver, the hybrid, the subscriber line, the square-root raised cosine receiver filter, and the MR preequalizer in sequence. All of these are linear devices, therefore they can be grouped together as one linear channel. These linear devices are represented in the frequency domain by their transfer functions, and the composite frequency response is obtained by multiplying all transfer functions. The transfer functions of these devices are shown in Appendix C.

To calculate the impulse response of the signal path, two unit impulses of opposite polarity are sent. This pattern produces a balanced signal with no dc offset. A data array of 2048 points (N) with a sampling resolution of 16 points/symbol is used. The positive impulse is located at point 513, and the negative impulse is located at point 1537. These two impulses are maximally separated so that their impulse response will die out before they interfere with each other. The spectrum of these impulses is calculated using the fast Fourier transform (FFT). The FFT routine will produce frequency domain components of a signal at discrete frequencies. The frequency spacing (Δf) of these components is given by:

$$\Delta f = \text{sampling frequency} / N$$

where sampling frequency = symbol rate * number of sample points/symbol

$$N = \text{number of sample points/symbol} * \text{number of symbols}$$

$$\therefore \quad \Delta f = \text{symbol rate} / \text{number of symbols}$$

The nature of the RA-DSL transmission method makes the symbol rate vary, and the number of symbols = 128 (2048 sample points / 16 sample points per symbol). The impulse response is obtained by taking the inverse fast Fourier transform (IFFT) of the product of the impulse spectrum and the signal path transfer function.

The positive unit impulse response is chosen from the data array to retrieve the timing information. The impulse response has a sampling resolution of 16 points/symbol, which is sufficient to define the impulse waveform accurately. The transmitted symbols (+3, +1, -1, -3) are generated using a quaternary number generator, and the received signal is calculated by convolving the generated quaternary symbols with the signal path impulse response. Because the input to the DFE is sampled at the baud rate, instead of convolving the impulse response with the quaternary symbols at 16 samples/symbol, the resolution can be reduced to 1 sample/symbol without loss of information. For instance, the impulse response can be sampled at baud rate, and the problem is to determine the proper timing phase.

The timing phase can be selected in two ways. First, the cursor is selected at the maximum value of the impulse response. Second, the cursor is selected such that the value of the first (immediately preceding) precursor is minimized. Although the second condition causes a loss of signal power, it is justified because the DFE has no forward filter for linear equalization of the precursor ISI. By minimizing the first precursor (usually the most dominant), the precursor ISI can be reduced prominently. The criterion of minimizing the first precursor is used in the simulation study because it provides better results, if linear equalization of the precursors is not used.

After choosing the proper sampling phase, the impulse response is sampled at the baud rate. This is an efficient method to generate the signal because only the samples that are fed into the DFE are calculated. This method is possible because impulses instead of waveforms are used to convolve with the signal path impulse response. A different approach is used to calculate the NEXT and coloured Gaussian noise, when the input can not be represented by impulses.

4.3.3 Local echo model

The local echo path is formed similarly to the signal path, and the transfer function of each device is shown in Appendix C. A real echo canceller (transversal filter) is not modelled, but the function of an echo canceller is included in the model by attenuating a prespecified portion of the echo before adding it to the received signal. In addition, echo cancellation is constant at all frequencies. In general, the echo impulse responses on the central office side and the subscriber side are different unless a homogeneous line is used. The local echo impulse response has 16 samples/symbol which can similarly be reduced to 1 sample/symbol. Instead of synchronizing the echo impulse response and the signal impulse response, the sampling phase is chosen to maximize the echo (worst case analysis). This can be accomplished by sampling the maximum value of the echo impulse response. The residual echo after echo cancellation is calculated by convolving the generated quaternary symbols with the echo impulse response.

4.3.4 Near-end crosstalk noise model

The NEXT noise path is shown in Fig. 4.8, and the transfer function of each device is shown in Appendix C. Only NEXT noise is considered because it is more dominant than the FEXT noise (explained in section 4.3.1). Theoretically, to simulate the NEXT noise for an N pair cable, NEXT noise from the N-1 disturbers has to be calculated. This method is not practical because it is too time consuming. By assuming that the NEXT noise contributed from the N-1 disturbers is Gaussian for large N due to the Central Limit Theorem, the calculation is simplified. The NEXT noise can be calculated by convolving white Gaussian noise samples which are generated from a zero-mean unit-variance Gaussian number generator with the NEXT noise impulse response, and then scaling the noise power to a desired level. This method is possible because spectral shaping of a Gaussian process will not change its statistics. The assumption of the NEXT noise with a Gaussian distribution agrees fairly good with the actual noise data except the actual noise distribution exhibits a much shorter tail [24]. For instance, the occurrence of large amplitude NEXT is more likely in the Gaussian case, which is a more conservative assumption. This discrepancy is acceptable and provides considerable gain in computational time.

The NEXT impulse response has a resolution of 16 samples/symbol, but unlike for the signal and the echo, it can not be reduced to 1 sample/symbol. The NEXT noise is band-limited by the square-root raised cosine receiver filter to the frequency 1/T (symbol rate). In order to prevent aliasing of the NEXT noise spectrum, the NEXT impulse response needs at least 2 samples/symbol to convolve with the input Gaussian samples. As a result, the NEXT impulse response is resampled at 2 points/symbol. The NEXT noises have random phases due to their nature as summation of the N-1 disturbing signals. The sampling phase of the NEXT impulse response is not important and the cursor is set at its maximum value. The output NEXT samples are calculated at the baud rate though the input Gaussian noise samples are generated at twice the baud rate. These noise samples have the desired spectra and probability distribution, but they are not scaled to a proper power level. The method to scale these noise samples will be discussed in a later section.

4.3.5 Coloured Gaussian noise model

The coloured Gaussian noise path is shown in Fig. 4.8. It enters the receiver via the load resistance. The noise samples are generated by convolving the white Gaussian samples with the noise impulse response, and scaled to proper noise power level. The impulse response is sampled at 2 points/symbol, and the cursor is chosen at the maximum value of the impulse response. The output coloured Gaussian noise samples are calculated at the baud rate though the input Gaussian samples are generated at twice the baud rate. The power scaling for these noise samples will be discussed in a later section.

4.3.6 Transmitted power at interface

The ISDN standard specifies the transmitted power at the line and network interfaces to be between 13.0 and 14.0 dBm in the frequency band from 0 Hz to 80 kHz, with a nominal average power of 13.5 dBm. Although the RA-DSL transmission method may not need to adhere to the ISDN standard, an average transmitted power of 13.5 dBm is used in the simulation. An increase in the transmitted power will increase the echo and the crosstalk noise power accordingly. The only improvement is that the signal to coloured Gaussian noise ratio increases, which is insignificant to enhance the system performance. As a result, it is better to follow the ISDN standard.

The transmitted signal path consists of the square-root raised cosine filter, the line driver, and the hybrid. The transfer functions of these devices are shown in Appendix C. A proper gain factor for the line driver will be calculated so that an average transmitted power of 13.5 dBm is achieved at the line/network interface. This is done in two steps. First, if the symbols transmitted are equiprobable, then the variance σ^2 for the 2B1Q code is given by:

$$\sigma^2 = (A^2 + (-A)^2 + (3A)^2 + (-3A)^2)/4 = 5A^2 \qquad (4.11)$$

where A is the amplitude for the transmitted symbol '1'. For measurement purposes, the termination impedance is 135 ohms resistive as specified by the ISDN standard. The transmitted power X in dBm is then given by:

$$X\ (\mathrm{dBm}) = 10\log_{10}(5A^2 * 1000/135) \qquad (4.12)$$

For X = 13.5 dBm, the transmitted amplitude A should be 0.777 V. Second, the transmitted voltage at the line/network interface will be scaled down by the impedance network of the hybrid circuit (the hybrid circuit is shown in Fig. B.1). The transmitted power at the line/network interface will be less than 13.5 dBm even if the proper voltage level is transmitted. The transmit amplitude has to be multiplied by a proper gain factor to compensate for the power loss. Two unit impulses of opposite polarity are sent, and the power of the impulses at the transmitter and at the interface are calculated. The powers in both cases are made equal by finding a suitable gain factor. The gain of the line driver to provide 13.5 dBm transmitted power at the interface is then given by:

$$\text{Gain of line driver} = 0.777*\text{Sqrt}(\text{power at transmitter}/\text{power at interface}) \quad (4.13)$$

The first factor will give the proper voltage level for a symbol '1', and the second factor will preserve the power loss at the hybrid circuit. The square-root is needed for voltage scaling. The line impedance as seen from the hybrid will generally be different at the line and network termination, and therefore the gain of the line driver will be different at both interfaces.

4.3.7 Power scaling of NEXT and coloured Gaussian noise

The methods of generating the NEXT and the coloured Gaussian noise samples were discussed in the previous sections. But these noise samples are not useful unless they are properly scaled. Because the noise samples are calculated by convolving unit-variance gaussian samples with the noise impulse responses, the power can be scaled by multiplying the noise impulse responses by a proper scaling factor, similarly to the signal level scaling at the line interface.

The NEXT noise is assumed to originate from the line interface, and to be coupled to the disturbed system through the NEXT transfer function. As a result, the NEXT noise source also has 13.5 dBm power at the line interface. The numbers generated from the Gaussian number generator have unit variance. If the variance is scaled to 5 such that it has the same variance as the transmitted symbols, then the problem is similar to signal power scaling at the line interface. Therefore, the NEXT scaling factor will be:

$$\text{NEXT scaling factor} = \text{Sqrt}(5) * \text{Gain of line driver} \qquad (4.14)$$

The square-root is needed because voltage scaling is used, and the gain of the line driver is different at both directions of transmission.

White Gaussian noise originating from the 135 ohms load resistance at the hybrid becomes coloured through filtering by the square-root raised cosine receiver filter and the $1\text{-}z^{-1}$ equalizer. If the white noise power spectral density is $N_0/2$, then the power P available after filtering is given by:

$$P = 2 \int_0^{BW} N_0/2 \, |H(f)|^2 \, df \qquad (4.15)$$

where H(f) is the composite transfer function of the two filters, and BW is the bandwidth of the system (a function of the roll-off factor α of the square-root raised cosine receive filter). After some manipulations, the expression is surprisingly simple.

$$P = N_0/T \qquad (4.16)$$

where 1/T is the transmitted symbol rate, and the power is not affected by the value of α. Knowing that $P = V^2/R$, the scaling factor for the coloured noise is then given by:

$$\text{Coloured noise scaling factor} = \mathrm{Sqrt}(N_0/T * R_{Load}) \qquad (4.17)$$

where R_{Load} is 135 ohms.

4.3.8 Length of impulse response

In the previous sections, the signal and different noise samples are generated via their impulse responses. An important aspect that has not been mentioned is to determine the length of the impulse response. In order to generate the signal or noise accurately, the ISI contributed by the neighbouring symbols must be accounted for. Ideally, the impulse response must be infinitely long to generate all the ISI terms, which is not possible to implement. The solution is to choose sufficient number of terms to approximate the impulse response so that the error is negligible. One possible way is to compare the value of the main sample to the values of precursor and postcursor samples successively, so that when their ratio is 60 dB (an arbitrary value), we decided that the required length has been reached. The problem is that when the impulse response is oscillatory, then the precursor and postcursor samples may prematurely hit zero or a very small value, and the required length will not be approximated correctly. A better approach is to compare the areas under the impulse response following the main sample and the precursor and postcursor samples within a symbol period so that the problem of zero-crossing can be eliminated. When the ratio is less than an assigned value, then the length required will be determined. Mathematically, this can be expressed as:

$$\text{Ratio} = \sum_{i=1}^{16} |X_i| \ / \sum_{i=1}^{16} |X_{i+16n}| \qquad n = \pm 1,2,3,\ldots \qquad (4.18)$$

where X_i are the samples of the impulse response, and X_1 is the main sample. The value of i extends from 1 to 16 because there are 16 samples in 1 symbol period. Positive values of n correspond to postcursor, and negative to precursor samples respectively. This algorithm is implemented in a subroutine so that the required length of an impulse response can be determined automatically. The maximum length will be set to 20 symbol periods for precursor samples, and 40 symbol periods for the postcursor ones if the ratio of the main sample to precursor and postcursor ones does not meet the predefined value earlier. By limiting the precursor and postcursor samples to a maximum of 60 symbol periods, we can reduce the time spent on convolution while maintaining reasonable numerical accuracy.

4.4 Training adaptive equalizers

An adaptive DFE equalizer is used to equalize the received signal so that the ISI is removed or reduced. The least mean-squares (LMS) algorithm [19,20] is used to update its taps. Before the DFE can be used to equalize the received signal, it must be trained so that the taps are adapted to the channel. This is done by sending a sequence of known symbols from the transmitter to the receiver. By comparing the difference between the desired and the received signal and updating the taps, the optimum taps setting can eventually be obtained. The time required for the taps to converge is different for each loop, and therefore a method has to be found to determine the convergence of the equalizer. If we calculate the mean tap value of the equalizer by averaging the tap values over a period of time, we will find that the tap values deviate only slightly from the mean if the equalizer is close to convergence. In the simulation program, the tap value of the automatic gain control device (a 1 tap forward filter) is averaged over 500 symbols (arbitrary). If this mean tap value and the old mean tap value (for the previous 500 symbols) are within 0.5% (arbitrary), then the DFE is assumed to have converged.

After the DFE has been trained, it will be switched into the decision-directed mode. In this mode, instead of using the known symbols to update the taps, the symbols determined by the decision device are used. The decision device is a simple threshold detector, and the threshold levels are set at the middle between adjacent symbols. For the 2B1Q code and received levels of +3, +1, -1, and -3 volts, the threshold levels will be set at +2, 0 and -2 volts. Errors detected by the decision device will propagate through the length of the feedback filter. If too large step size is set, the DFE will diverge due to error propagation. Under high signal to noise ratio conditions, which is the usual mode of operation, the problem of error propagation is negligible.

4.5 System performance parameters

The usual parameters of measuring the transmission system performance are the bit-error rate (BER), quantiser signal to noise ratio (QSNR), and the eye diagram. Although the bit error rate is the most important factor in a digital communication system, its evaluation by simulation is time consuming. The SNR and the eye diagram which take less time to evaluate will give us a rough idea about how the system performs. The following subsections will discuss methods of evaluating these performance parameters.

4.5.1 Quantiser signal to noise ratio

A useful parameter to measure the system performance is the signal to noise ratio (SNR). There are two locations where we can define the SNR of the system. We can define the SNR either at the input of the DFE or at the input to the quantiser (decision device) of the DFE. Because the signal samples at the input of the DFE have not been equalized, the SNR thus defined will include the ISI noise power. If we measure the SNR at the input of the quantiser of the DFE, then the ISI noise power has mostly been removed by the DFE. Hence, this SNR known as the quantiser signal to noise ratio (QSNR) is a better reflection of the system performance. As a result, the QSNR will be used in this simulation study. The QSNR can be calculated as:

$$QSNR = \sum_{n=1}^{N} (d_n)^2 \Big/ \sum_{n=1}^{N} (Sig_n - d_n)^2 \qquad (4.19)$$

where d_n are the desired symbols, and Sig_n are the equalized signal samples. The number of samples N used to estimate the QSNR must be large so that the result will be reliable.

Two sample outputs calculated by the FORTRAN program dfeber.for (Appendix D) are shown in Table 4.1. The two loops used are the null loop (loop with zero loop length) and loop #1 from the ANSI ISDN standard [1], and transmission rate of 80 kbaud (basic rate) with 40 dB echo cancellation is assumed. The QSNR at both sides of the loop should be the same for Table 4.1a, but since the symbol sequences transmitted at both sides are generated by a different random number generator, the results are slightly different. The QSNR is better on the central office side as shown in Table 4.1b. The transmitted power at the line interface is very close to 13.5 dBm in both cases.

4.5.2 Eye diagram

An eye diagram consists of many overlaid traces of a signal. If the data symbols are random and independent, it summarizes visually all possible intersymbol interference waveforms. The vertical eye opening indicates the immunity to noise, while the horizontal eye opening indicates the immunity to timing phase jitter.

In order to display the eye diagram accurately, the resolution of the signal in time must be greater than the single sample per symbol used in the DFE in the basic simulation. For purpose of constructing an eye diagram from the simulation, the sampling rate of impulse responses of the signal and noises are sampled 8 times per symbol. The signal and noise samples are calculated as before, and the eye diagram is taken just before the decision device of the DFE. After the DFE is trained,

(a) TRANSMITTED SYMBOLS = 4000

CO :

SYMBOL 3 =	959	SYMBOL 1 =	1010
SYMBOL -1 =	1020	SYMBOL -3 =	1011

TRANSMITTED POWER(dBm) = 13.443770
RECEIVED POWER = 3.876915
TOTAL NOISE POWER = 3.607664E-05
WNOISE POWER = 5.581560E-07
NEXT POWER = 1.705676E-06
ECHO POWER = 3.369276E-05
QSNR(dB) = 36.32207

SUB :

SYMBOL 3 =	1042	SYMBOL 1 =	1001
SYMBOL -1 =	986	SYMBOL -3 =	971

TRANSMITTED POWER(dBm) = 13.520580
RECEIVED POWER = 3.657208
TOTAL NOISE POWER = 3.774803E-05
WNOISE POWER = 5.364111E-07
NEXT POWER = 1.769078E-06
ECHO POWER = 3.538254E-05
QSNR(dB) = 36.34091

(b) TRANSMITTED SYMBOLS = 4000

CO :

SYMBOL 3 =	959	SYMBOL 1 =	1010
SYMBOL -1 =	1020	SYMBOL -3 =	1011

TRANSMITTED POWER(dBm) = 13.496460
RECEIVED POWER = 5.613702E-04
TOTAL NOISE POWER = 2.685090E-06
WNOISE POWER = 5.581560E-07
NEXT POWER = 1.223832E-06
ECHO POWER = 9.687222E-07
QSNR(dB) = 18.4972

SUB :

SYMBOL 3 =	1042	SYMBOL 1 =	1001
SYMBOL -1 =	986	SYMBOL -3 =	971

TRANSMITTED POWER(dBm) = 13.473760
RECEIVED POWER = 5.357420E-04
TOTAL NOISE POWER = 3.340057E-06
WNOISE POWER = 5.364111E-07
NEXT POWER = 1.414824E-06
ECHO POWER = 1.389802E-06
QSNR(dB) = 16.66962

Table 4.1 QSNR at basic rate of (a) null loop (b) standard loop #1 the tap values are fixed. The feedback filter will output a train of square pulses whose magnitudes depend on the tap values and the transmitted symbols. This pulse train is subtracted from the signal to remove the postcursor ISI. A total of 256 symbols are plotted, which should be sufficient to illustrate the performance of the system. For the 2B1Q code, there are three eye openings. The percentage eye openings for the top, middle and bottom eyes are calculated. Under ideal conditions (no noise and ideal line), the eye opening should be 100% because Nyquist pulses are used. Fig. 4.11 shows the eye diagram for the null loop and the ANSI standard loop #1. The eye statistics for these two loops are shown in Table 4.2. The eye opening for the null loop is about the same for both directions of transmission, and the eye opening is close to 100%. The eye opening on the central office side for loop #1 is 61%, which is slightly better than the eye opening on the subscriber side.

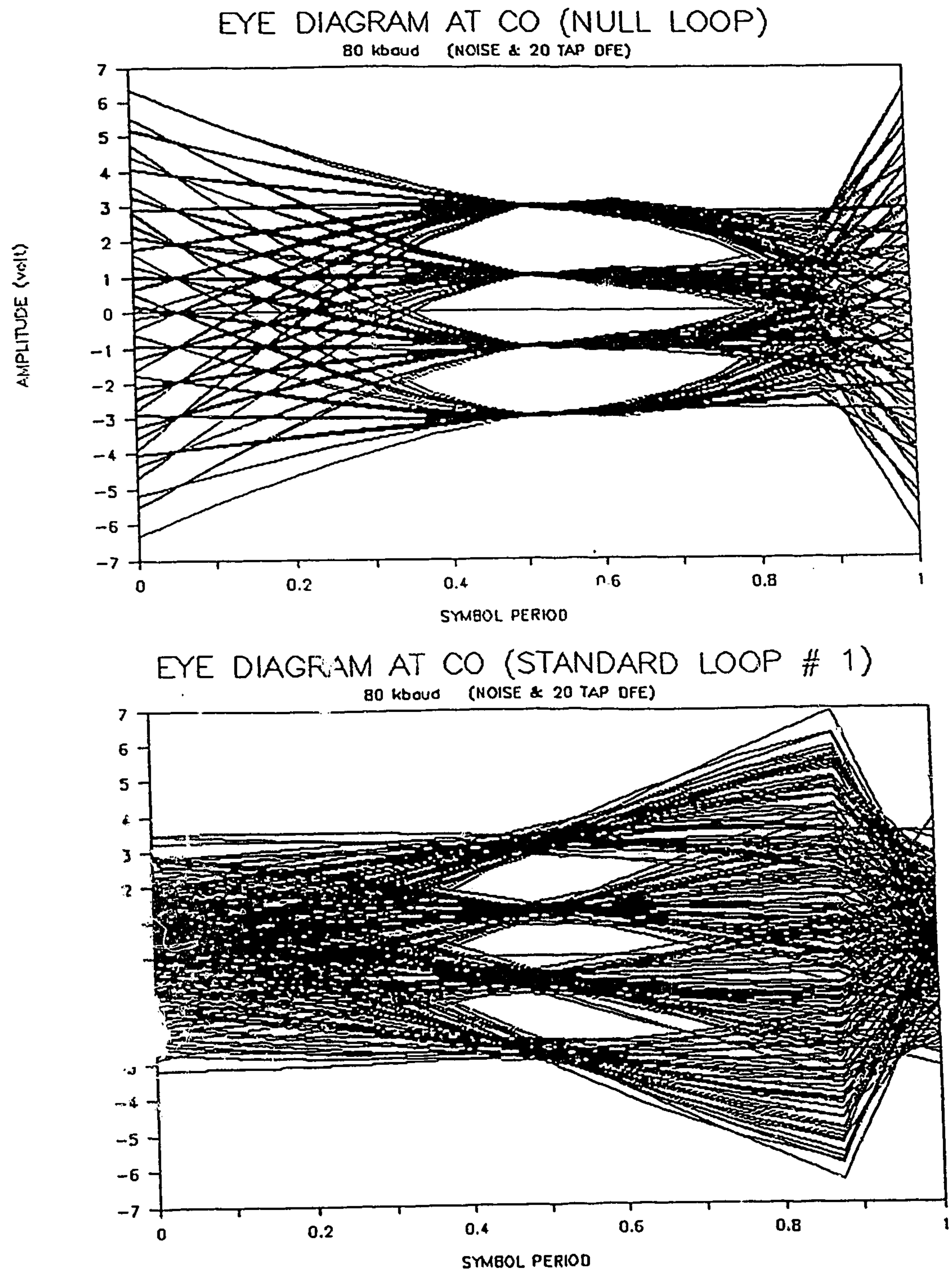


Fig. 4.11 Eye diagram of (a) null loop (b) standard loop #1

(a)

CENTRAL OFFICE :

NUMBER OF ERRORS =	0
TOP EYE OPENING =	93.321520
MIDDLE EYE OPENING =	93.238370
BOTTOM EYE OPENING =	92.734970
AVERAGE EYE OPENING =	93.098280

SUBSCRIBER SIDE :

NUMBER OF ERRORS =	0
TOP EYE OPENING =	94.142760
MIDDLE EYE OPENING =	93.594020
BOTTOM EYE OPENING =	94.171000
AVERAGE EYE OPENING =	93.969260

(b)

CENTRAL OFFICE :

NUMBER OF ERRORS =	0
TOP EYE OPENING =	61.887100
MIDDLE EYE OPENING =	59.982060
BOTTOM EYE OPENING =	61.442740
AVERAGE EYE OPENING =	61.103970

SUBSCRIBER SIDE :

NUMBER OF ERRORS =	0
TOP EYE OPENING =	51.197200
MIDDLE EYE OPENING =	57.483300
BOTTOM EYE OPENING =	53.304420
AVERAGE EYE OPENING =	53.994970

Table 4.2 Eye statistics of (a) null loop (b) standard loop #1

CHAPTER 5

SIMULATION STUDY METHOD AND RESULTS

In the previous chapter, methods used to simulate the subscriber loop and the transceiver were explained. FORTRAN programs were written based on those methods. Routines such as FFT, random number generator, and the cubic spline interpolation were obtained from literature [25]. In this chapter, simulation results on some selected subscriber loops will be presented and discussed. The study is first carried out on an IBM-AT compatible computer, with an Intel 80287 math co-processor. Because of the limited processing power of the AT computer, the simulation study is later carried out on a Sun 3-60 workstation. For calibration, known results from AT were compared to the results of SUN, and the portability of the FORTRAN programs did not cause any problems during the transfer. The computation performance increases by 5 times after the transfer. The setup of the simulation study is described in sections 5.1-5.2, while the results are discussed in section 5.3.

5.1 Selection of loops

The purpose of the simulation study is to investigate the possible improvements in the utilization of the digital subscriber loops using the rate-adaptive transmission method. In order to provide meaningful results for the simulation study, the tested loop configurations must be chosen to represent the loop plant in North America. Thirty five average loops have been chosen to represent all non-loaded loops ranging from 0-10.7 km, with an increment of approximately 300 m for each loop. These average loops are characterized by different lengths, gauges, and number of bridged taps. The configurations of the first eighteen loops ranging from 0-5.5 km are obtained from [21]. Some essential statistics of these loops are shown in Table 5.1. The configurations of loops ranging from 5.5-10.7 km are constructed using the data from the 1981 Bell Canada subscriber loop survey [7]. There is no information in the loop survey about the location of bridged taps in the loop, and therefore bridged taps are located arbitrarily. The actual loop configurations for the 35 loops are shown in Fig. 5.1.

5.2 Bit error rate estimation method

The Monte Carlo method is used to estimate the bit error rate of the system. After the DFE equalizer is trained and set in the decision-directed mode, errors detected will be counted. If N bits are sent and n errors are detected, the estimated BER is given by:

$$p = n/N \tag{5.1}$$

In the limit as $N \to \infty$, the estimated p will converge to the true value P. For finite N, the reliability of the estimator can be expressed in terms of the confidence interval. Given two numbers h_1 and h_2, such that $h_2 \le p \le h_1$, the confidence interval is $h_1 - h_2$. If the confidence level α is also specified, then the following relation can be defined.

$$\mathrm{Prob}[\ h_2 \le p \le h_1\ ] = \alpha \tag{5.2}$$

Fig. 5.2. shows confidence intervals corresponding to confidence levels of 90, 95, and 99 percent for an observed value of p as a function of N [26]. As a rule of thumb, if N is on the order of 10/p, then the confidence interval will be about (2p,0.5p) with a 95% confidence level.

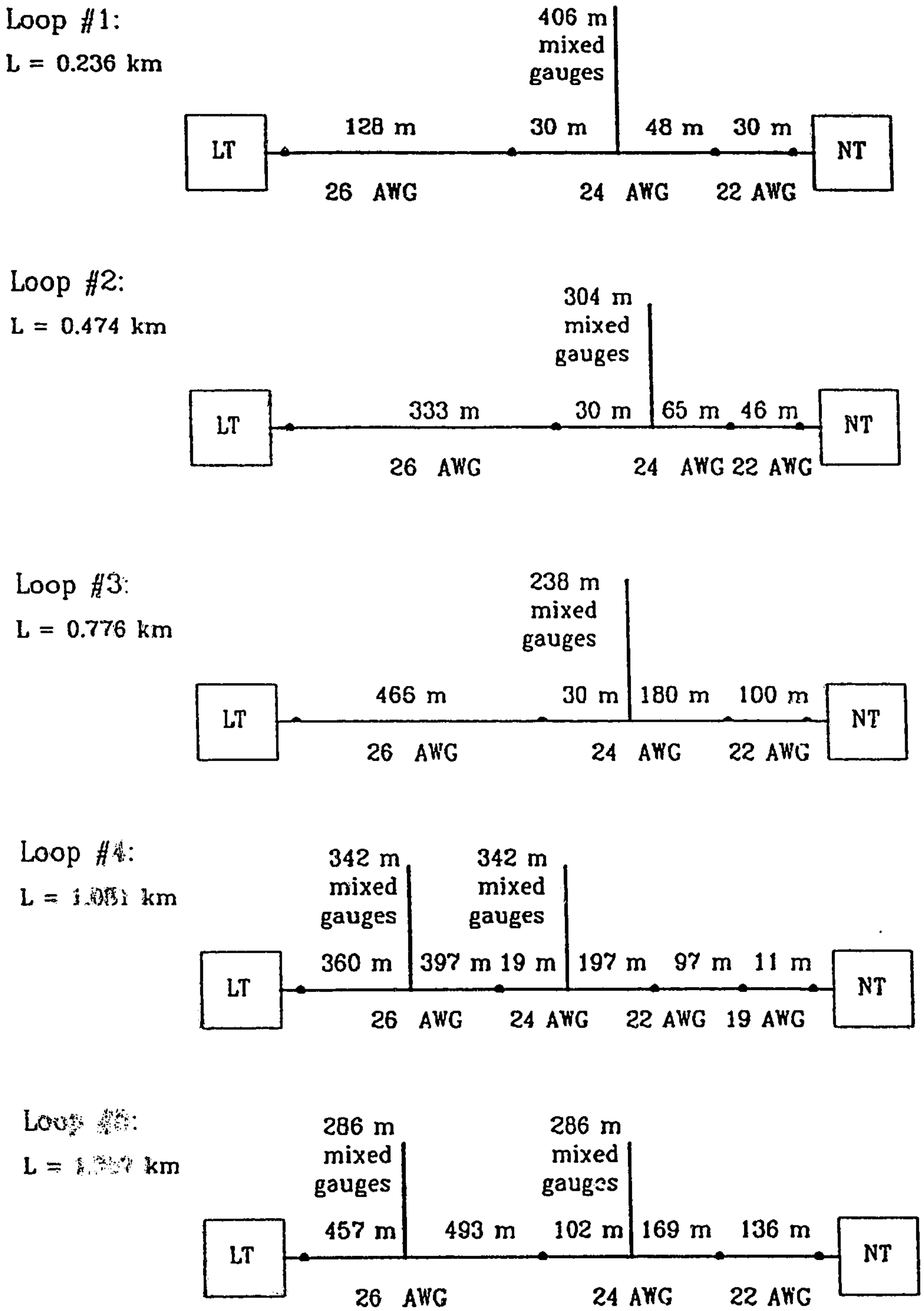
(a)

Loop Length (km)	Average Loop Length	Average No. of Bridged Taps	Average Bridged Tap Length
0 - 0.3048	0.237	1.15	0.406
0.3048 - 0.6096	0.475	1.26	0.303
0.6096 - 0.9144	0.777	1.46	0.238
0.9144 - 1.2192	1.080	1.51	0.342
1.2192 - 1.5240	1.384	1.62	0.286
1.5240 - 1.8288	1.660	1.49	0.293
1.8288 - 2.1336	1.975	1.78	0.226
2.1336 - 2.4384	2.281	1.60	0.292
2.4384 - 2.7432	2.604	1.81	0.276
2.7432 - 3.0480	2.879	1.83	0.259
3.0480 - 3.3528	3.223	1.75	0.257
3.3528 - 3.6576	3.495	1.67	0.333
3.6576 - 3.9624	3.790	1.55	0.286
3.9624 - 4.2672	4.079	1.61	0.284
4.2672 - 4.5720	4.393	1.73	0.280
4.5720 - 4.8768	4.708	1.52	0.282
4.8768 - 5.1816	5.026	2.25	0.310
5.1816 - 5.4864	5.283	2.87	0.191

(b)

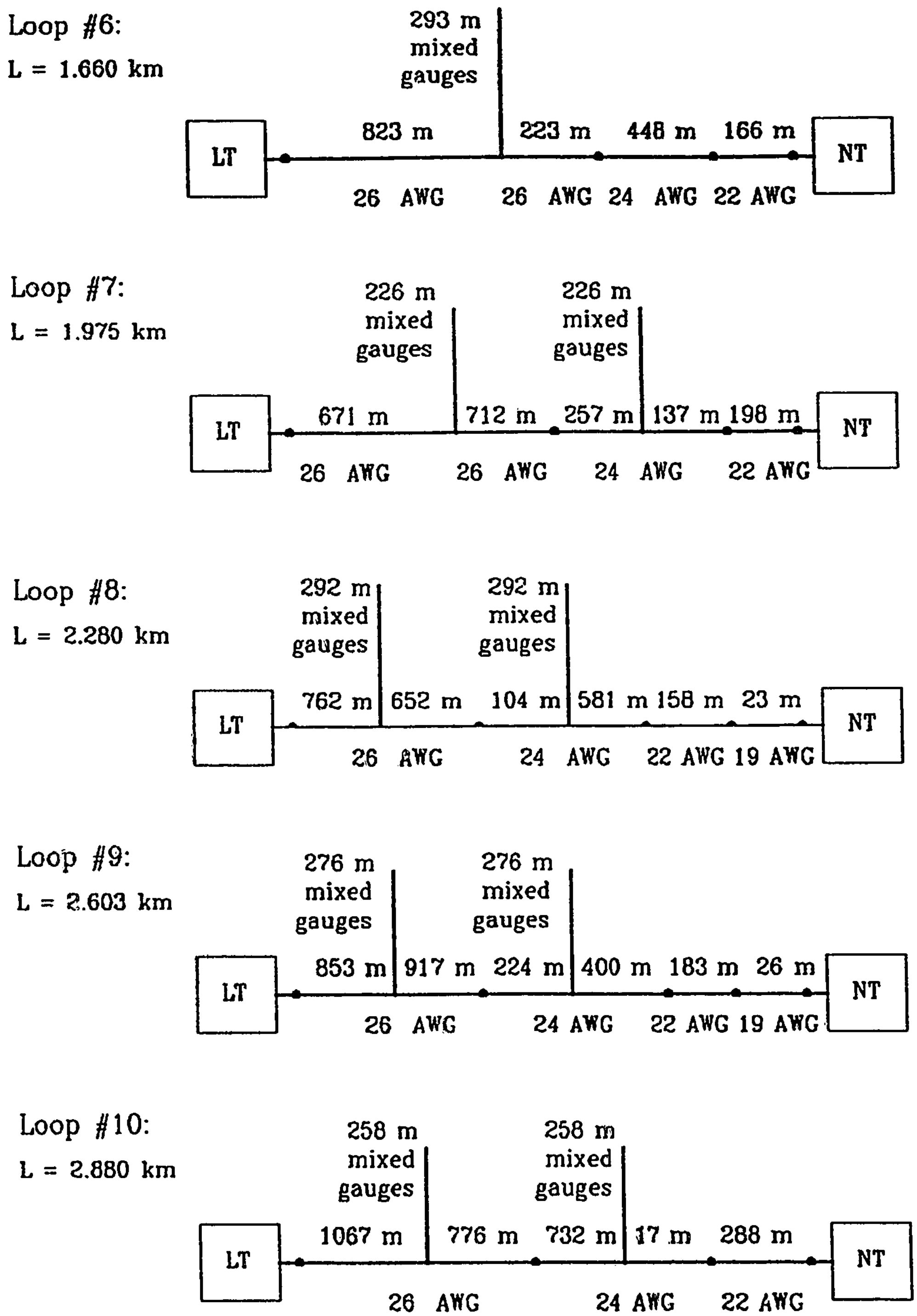
Loop Length (km)	No. of Bridged Taps	No. of Loops, No Bridged Taps	No. of Loops
0 - 0.3048	23	3	20
0.3048 - 0.6096	48	6	38
0.6096 - 0.9144	82	4	56
0.9144 - 1.2192	104	13	69
1.2192 - 1.5240	104	12	64
1.5240 - 1.8288	115	16	77
1.8288 - 2.1336	114	5	64
2.1336 - 2.4384	115	7	72
2.4384 - 2.7432	121	5	67
2.7432 - 3.0480	97	3	53
3.0480 - 3.3528	96	8	55
3.3528 - 3.6576	87	8	52
3.6576 - 3.9624	59	3	38
3.9624 - 4.2672	50	5	31
4.2672 - 4.5720	45	1	26
4.5720 - 4.8768	32	3	21
4.8768 - 5.1816	27	0	12
5.1816 - 5.4864	46	2	16

Table 5.1 Statistics of loop #1 - #18 (a) Average loop length and average bridged tap length distribution (b) gauge number distribution with loop length

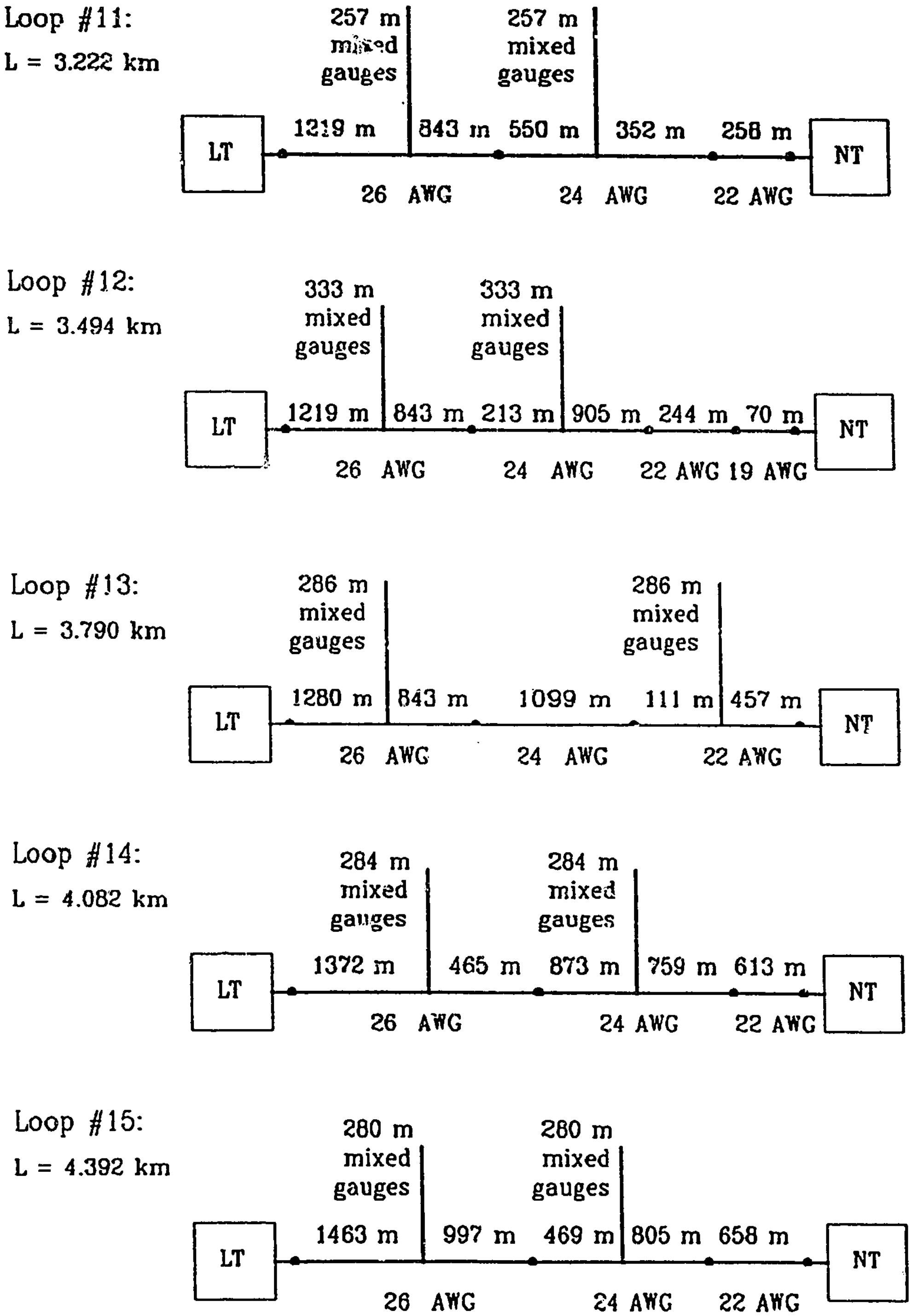


(a) Loops #1 - #5

Fig. 5.1 Loops for testing received signal performance


Loop #6:
L = 1.660 km
293 m mixed gauges
823 m
223 m
448 m
166 m
LT
NT
26 AWG
26 AWG
24 AWG
22 AWG
Loop #7:
L = 1.975 km
226 m mixed gauges
226 m mixed gauges
671 m
712 m
257 m
137 m
198 m
LT
NT
26 AWG
26 AWG
24 AWG
22 AWG
Loop #8:
L = 2.280 km
292 m mixed gauges
292 m mixed gauges
762 m
652 m
104 m
581 m
158 m
23 m
LT
NT
26 AWG
24 AWG
22 AWG
19 AWG
Loop #9:
L = 2.603 km
276 m mixed gauges
276 m mixed gauges
853 m
917 m
224 m
400 m
183 m
26 m
LT
NT
26 AWG
24 AWG
22 AWG
19 AWG
Loop #10:
L = 2.880 km
258 m mixed gauges
258 m mixed gauges
1067 m
776 m
732 m
17 m
288 m
LT
NT
26 AWG
24 AWG
22 AWG

(b) Loops #5 - #10


Loop #11:
L = 3.222 km
257 m mixed gauges
257 m mixed gauges
LT
1219 m
843 m
550 m
352 m
258 m
NT
26 AWG
24 AWG
22 AWG
Loop #12:
L = 3.494 km
333 m mixed gauges
333 m mixed gauges
LT
1219 m
843 m
213 m
905 m
244 m
70 m
NT
26 AWG
24 AWG
22 AWG
19 AWG
Loop #13:
L = 3.790 km
286 m mixed gauges
286 m mixed gauges
LT
1280 m
843 m
1099 m
111 m
457 m
NT
26 AWG
24 AWG
22 AWG
Loop #14:
L = 4.082 km
284 m mixed gauges
284 m mixed gauges
LT
1372 m
465 m
873 m
759 m
613 m
NT
26 AWG
24 AWG
22 AWG
Loop #15:
L = 4.392 km
280 m mixed gauges
280 m mixed gauges
LT
1463 m
997 m
469 m
805 m
658 m
NT
26 AWG
24 AWG
22 AWG

(c) Loops #11 - #15

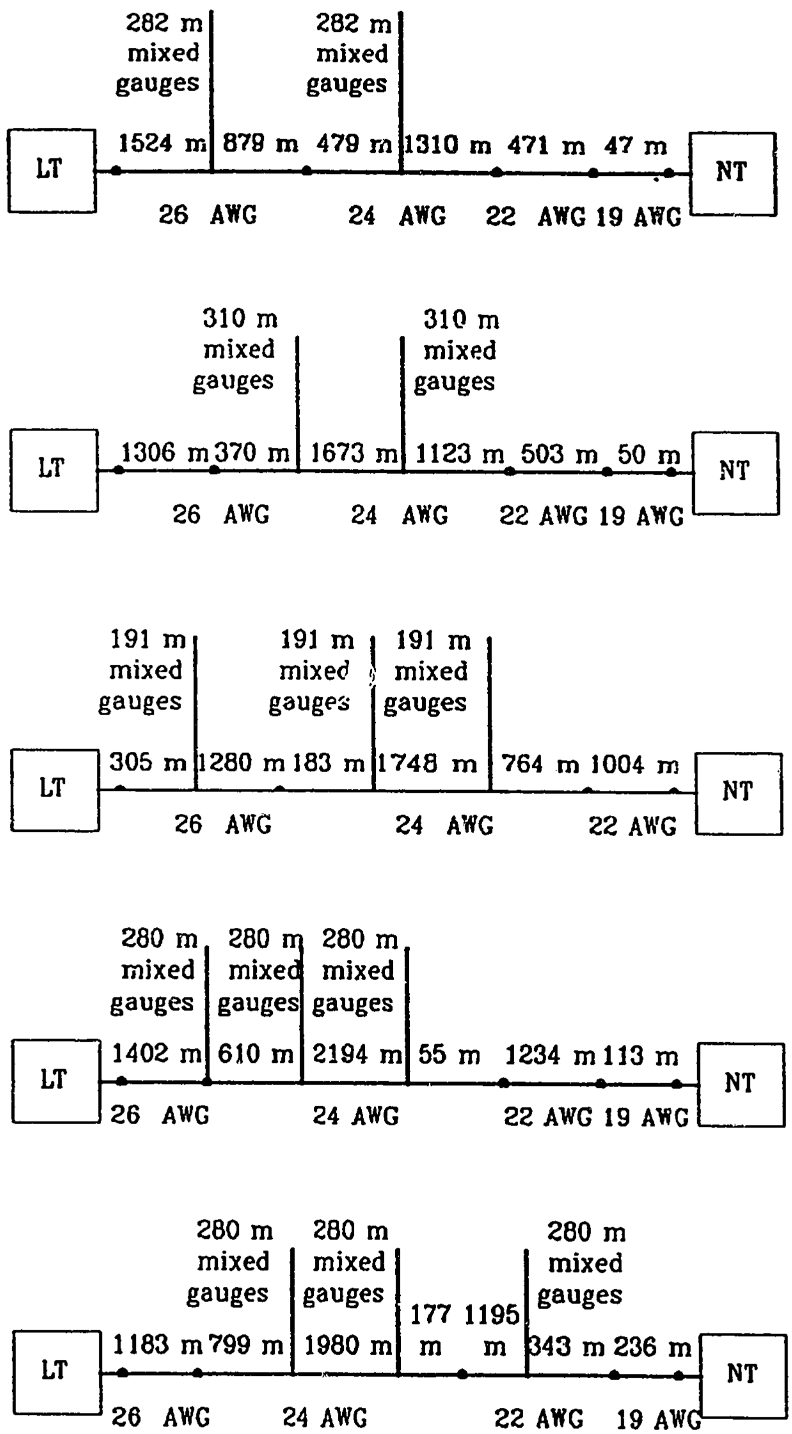
Loop #16:
L = 4.710 km

Loop #17:
L = 5.025 km

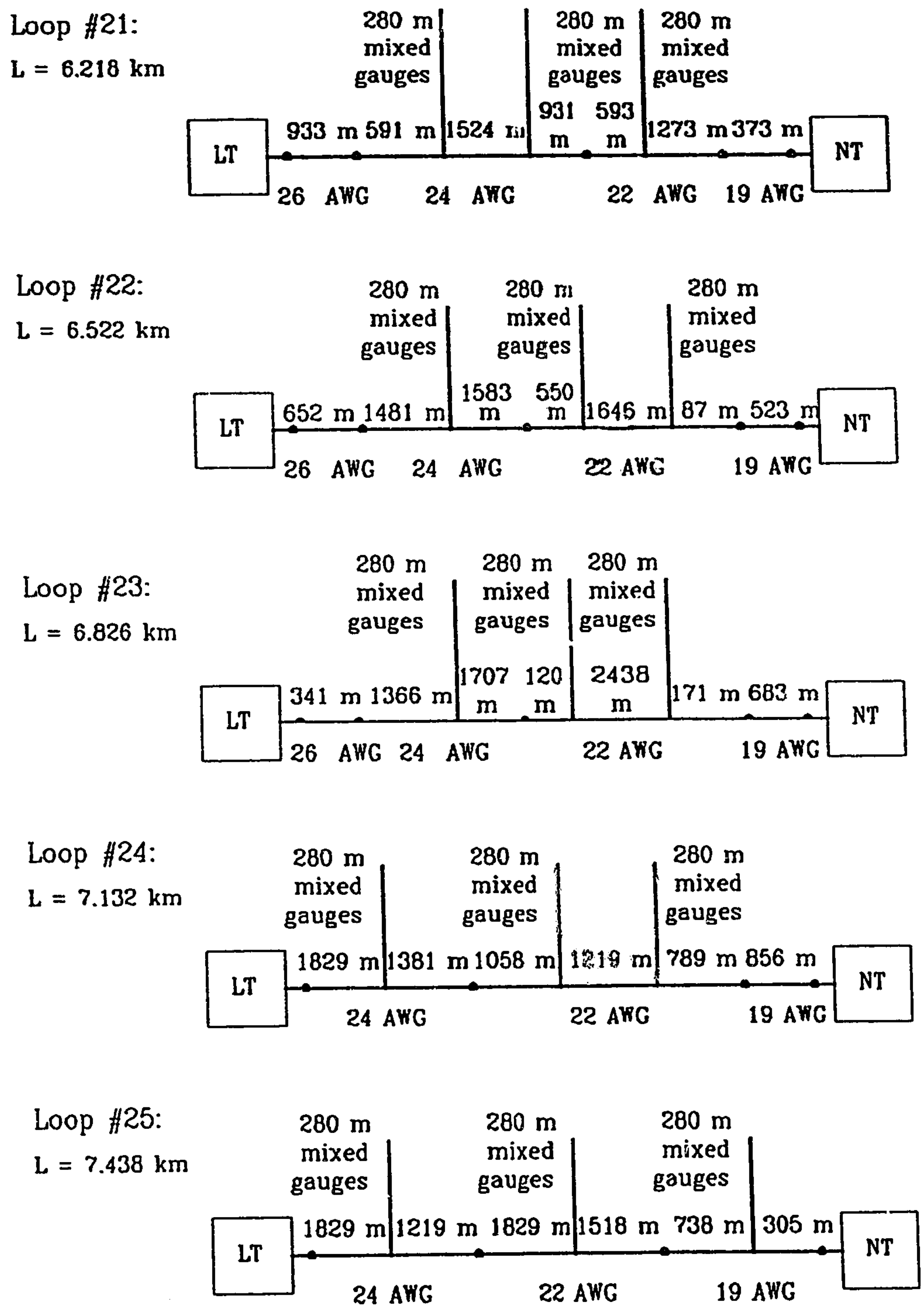
Loop #18:
L = 5.284 km

Loop #19:
L = 5.608 km

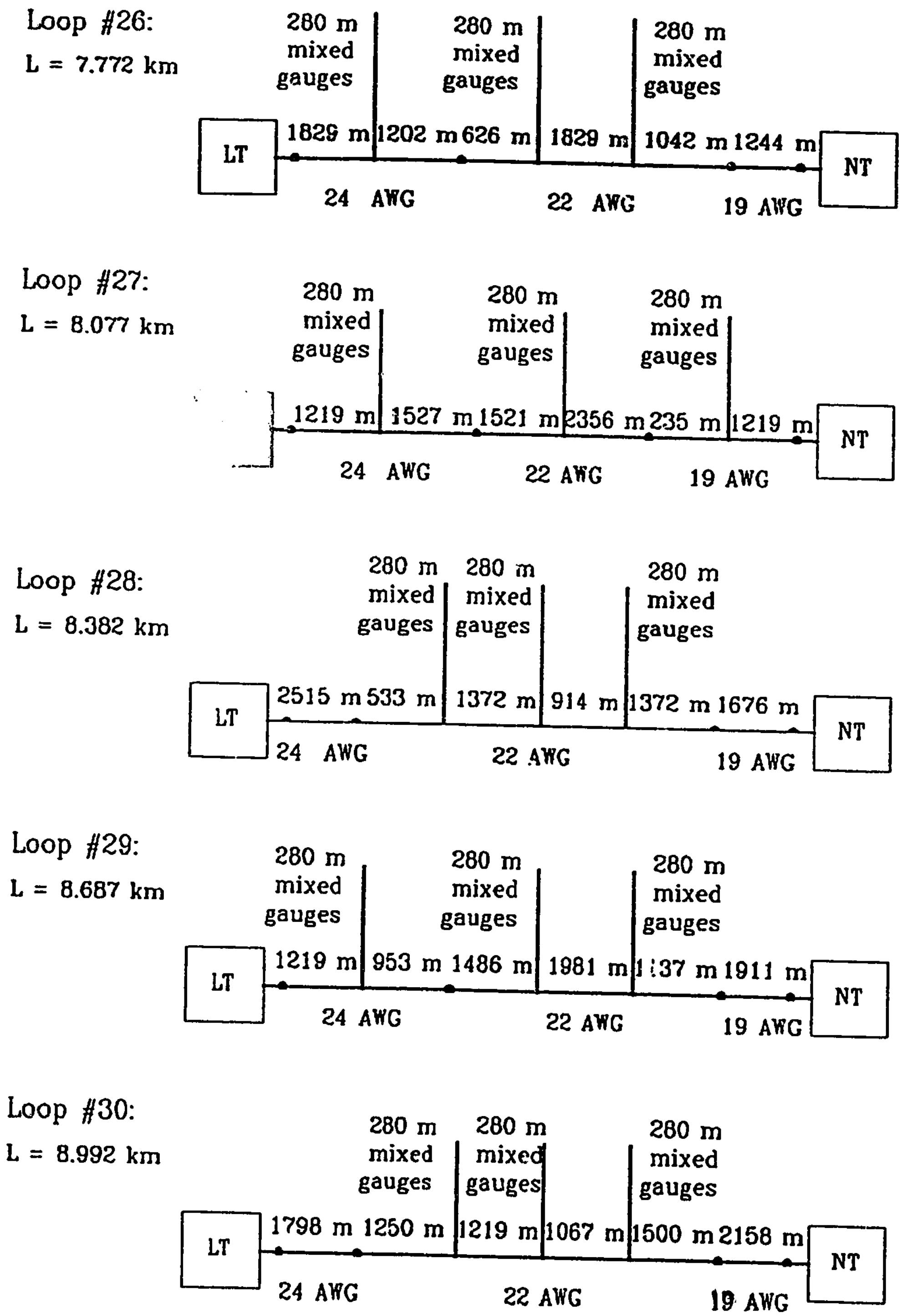
Loop #20:
L = 5.913 km



(d) Loops #16 - #20

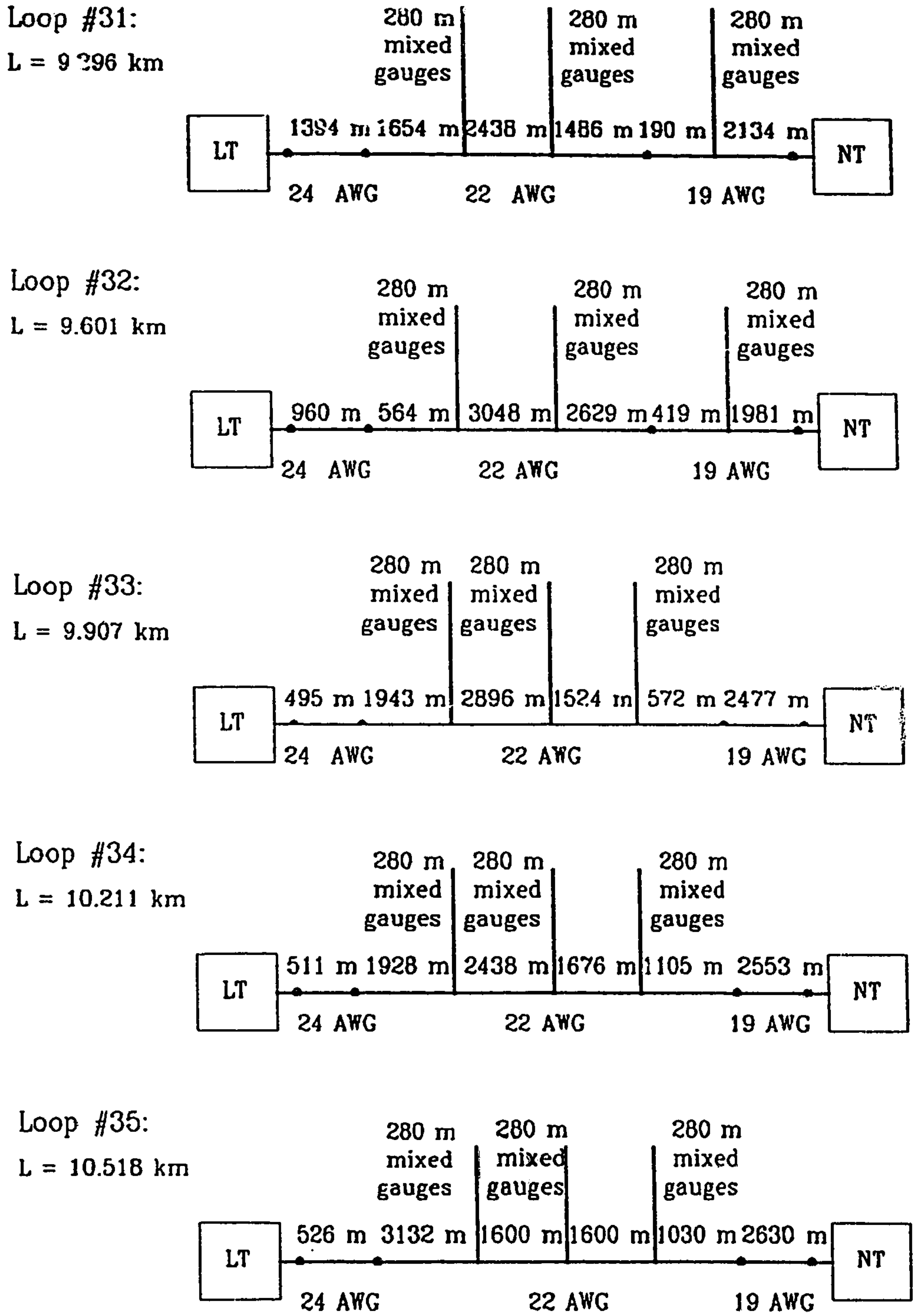

Loop #21:
L = 6.218 km
280 m mixed gauges
280 m mixed gauges
280 m mixed gauges
LT
933 m
591 m
1524 m
931 m
593 m
1273 m
373 m
NT
26 AWG
24 AWG
22 AWG
19 AWG
Loop #22:
L = 6.522 km
280 m mixed gauges
280 m mixed gauges
280 m mixed gauges
LT
652 m
1481 m
1583 m
550 m
1646 m
87 m
523 m
NT
26 AWG
24 AWG
22 AWG
19 AWG
Loop #23:
L = 6.826 km
280 m mixed gauges
280 m mixed gauges
280 m mixed gauges
LT
341 m
1366 m
1707 m
120 m
2438 m
171 m
683 m
NT
26 AWG
24 AWG
22 AWG
19 AWG
Loop #24:
L = 7.132 km
280 m mixed gauges
280 m mixed gauges
280 m mixed gauges
LT
1829 m
1381 m
1058 m
1219 m
789 m
856 m
NT
24 AWG
22 AWG
19 AWG
Loop #25:
L = 7.438 km
280 m mixed gauges
280 m mixed gauges
280 m mixed gauges
LT
1829 m
1219 m
1829 m
1518 m
738 m
305 m
NT
24 AWG
22 AWG
19 AWG

(e) Loops #21 - #25



Loop #26:
L = 7.772 km
280 m mixed gauges
280 m mixed gauges
280 m mixed gauges
LT
1829 m
1202 m
626 m
1829 m
1042 m
1244 m
NT
24 AWG
22 AWG
19 AWG
Loop #27:
L = 8.077 km
280 m mixed gauges
280 m mixed gauges
280 m mixed gauges
1219 m
1527 m
1521 m
2356 m
235 m
1219 m
NT
24 AWG
22 AWG
19 AWG
Loop #28:
L = 8.382 km
280 m mixed gauges
280 m mixed gauges
280 m mixed gauges
LT
2515 m
533 m
1372 m
914 m
1372 m
1676 m
NT
24 AWG
22 AWG
19 AWG
Loop #29:
L = 8.687 km
280 m mixed gauges
280 m mixed gauges
280 m mixed gauges
LT
1219 m
953 m
1486 m
1981 m
1911 m
NT
24 AWG
22 AWG
19 AWG
Loop #30:
L = 8.992 km
280 m mixed gauges
280 m mixed gauges
280 m mixed gauges
LT
1798 m
1250 m
1219 m
1067 m
1500 m
2158 m
NT
24 AWG
22 AWG
19 AWG

(f) Loops #26 - #30


Loop #31:
L = 9.296 km
280 m mixed gauges
280 m mixed gauges
280 m mixed gauges
LT
1394 m
1654 m
2438 m
1486 m
190 m
2134 m
NT
24 AWG
22 AWG
19 AWG
Loop #32:
L = 9.601 km
280 m mixed gauges
280 m mixed gauges
280 m mixed gauges
LT
960 m
564 m
3048 m
2629 m
419 m
1981 m
NT
24 AWG
22 AWG
19 AWG
Loop #33:
L = 9.907 km
280 m mixed gauges
280 m mixed gauges
280 m mixed gauges
LT
495 m
1943 m
2896 m
1524 m
572 m
2477 m
NT
24 AWG
22 AWG
19 AWG
Loop #34:
L = 10.211 km
280 m mixed gauges
280 m mixed gauges
280 m mixed gauges
LT
511 m
1928 m
2438 m
1676 m
1105 m
2553 m
NT
24 AWG
22 AWG
19 AWG
Loop #35:
L = 10.518 km
280 m mixed gauges
280 m mixed gauges
280 m mixed gauges
LT
526 m
3132 m
1600 m
1600 m
1030 m
2630 m
NT
24 AWG
22 AWG
19 AWG

(g) Loops #31 - #35

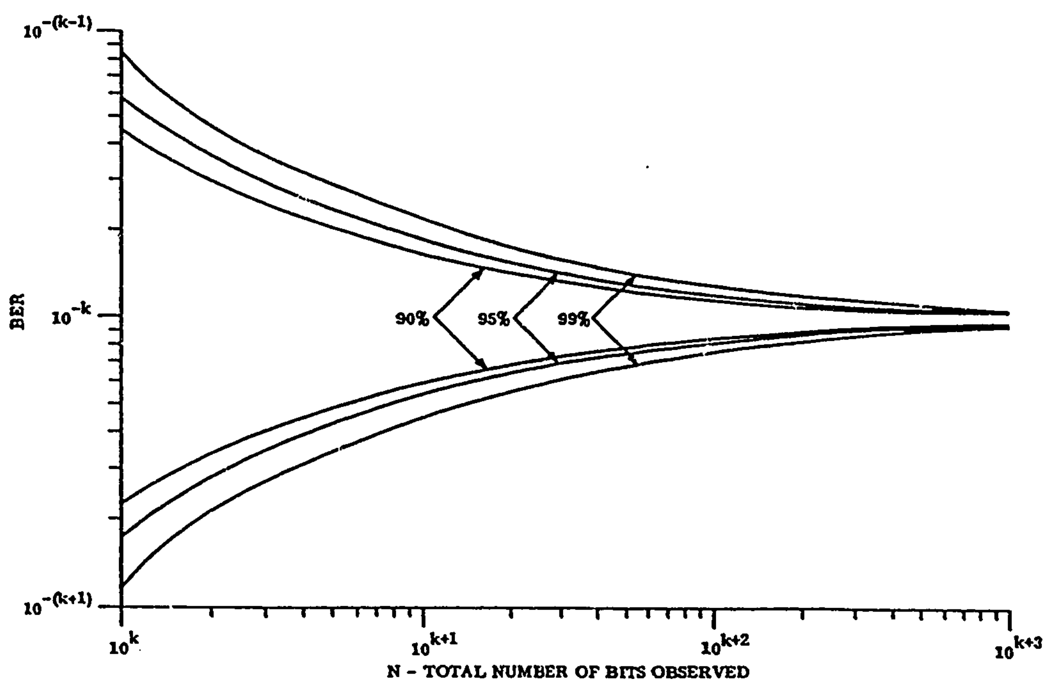


Fig. 5.2 Confidence bands on BER of Monte Carlo technique

In the simulation study, a maximum of 10^6 symbols are sent, and therefore the observed BER is reliable if it is greater than 10^{-5}.

The Monte Carlo method is inefficient because a large number of noise samples generated will not cause errors, which is a waste of computational efforts in BER estimation. The importance sampling method [27,28], which reduces the variance of p through biasing of the noise statistics, is however not feasible due to the non-Gaussian character of the disturbing echo and the non-linearity of the DFE. Because of the lack of a better BER estimation method, the Monte Carlo method was used.

5.3 Simulation results

5.3.1 Transceiver with DFE equalizer

Bit error rate simulation study is performed on each of the 35 chosen loops using the FORTRAN program dfeber.for in Appendix D. The flow chart of the simulation program is shown in Fig. 5.3. For each loop, the relationship between the transmission rate and the bit error rate is exploited. Instead of making the transmit symbol rate a continuous function, it ranges from 16 kbaud to 768 kbaud in 16 kbaud increments. This is justified because the envisaged RA-DSL transceiver uses quantised bit rates for practical reasons. Also, the maximum transmit symbol rate is 768 kbaud (1.536 kbit/s = 24 B channels) because we intended to support a maximum transmission rate up to the T1 rate. Some of the assumptions that are used in the simulation study are as follows:

(1) Signal level of 13.5 dBm is assumed at the line and network interface.

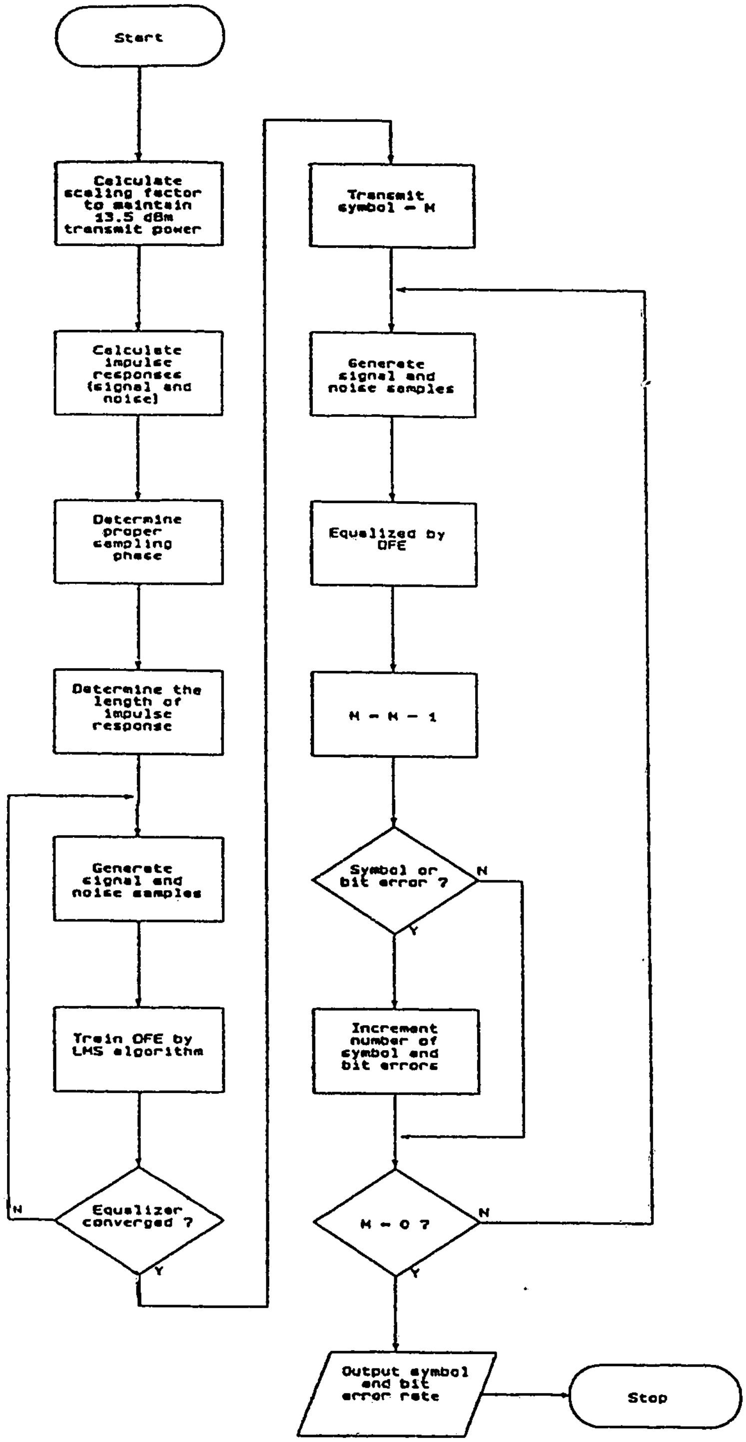


Fig. 5.3 Flow chart for the bit error rate estimation program

(2) White Gaussian noise with $N_0/2 = 2.5 \times 10^{-14}$ W/Hz is assumed at the output of the hybrid [2].

(3) The near-end crosstalk coupling constant K_{NEXT} has a value of 88.18×10^{-15} $Hz^{-1.5}$, which is equivalent to -57dB at 80 kHz [1].

(4) The echo canceller is assumed to provide 40 dB or 65 dB echo cancellation in two independent sets of simulations.

(5) The load impedance is 135 ohms resistive and the balance network shown in Fig. 2.3 is used.

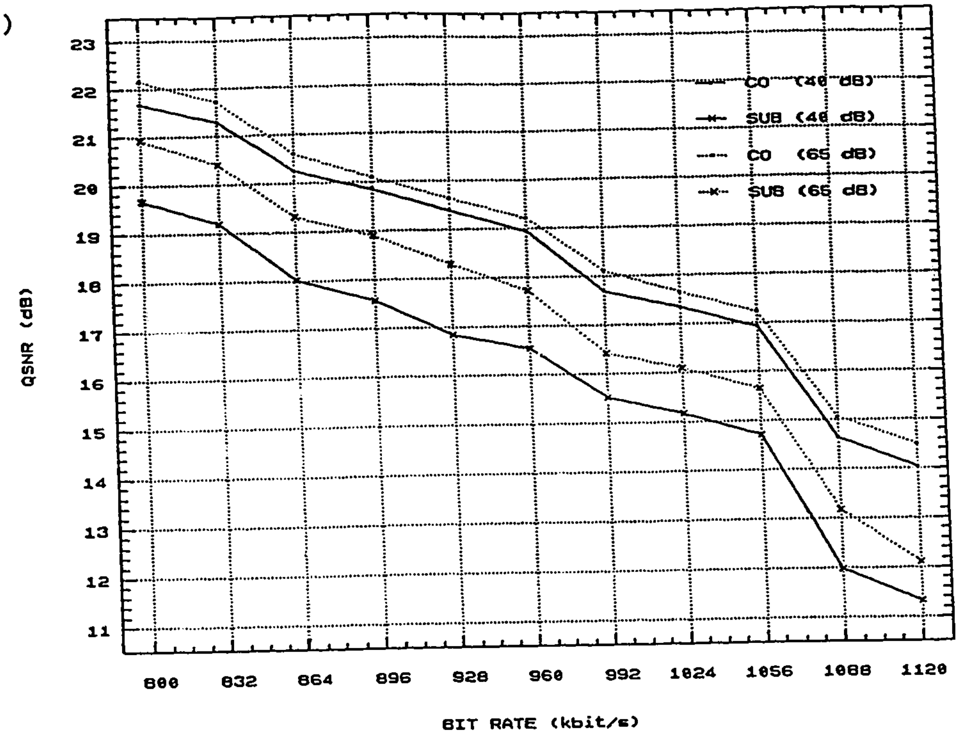
(6) The excess bandwidth factor α is set to 1.0. This may not be an optimized value to reduce the bit error rate, but the amount of ISI can be reduced.

(7) The DFE with no forward filter has 20 taps. This value is chosen because the postcursors are negligible after 20 symbol periods in most cases.

(8) The step size parameter μ is set to 7.5×10^{-4} (trial and error).

The relationships between bit error rate, quantiser signal to noise ratio, and transmission rate for loops number 8, 12, and 18 are shown in Figs. 5.4 to 5.6, while the results for loops number 26 and 33 are shown in Table 5.2. The results of these two loops are shown in table because there are not enough data to plot the BER curve. The selection of these loops is arbitrary, but they represent loops of gradually increasing loop length. The notation 'CO' in the graph means that the receiver is at the central office, while 'SUB' denotes the receiver at the subscriber. Two cases with 40 dB and 65 dB echo

(a)



(b)

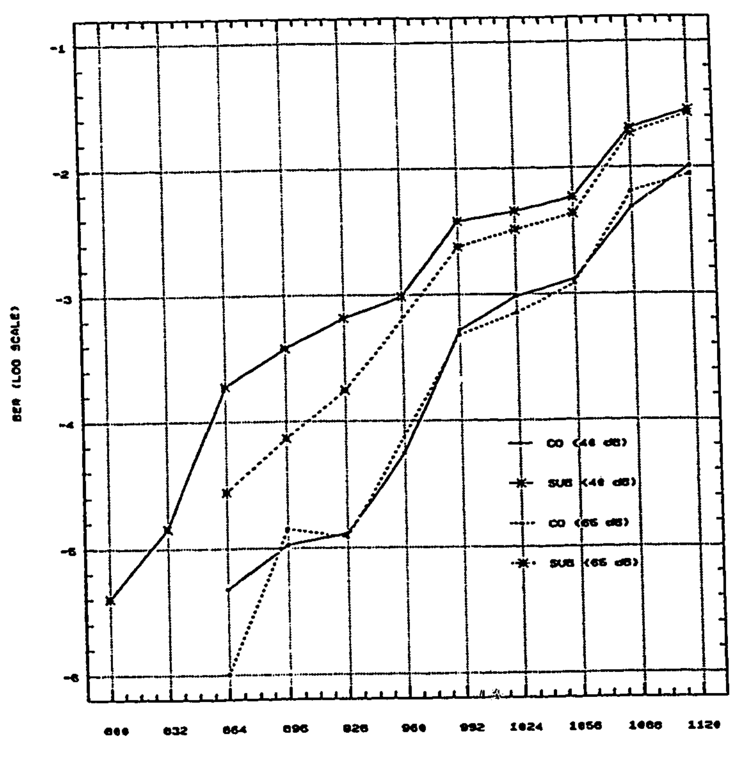
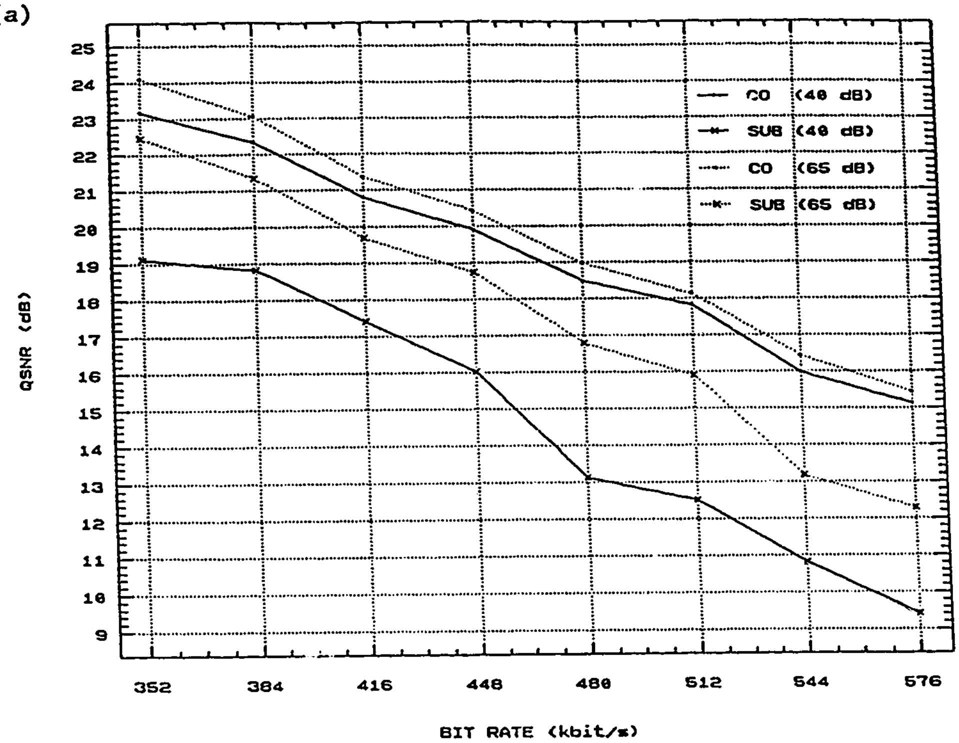


Fig. 5.4 (a) QSNR versus transmission rate (b) BER versus transmission rate of loop #8 using DFE equalizer

(a)



(b)

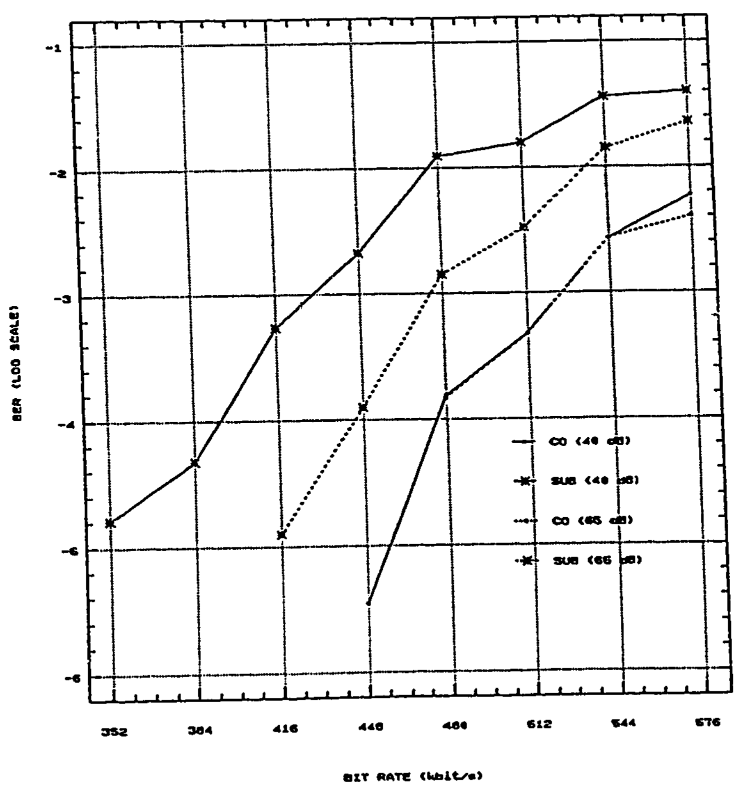


Fig. 5.5 (a) QSNR versus transmission rate (b) BER versus transmission rate of loop #12 using DFE equalizer

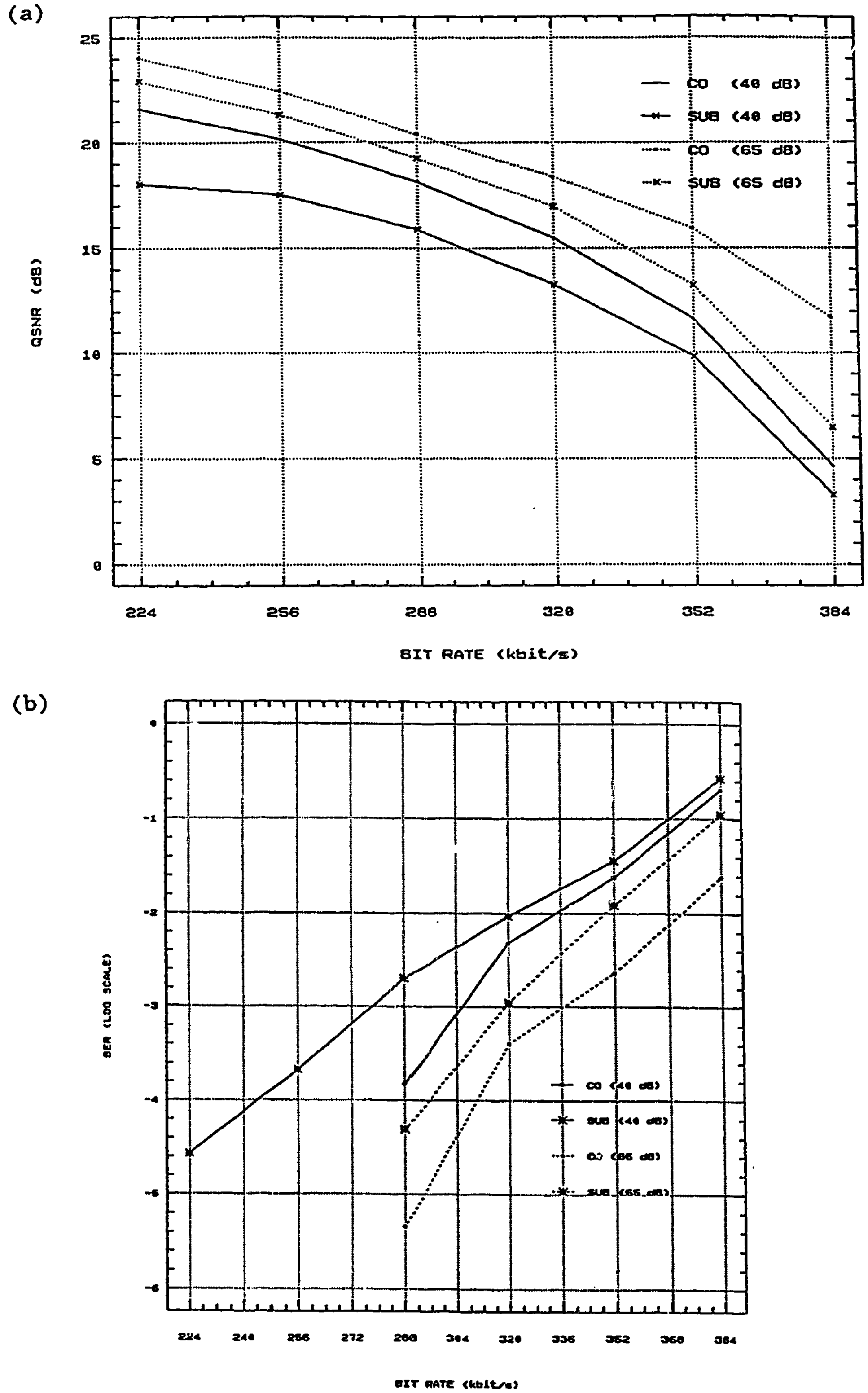


Fig. 5.6 (a) QSNR versus transmission rate (b) BER versus transmission rate of loop #18 using DFE equalizer

(a)

Bitrate (kbit/s)	40 dB Echo Cancellation QSNR(dB) CO	SUB	BER(log) CO	SUB	65 dB Echo Cancellation QSNR(dB) CO	SUB	BER(log) CO	SUB
96	36.76	17.33	?	?	37.66	32.17	?	?
128	29.16	15.37	?	-3.137	30.66	28.36	?	?
160	25.97	12.92	?	*	27.27	25.22	?	?
192	22.56	6.61	?	*	23.73	21.86	?	?
224	20.43	5.14	?	*	21.33	19.59	?	-4.559
256	17.84	3.33	-3.128	*	18.43	15.69	-3.512	-2.502
288	12.16	2.50	-1.975	*	13.02	3.64	-2.022	*

(b)

Bitrate (kbit/s)	40 dB Echo Cancellation QSNR(dB) CO	SUB	BER(log) CO	SUB	65 dB Echo Cancellation QSNR(dB) CO	SUB	BER(log) CO	SUB
64	30.51	18.11	?	?	38.36	35.71	?	?
96	28.48	15.97	?	-4.067	33.01	30.71	?	?
128	26.00	14.39	?	-2.372	28.71	26.73	?	?
160	23.00	9.96	?	*	24.60	22.84	?	?
192	20.87	6.78	?	*	21.83	20.21	?	?
224	18.24	4.37	-3.262	*	18.86	16.83	-3.983	-2.805
256	13.11	3.12	-1.855	*	14.26	10.18	-1.855	*

Remarks : * BER meaningless (too large)
? No error for 10^6 transmitted symbols (too small)

Table 5.2 QSNR and BER versus transmission rate for (a) loop #26 (b) loop #33 using a DFE equalizer cancellation are considered. The results show that the BER is a monotonically increasing function of transmission rate, while the QSNR (as defined in chapter 4) is a monotonically decreasing function of transmission rate.

For very short loops such as loops 1-4 ($\leq$ 1 km), the transmission rate can go up to 1.536 Mbit/s with BER < 10^{-5}. Therefore, the figures for these loops are not included because they are not too informative. Fig. 5.4a shows the QSNR versus transmission rate for loop #8 (2.28 km). The performance is worst at the receiver on the subscriber side, and the QSNR at 800 kbit/s is 2 dB smaller for 40 dB echo cancellation and 1.3 dB smaller for 65 dB echo cancellation. Fig. 5.4b shows the plot of BER versus transmission rate. The result shows that the echo is not a dominant noise component on the central office side because the performance is similar regardless of the echo cancellation level. On the other hand, the echo on the subscriber side is significant. A much better BER performance is obtained when the echo level is reduced. The effect is more apparent at lower transmission rates when the NEXT power decreases, and the echo power becomes dominant.

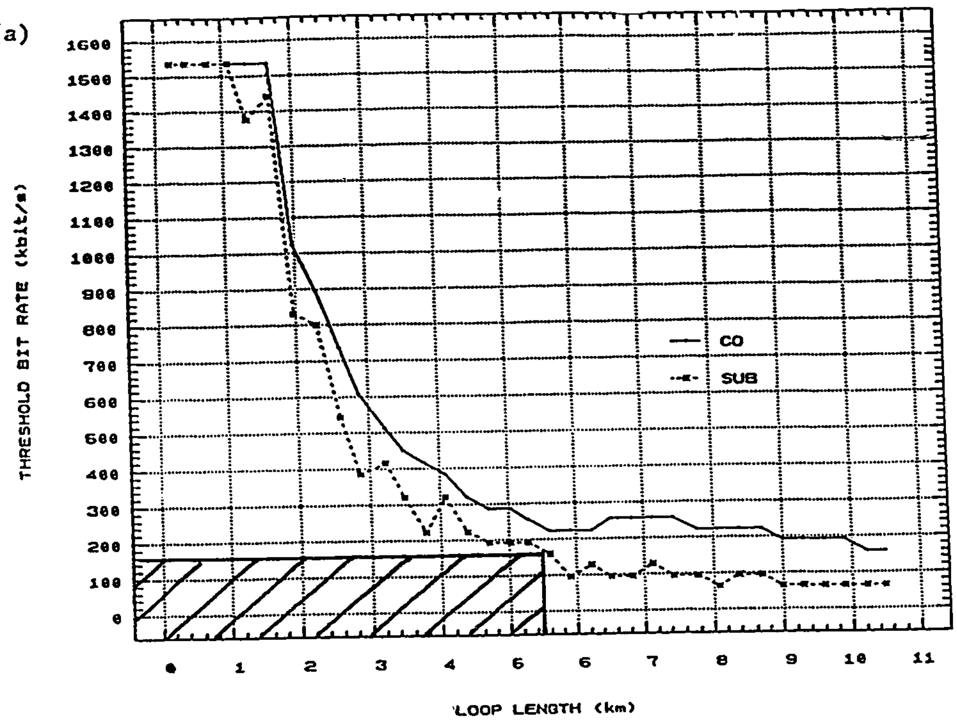
Similar results are obtained for loop #12 (Fig. 5.5), which is a moderately long loop of 3.49 km. The results of loop #18 (Fig. 5.6), which is a long loop of 5.28 km, show different characteristics. The performances of the QSNR and BER versus transmission rate are enhanced substantially by increasing the echo cancellation level. This is because at a lower transmission rate on longer loops, the NEXT becomes less important and the echo becomes dominant. The reduction of echo will enhance the QSNR or the BER performance to a greater extent. Table 5.2 shows the results of loops #26 and #33.

The QSNR versus transmission rate can be divided in two linear region, a high and a low QSNR region, and there is a sharp change of QSNR at the transition. For instance, the QSNR on the subscriber side of loop #26 dropped from 12.9 to 6.6 dB as the transmission rate increased from 160 kbit/s to 192 kbit/s. This may be explained by the error propagation in the DFE. When the transmission rate increases, error propagation becomes important and deteriorates the QSNR more rapidly. Therefore, the slope of the low QSNR region is much steeper than the high QSNR region due to error propagation. The echo is dominant on the subscriber side, and the QSNR at 96 kbit/s can be better by 15 dB with a better echo canceller. The gain is not as substantial on the central office side, but improvement in performance is still noticeable. The BER performance shows similar characteristics.

The capacity of the subscriber loops can be estimated from these BER versus bit rate curves. If we assumed the threshold bit error rate to be less than or equal to 10^{-5}, then the maximum transmission rate that is feasible on each of the chosen loops can be determined. The results which demonstrate the relationships between loop length and its capacity are shown in Fig. 5.7. The cross hatched area in the figure shows the capacity that is available for the ISDN basic access.

Fig. 5.7a shows the length-capacity relationship with 40 dB echo cancellation. It is possible to achieve bit rate of 1.536 Mbit/s over loops up to 1 km long and with bit rate reduced to 64 kbit/s, a reach of 10.7 km is achievable. Unfortunately, the performance is worse on the subscriber side with a difference of around 100-200 kbit/s. This will greatly reduce the potential gain of the RA-DSL transmission method because the transmission rates are kept the same for both directions.

(a)



(b)

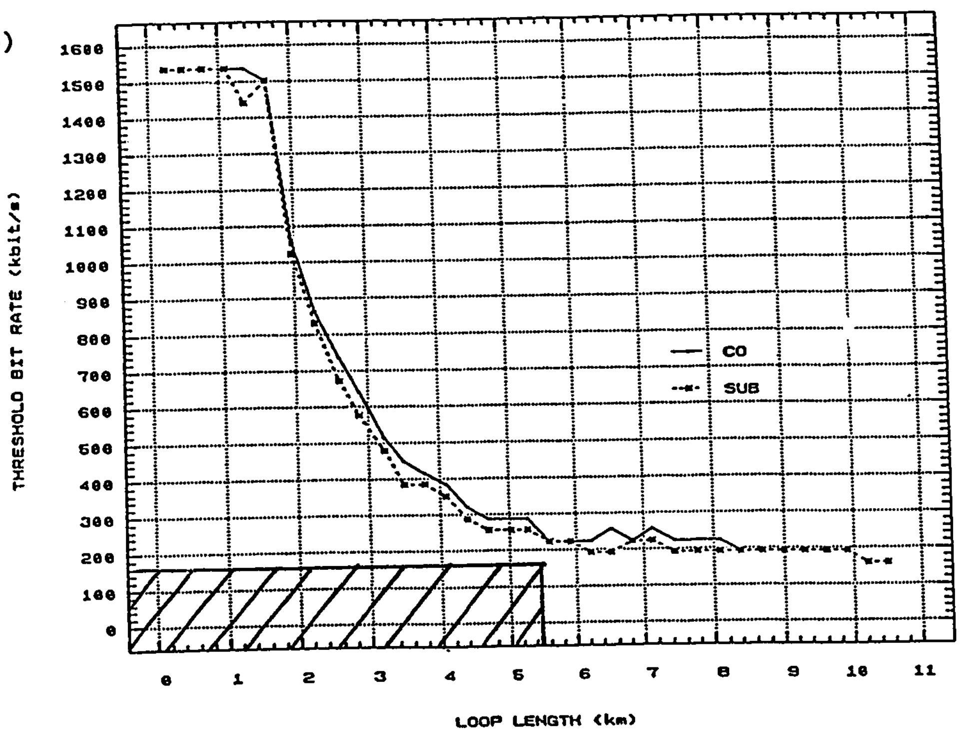


Fig. 5.7 Length-capacity relationship for sample loop models using DFE equalizer (a) 40 dB and (b) 65 dB echo cancellation

The difference may be explained by the fact that bridged taps are usually located closer to the subscriber side, and hence the impedance matching is worse and echo cancellation is less effective. The greater echo power reduces the performance of the receiver on the subscriber side.

Fig. 5.7b showing the length-capacity relationship with 65 dB echo cancellation justifies this explanation. If we assume the hybrid can provide about 10 dB echo cancellation, a state of the art echo canceller can provide an additional 65 dB [9]. This reduces the digital subscriber loop to a crosstalk limited environment. The results show that the performance is then nearly the same for both directions of transmission, with a slightly better overall performance than the 40 dB echo cancellation case. The gain in performance is more significant for longer loops in which a bit rate of 160 kbit/s is now achievable at a reach of 10.5 km.

Comparing to the ISDN 2B+D basic access, the RA-DSL transmission method can provide higher transmission rate on the same loop plant. We can recognize the extra transmission capacity as the gain of the RA-DSL transmission method. A rough estimate of the potential gain in service capacity of the RA-DSL transmission method can be calculated. Table 5.3 shows the relationship between the loop population and loop length, in which the data are extracted from Fig. 5.8 [7]. The available capacity of the loop plant can be defined as:

$$\text{Capacity} = \sum_{i=1}^{N} \text{Bitrate}(i) * \text{Population}(i) \qquad (5.3)$$

Loop Length (km)	Loop Number	Percentage of loop population
0 - 0.305	1	3.5
0.305 - 0.610	2	3.0
0.610 - 0.915	3	3.3
0.915 - 1.220	4	5.2
1.220 - 1.425	5	8.0
1.525 - 1.830	6	7.0
1.830 - 2.135	7	6.0
2.135 - 2.440	8	6.0
2.440 - 2.745	9	8.0
2.745 - 3.050	10	7.0
3.050 - 3.355	11	7.0
3.355 - 3.660	12	6.0
3.660 - 3.965	13	4.0
3.965 - 4.270	14	5.0
4.270 - 4.575	15	3.0
4.575 - 4.880	16	3.0
4.880 - 5.185	17	3.0
5.185 - 5.490	18	2.0
5.490 - 5.795	19	1.0
5.795 - 6.100	20	0.8
6.100 - 6.405	21	0.8
6.405 - 6.710	22	0.8
6.710 - 7.015	23	0.8
7.015 - 7.320	24	0.8
7.320 - 7.625	25	0.3
7.625 - 7.930	26	0.3
7.930 - 8.235	27	0.3
8.235 - 8.540	28	0.3
8.540 - 8.845	29	0.3
8.845 - 9.150	30	0.3
9.150 - 9.455	31	0.3
9.455 - 9.760	32	0.3
9.760 - 10.065	33	0.2
10.065 - 10.370	34	0.2
10.370 - 10.675	35	0.2

Table 5.3 Loop population versus loop length

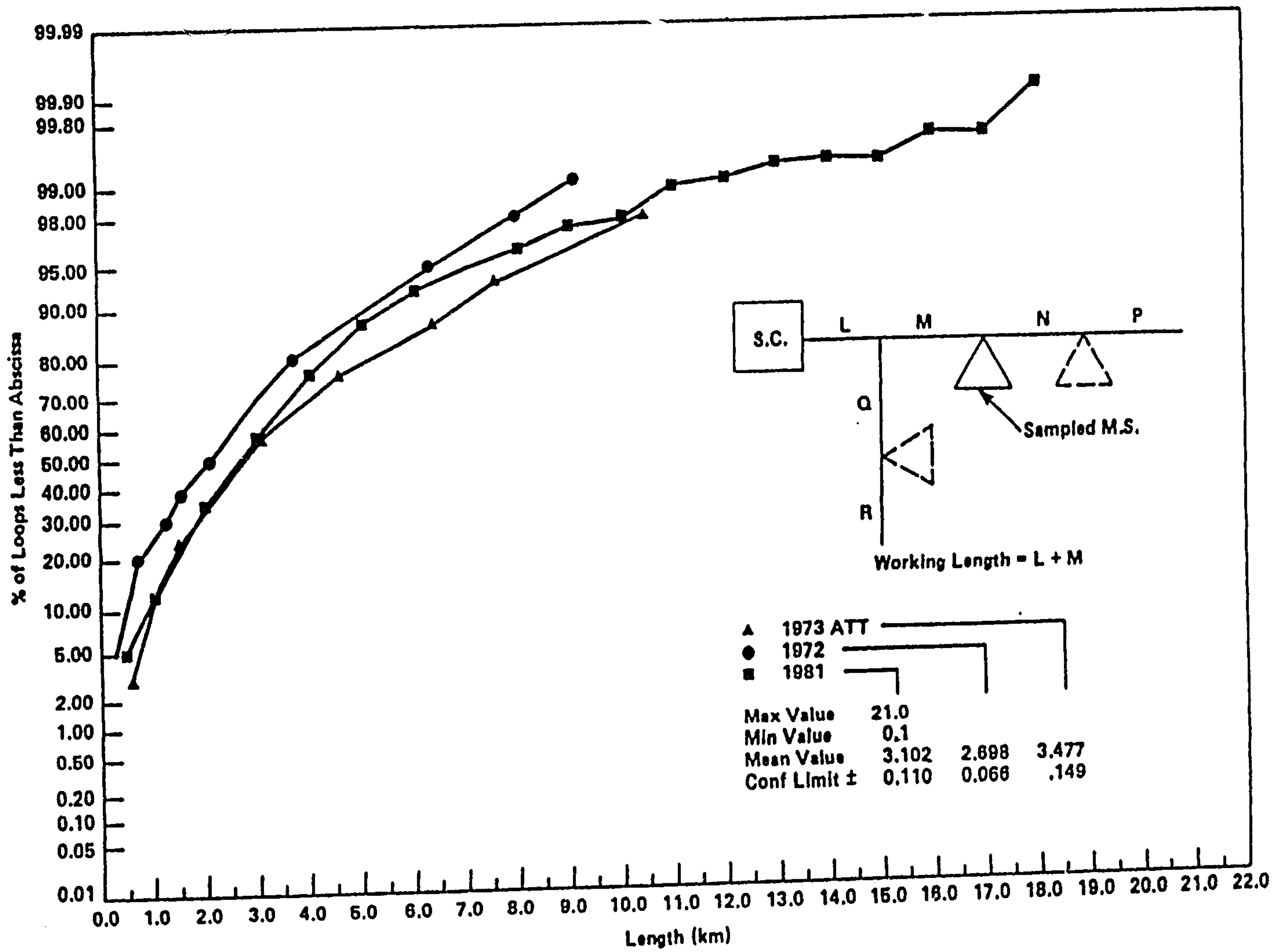


Fig. 5.8 Working length to main station

Index i denotes one of N loop length ranges (305 m length increments), bitrate(i) is the maximum transmission rate with BER $\leq 10^{-5}$ for an average loop in the range i, population(i) is the percentage of loops that are within the range i. For the ISDN basic access N=18, which includes all loops up to 5.5 km long. Consequently, the potential service capacity of the ISDN basic rate access is given by:

$$\text{Capacity}_1 = \sum_{i=1}^{18} (160 \text{ kbit/s})*\text{Population}(i) \qquad (5.4)$$

For the RA-DSL transmission method N=35, which corresponds to the coverage of all loops within 10.7 km from a central office. Consequently, the potential service capacity of the RA-DSL transmission method is given by:

$$\text{Capacity} = \sum_{i=1}^{18} \text{Bitrate}(i)*\text{Population}(i) + \sum_{i=19}^{35} \text{Bitrate}(i)*\text{Population}(i)$$

$$= \text{Capacity}_2 + \text{Capacity}_3 \qquad (5.5)$$

The capacity gain due to the increase in transmission rate can be defined as $\text{Capacity}_2/\text{Capacity}_1$, while the gain due to the extension in coverage area is defined as $\text{Capacity}_3/\text{Capacity}_1$. The following table shows the results of the capacity gain calculation using the data of Table 5.3, and previously determined length-capacity relations.

Echo cancellation	Rate gain	Extension gain	Total gain
40 dB	4.90	0.06	4.96
65 dB	5.38	0.11	5.49

The overall capacity gain is greater with better echo cancellation, and the rate gain is more important than the extension gain in both cases. The reason is that 90% of the loop population is within 5.5 km, while only 8.5% of the loop population is within the area between 5.5 km and 10.7 km. Because of the smaller population, the extension gain is less important. On the other hand, a rate gain of 5 is quite substantial. It will provide a weighted average of 160 kbit/s * 5 = 800 kbit/s to a subscriber within 5.5 km. This figure is somewhat optimistic because ISDN basic access requires BER $\leq 10^{-7}$ and not 10^{-5} assumed here. If we linearly extrapolate the BER versus transmission rate curves (Fig. 5.4b - 5.6b), we can expect that BER $\leq 10^{-7}$ is achievable if the transmission rate is lowered by 96 kbit/s. For longer loops such as loops #26 and #33, the transmission rate can decrease by less than 96 kbit/s to achieve BER $\leq 10^{-7}$. Therefore, the length-capacity relationship with threshold BER $\leq 10^{-7}$ can be made artificially by lowering the curves of Fig. 5.7 by 96 kbit/s. The new gain factor can now be calculated to be 4.35, which is comparable to the value of 5 (a 13% drop).

5.3.2 Transceiver with fractionally spaced DFE equalizer

In order to test the possibility of further gain using a better equalizer structure, short simulation runs have been performed on some individual loop models. Transceivers with T-spaced and T/2 fractionally spaced forward filters in front of the DFE were used. The results show that DFE with a T-spaced forward filter gives no apparent improvement, while DFE with a T/2 fractionally spaced forward filter shows significant improvement. This result should be expected because the fast rising edge of the signal impulse response has insignificant precursors, and the method of minimizing the first precursor through sampling phase (explained in section 4.3.2) will further reduce the precursor ISI to a negligible level. As a result, removal of precursor ISI through a forward equalizer will not enhance performance, while a T/2 fractionally spaced equalizer acting as a matched filter can improve it.

A transceiver with an ISI canceller was also studied, but its performance was similar to the fractionally spaced DFE. This can again be explained by the insignificant amount of precursor ISI in the received signal. Based on these observations, the T/2 fractionally spaced DFE equalizer was chosen for another set of simulations, since it offers significant improvement, while its structure is not overly complex.

The T/2 fractionally spaced equalizer had 20 front and back taps, while the DFE had 20 taps. The step size parameter μ was 7.5×10^{-4} for both equalizers. The relationships of bit error rate and quantiser signal to noise ratio versus transmission rate for selected loops number 8, 12, and 18 are shown in Figs. 5.9 to 5.11, while the results for loop number 26 and 33 are shown in Table 5.4. Comparing with the previous results, these loops show better performance, and the transmission rate can increase by more than 300 kbit/s for short loops and more than 32 kbit/s for long loops. Also, we observe that the BER versus transmission rate curve can be divided into two linear regions, the high bit error rate and the low bit error rate region. This is different from the performance of the plain DFE in which the BER increases gradually with the transmission rate. At the high bit error rate region, in which the BER > 10^{-3}, the slope is quite flat. This may be due to the error

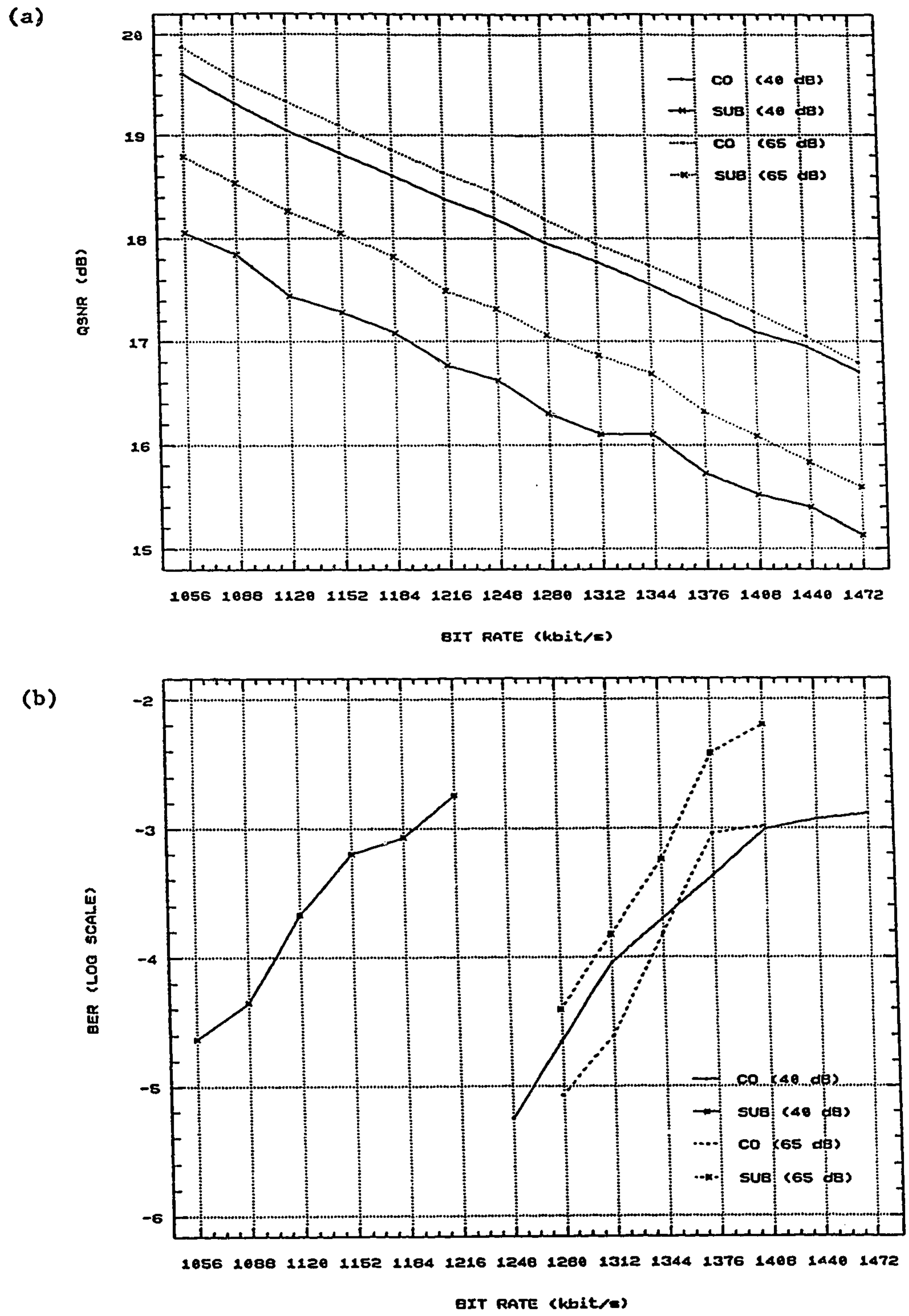
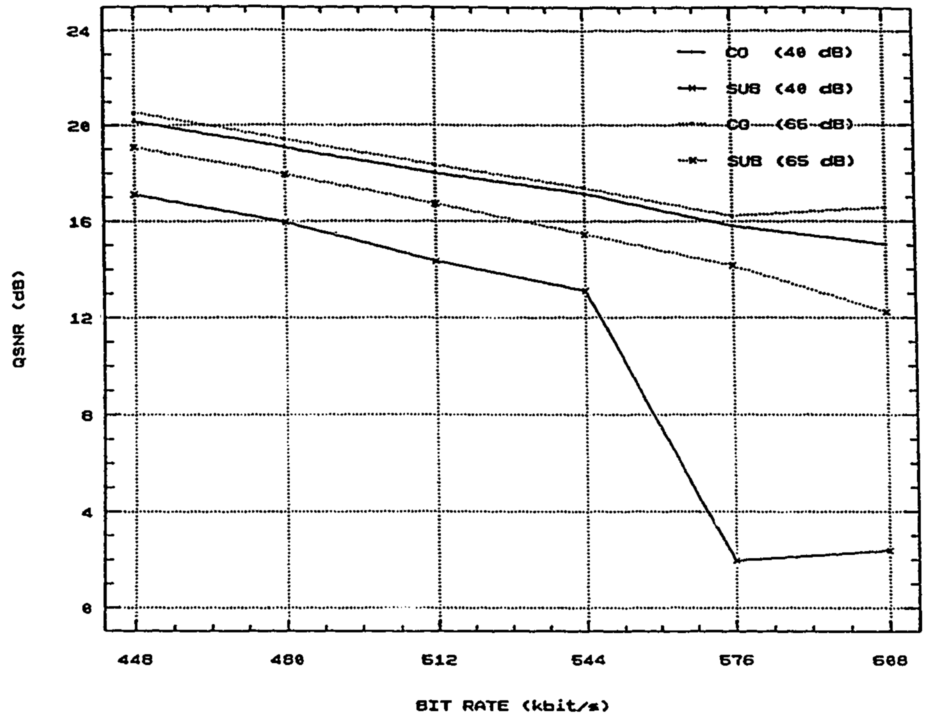


Fig. 5.9 (a) QSNR versus transmission rate (b) BER versus transmission rate of loop #8 using FSDFE equalizer

(a)



(b)

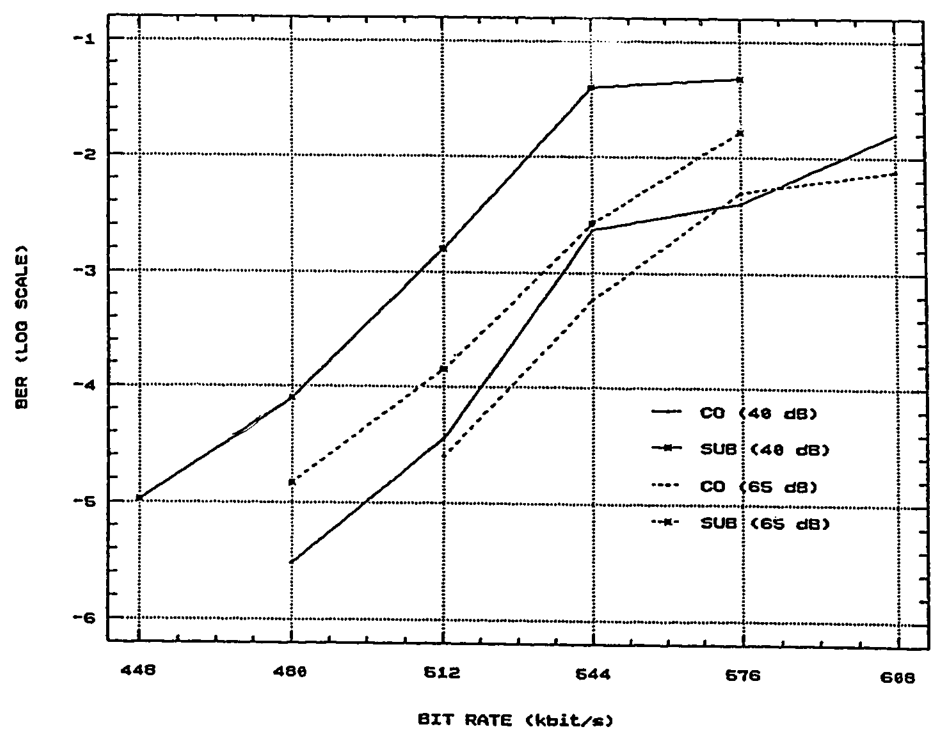


Fig. 5.10 (a) QSNR versus transmission rate (b) BER versus transmission rate of loop #12 using FSDFE equalizer

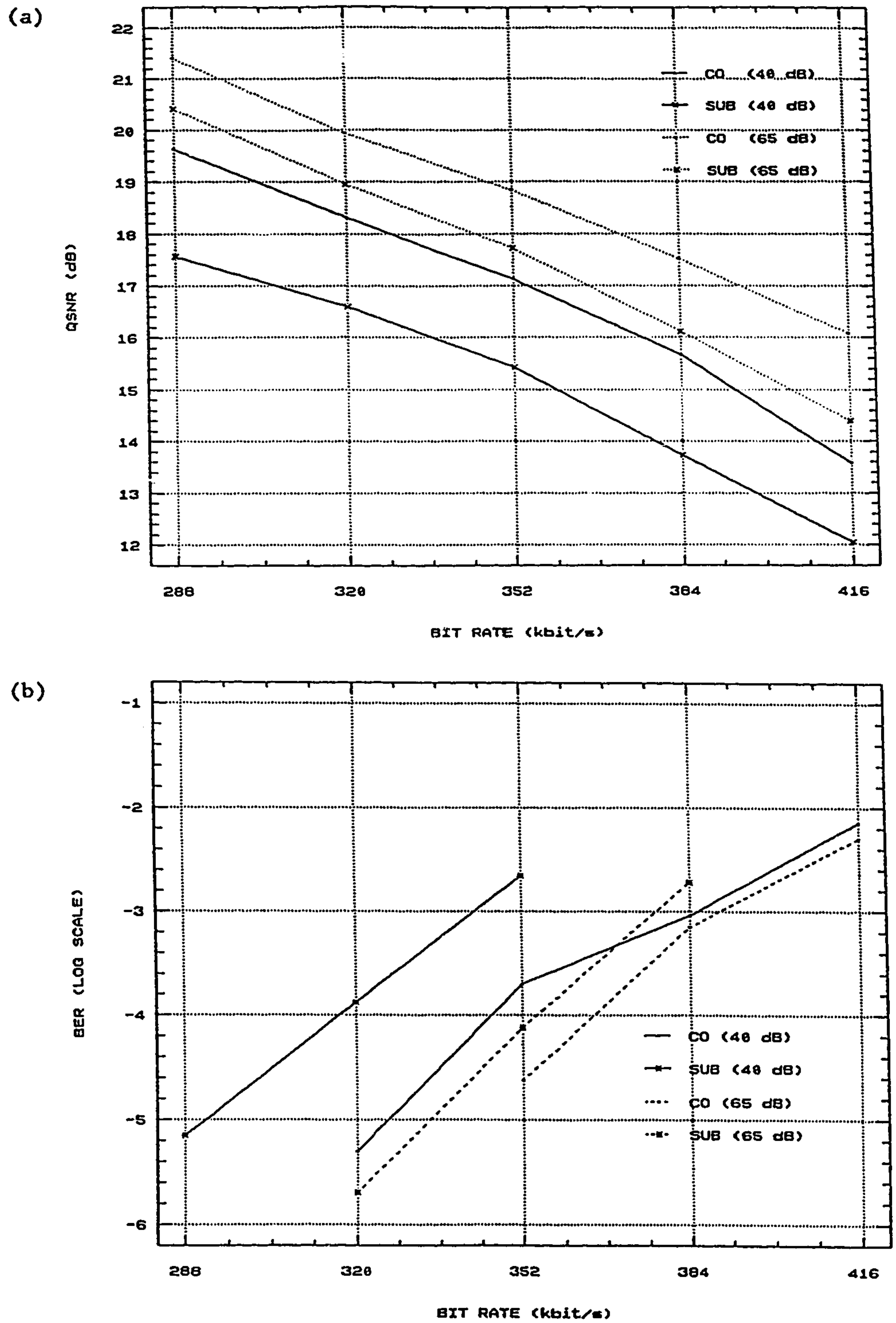


Fig. 5.11 (a) QSNR versus transmission rate (b) BER versus transmission rate of loop #18 using FSDFE equalizer

(a)

Bitrate (kbit/s)	40 dB Echo Cancellation QSNR(dB) CO	SUB	BER(log) CO	SUB	65 dB Echo Cancellation QSNR(dB) CO	SUB	BER(log) CO	SUB
160	26.53	16.32	?	?	27.64	25.72	?	?
192	23.99	14.75	?	-3.910	25.10	23.27	?	?
224	21.99	13.99	?	-2.360	22.98	21.22	?	?
256	20.32	12.49	?	*	21.05	19.39	?	-5.823
288	18.82	2.61	-4.661	*	19.39	17.72	-5.076	-4.042
320	17.15	4.14	-3.415	*	17.71	15.62	-3.722	*
352	15.35	1.23	?	*	15.56	13.04	*	*

(b)

Bitrate (kbit/s)	40 dB Echo Cancellation QSNR(dB) CO	SUB	BER(log) CO	SUB	65 dB Echo Cancellation QSNR(dB) CO	SUB	BER(log) CO	SUB
160	24.40	15.20	?	-4.283	26.01	24.17	?	?
192	22.50	13.24	?	-2.853	23.62	21.79	?	?
224	20.67	11.94	?	-1.727	21.46	19.73	?	?
256	18.95	3.87	-5.995	*	19.54	17.91	-5.995	-4.716
288	17.35	1.97	-3.163	*	17.71	15.90	-3.861	-2.712
320	14.92	1.60	*	*	15.61	12.46	-2.188	*

Remarks : * BER meaningless (too large)
? No error for 10^6 transmitted symbols (too small)

Table 5.4 QSNR and BER versus transmission rate for (a) loop #26 (b) loop #33 using a fractionally spaced DFE equalizer propagation in the DFE that made the BER deteriorate. At the extreme, error propagation can cause the adaptive equalizer to diverge. This happened occasionally, and then the value of μ had to be decreased to 5 x 10^{-4}. This problem does not occur for the plain DFE, and we suspect the value of μ for the fractionally spaced equalizer was not chosen properly. The value of μ for the FSE has not been changed because we want to have a fair comparison between the two transceiver structures (the value of the MSE is proportional to μ). The slope of the BER versus transmission rate curve is very steep at the low bit error region, and BER can decrease by more than one order of magnitude if the transmission rate decreases by 32 kbit/s. This is more noticeable for longer loops. The differences in the performance characteristics for the plain and fractionally spaced DFE may be explained as follows: At high BER region, the problem of error propagation causes the fractionally spaced DFE performs no better than the plain DFE. Therefore, the BER versus transmission rate curves are quite flat for both equalizer structures. However, at a low BER region, the problem of error propagation no longer exists. The fractionally spaced DFE now performs significantly better than the plain DFE and results in a steeper BER versus transmission curves.

The length - capacity relationship for the 35 loops can again be calculated. Figs. 5.12a and 5.12b show the case with 40 dB and 65 dB echo cancellation respectively. These two figures are very similar to Fig. 5.7a and 5.7b except that the transmission rate is higher at the same loop length. Any remarks made on the previous two figures will be applicable to these two figures. It is now possible to achieve bit rate of 1.536 Mbit/s up to 1.5 km and with bit rate reduced to 128 kbit/s

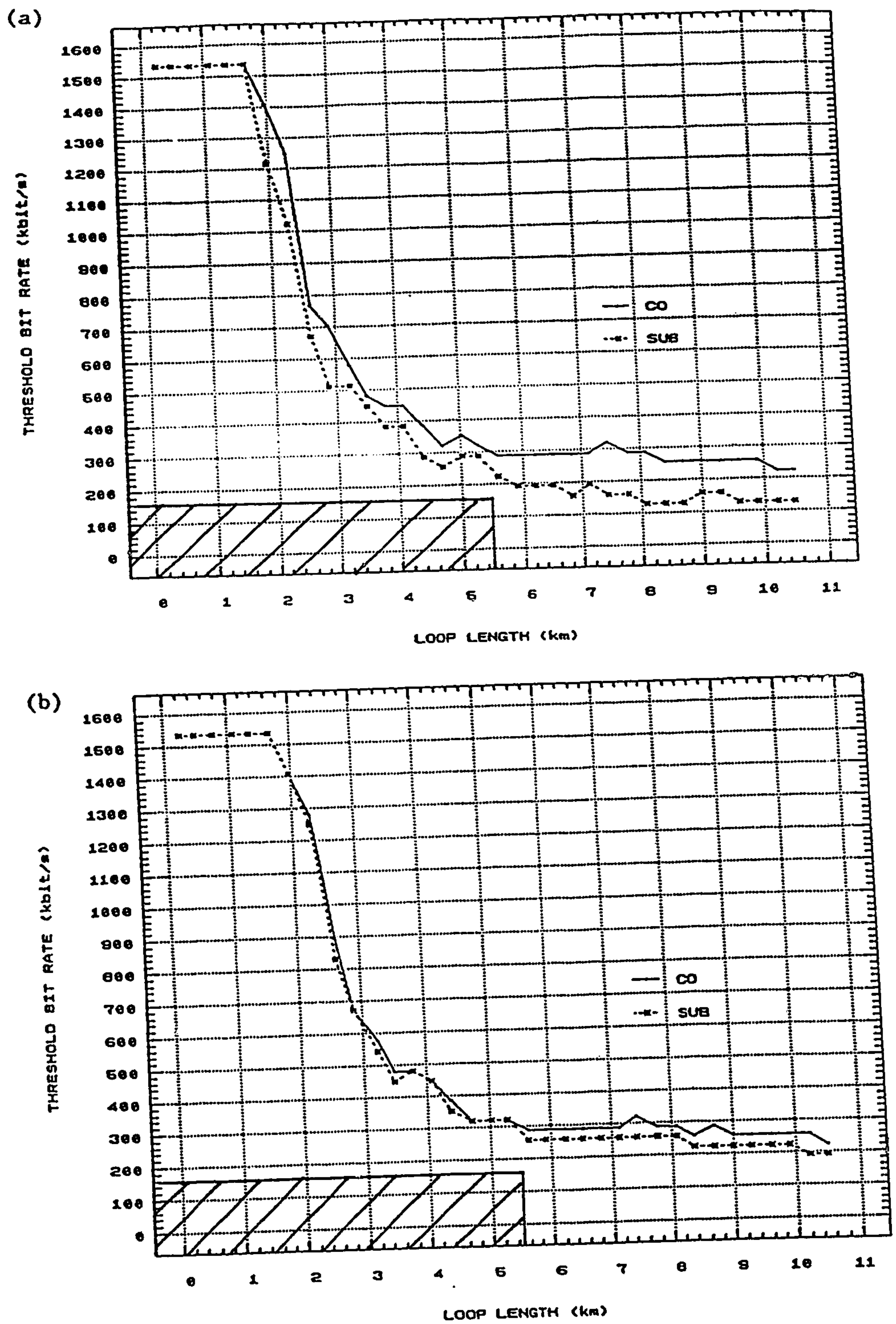


Fig. 5.12 Length-capacity relationship for sample loop models using FSDFE equalizer (a) 40 dB and (b) 65 dB echo cancellation

(192 kbit/s with 65 dB echo cancellation), a reach of 10.7 km is achievable. It will be interesting to calculate the capacity gain obtained by employing a more complex equalizer structure. Using the previous loop population data, the following results are obtained.

Echo cancellation	Rate gain	Extension gain	Total gain
40 dB	5.64	0.10	5.74
65 dB	6.10	0.13	6.23

The extension gain is again insignificant, and the rate gain is around 5.9. This can provide an average of 160 kbit/s * 5.9 = 944 kbit/s to the subscriber within 5.5 km, which is a 18% increase comparing to the previous result. If we linearly extrapolate the BER versus transmission rate curves (Figs. 5.9b - 5.11b), we can expect that BER $\leq 10^{-7}$ is achievable if the transmission rate is lowered by 64 kbit/s. The decrease in transmission rate is smaller than the plain DFE case because the BER versus transmission rate curves have a steeper slope at the low BER region. For longer loops such as loops #26 and #33, the transmission rate can decrease by less than 64 kbit/s to achieve BER $\leq 10^{-7}$. As a result, the length-capacity relationship with threshold BER $\leq 10^{-7}$ can be made artifically by lowering the curves of Fig. 5.12 by 64 kbit/s. The new gain factor can now be calculated to be 5.46, which is comparable to the value of 5.9 (a 7.5% drop). The increase in capacity by using a more complex equalizer structure is quite significant, but the cost and complexity of the increased hardware may be a concern.

The last case we will consider is to calculate the potential gain in service capacity using an uniform subscriber density. This will probably be the situation when the first ISDN office is introduced in the metropolitan area. The calculation is similar to the previous one, and the available capacity of the loop plant is defined as:

$$\text{Capacity} = \sum_{i=1}^{N} S*\pi(i^2-(i-1)^2)*\text{Bitrate}(i) \qquad (5.6)$$

Index i denotes one of N loop length ranges (305 m length increments), bitrate(i) is the maximum transmission rate with BER $\leq 10^{-5}$ for an average loop in the range i, S is the subscriber density which is uniform for all values of i. For the ISDN basic access N=18, which includes all loops up to 5.5 km long. Consequently, the potential service capacity of the ISDN basic rate access is given by:

$$\text{Capacity}_1 = (160 \text{ kbit/s})*S*\pi N^2; \qquad N=18 \qquad (5.7)$$

while the potential service capacity of the RA-DSL transmission method N=35 is given by:

$$\text{Capacity} = \sum_{i=1}^{18} \text{Bitrate}(i)*S*\pi(i^2-(i-1)^2) + \sum_{i=19}^{35} \text{Bitrate}(i)*S*\pi(i^2-(i-1)^2)$$

$$= \text{Capacity}_2 + \text{Capacity}_3 \qquad (5.8)$$

The following table shows the results of the capacity gain calculation for uniform subscriber density.

	Echo cancellation	Rate gain	Extension gain	Total gain
DFE	40 dB	2.75	1.44	4.19
	65 dB	3.23	3.24	6.47
FSDFE	40 dB	3.43	2.57	6.00
	65 dB	3.85	3.91	7.76

The extension gain is now comparable to the rate gain because the ring area for longer loops is larger. As a result, the total gain is larger than the previous results. It is important to have a good echo cancelling level because the gain can be increased substantially. The usage of a better equalizer structure can increase the total gain from 4.19 to 6.00 for 40 dB echo cancellation case (43% increase) or 6.47 to 7.76 for 65 dB echo cancellation case (15% increase). From the above calculations, we find that more promising results are obtained for uniform subscriber density. Nevertheless, the simulation study showed that better use of the loop plant capacity by bit-rate adaptation is a reasonable proposition.

Chapter 6

CONCLUSIONS

This thesis investigates the possible improvements in the utilization of subscriber loop plant due to the deployment of adaptive bit rate transceivers. The investigation has been carried out by the simulation of a multi-rate transceiver working with a set of 35 subscriber loops. These loops ranging from 0-10.7 km are made up using the Bell System loop statistics data. Echo, near-end crosstalk, and coloured Gaussian noise are simulated and introduced into the system to make the study more realistic. On each of the 35 subscriber loops, the quantiser signal to noise ratio and the bit error rate performance at different transmission rates are evaluated.

Two transceiver structures, DFE with and without a forward fractionally spaced equalizer, are studied. The capacity gain is around 5.9 for the fractionally spaced DFE and 5 for the DFE without forward filter. This amounts to 18% increase in capacity gain for the fractionally spaced DFE. If an uniform subscriber density is assumed, the capacity gain is around 6.9 for the fractionally spaced DFE and 5.3 for the DFE without forward filter. This amounts to 30% increase in capacity gain for the fractionally spaced DFE.

The use of a more complex equalizer structure such as an ISI canceller does not improve the performance. A better echo canceller can dramatically improve the system performance in two situations. First, the BER performance is usually worst on the subscriber side due to poor impedance matching at the hybrid. Increasing the echo cancellation level can help to narrow the difference in transmission rate on both sides, which indirectly enhances the system performance because the RA-DSL transmission method requires the transmision rate to be the same in both directions. Second, the NEXT power is proportional to the $f^{1.5}$ of frequency, and therefore insignificant at lower transmission rates. The local echo is the dominant noise source on long loops where the transmission rate is low. A better echo canceller is crucial, because the system performance can be improved significantly by reducing the local echo. An echo canceller with 65 dB echo cancellation is adequate.

Another factor that will affect the system performance is the step size parameter μ. The MSE of the adaptive equalizer is directly proportional to the value of μ, and if μ is not chosen properly, it will cause equalizer instability. The optimal value of μ is difficult to choose because it is a function of the eigenvalues of the auto-correlation matrix of the input vector. A new LMS algorithm described in [29] that is robust to noise in the gradient estimate and has fast convergence without having to use the optimal step size μ may be a solution to the problem.

Some suggestions for future research work are:

1. Introduce more realistic NEXT and FEXT models taking into account cyclostationary nature of crosstalk in the DSL.
2. Introduce a model for the simulation of impulse noise, and incorporate the impulse noise model in the simulation software.
3. Research and test the new LMS algorithm that is insensitive to the step size parameter μ [29].

4. Research the possibility of further capacity gain by using more complex signal processing techniques; for instance, coding and maximum-likelihood sequence reception.

REFERENCES

[1] "Integrated Services Digital Network - Basic Access Interface for use on metallic loops for application on the network side of the NT-Layer 1 specification", ANSI Standard T1.601-1988.

[2] V.Joshi, D.D. Falconer, "Channel capacity bounds for the subscriber loop", Proc. of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, June 1987.

[3] J.W. Lechleider, "The feasibility and advisability of a DSL operating at substantially higher information rates than that of the Basic Access Interface", ECSA document T1E1.4/88-038.

[4] W.D. Grover, "Multi-channel digital subscriber loop interface with automatic maximization of transmission capacity", ATRC Technical Report, March 1988.

[5] A.Shen, W.D. Grover, W.A. Krzymien, "Simulation and experimental studies of a rate-adaptive digital subscriber loop (RA-DSL) transmission method", Proc. of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, June 1989.

[6] W.D Grover, W.A. Krzymien, T.Fong, A.Shen, "Turbo Loop: A Rate-Adaptive Digital Subscriber Loop Technology for Advanced ISDN Access Applications", Proc. of the IEEE Global Telecommunications Conference and Exhibition, Dallas, November 1989.

[7] "1981 Subscriber Loop Survey Results", Bell Canada, May, 1983.

[8] S.V. Ahamed, P.P. Bohn, N.L. Gottfried, " A tutorial on two-wire digital transmission in the loop plant", IEEE Trans. Commun., Vol. COM-29, Nov. 1981.

[9] E. Arnon, W. Chomik, S. Aly, "Performance of 2B1Q transmission system for ISDN basic access", The International Symposium on Subscriber Loops and Services, Boston, Sep. 1988.

[10] N.S. Lin, C.P. Tzeng, "Full duplex data over local loops", IEEE Commun. Mag., Vol. 26, No. 2, Feb. 1988.

[11] R.M. Smith, "ISDN Service Implications Arising From Telephone Plant Testing", Proc. of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, June 1989.

[12] K. Szechenyi, K. Bohm, "Impulsive noise limited transmission performance of ISDN subscriber loops", The International Symposium on Subscriber Loops and Services, Boston, Sep. 1988.

[13] U.H. Qureshi, G.D. Forney, "Performance and properties of a T/2 equalizer", Nat. Telecomm. Conf. Rec., Dec. 1977.

[14] D.L. Duttweiler, J.E. Mazo, and D.G. Messerschmitt, "Error propagation in decision feedback equalizer", IEEE Trans. Inform. Theory, Vol. 20, pp. 490-497, July 1974.

[15] A. Gersho, T.L. Lim, "Adaptive cancellation of intersymbol interference for data transmission", Bell System Technical Journal, Vol. 60, No. 11, Nov. 1981.

[16] M.S. Mueller, J. Salz, "A unified theory of data-aided equalization", Bell System Technical Journal, Vol. 60, No. 9, Nov. 1981.

[17] K. Yao, "On the minimum average probability of error expression for binary pulse communication systems with intersymbol interference" IEEE Trans. Inform. Theory, Vol. 18, pp. 528-531, July 1972.

[18] E.A. Lee, D.G. Messerschmitt, "Digital Communication", Kluwer Academic Publishers, 1988.

[19] M.L. Honig, D.G. Messerschmitt, "Adaptive filters - structures, algorithms, and applications", Kluwer Academic Publishers, 1984.

[20] S. Haykin, "Adaptive filter theory", Prentice-Hall, 1986.

[21] S.V. Ahamed, "Simulation and design studies of digital subscriber lines", Bell System Technical Journal, Vol. 61, No. 6, July 1982.

[22] S.A. Cox, P.F. Adams, "An analysis of digital transmission techniques for the local network", Br. Telecom Tech. Journal, Vol. 3, No. 3, July 1985.

[23] W.J. Chiu, S. Siu, B.Y. Jaw, "Crosstalk performance of MS43 and Bipolar (AMI) codes for ISDN basic access", Proc. of the Int. Conf. on Commun. Seattle, 1987.

[24] R.A. Conte, "A crosstalk model for balanced digital transmissions in multipair cable", AT&T Tech. Jour., Vol. 65, Issue 3, May 1986.

[25] W.T. Vetterling, S.A. Teukolsky, W.H. Press, B.P. Flannery, " Numerical Recipes", Cambridge University Press, 1985.

[26] M.C. Jeruchim, "Techniques for estimating the bit error rate in the simulation of digital communication systems", IEEE J. Select. Areas Commun., SAC-2, No. 1, Jan. 1984.

[27] P.M. Hahn, M.C. Jeruchim, "Developments in the theory and application of importance sampling", IEEE Trans. Commun., Vol. COM-35, No. 7, July 1987.

[28] D. Lu, K. Yao, "Improved importance sampling technique for efficient simulation of digital communication systems", IEEE J. Select. Areas Commun., Vol. 6, No. 1, Jan. 1988.

[29] S. Chung, P. Mcleod, "Stochastic LMS with self adaptive forgetting factor", Proc. of the IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, Victoria, June 1989.

APPENDIX A

Derivation of ABCD matrix for a bridged tap

The ABCD matrix relates the input voltage and current with output voltage and current. The general equations are given by:

$$V_{in} = AV_{out} + BI_{out} \tag{A.1}$$

$$I_{in} = CV_{out} + DI_{out} \tag{A.2}$$

Fig. A.1 shows a 2 port network with a shunt admittance. The ABCD matrix for this shunt admittance Y_{eq} can be derived by solving the following equations.

$$V_{in} = V_{out} \tag{A.3}$$

$$I_{in} = V_{out}Y_{eq} + I_{out} \tag{A.4}$$

Comparing equations (1,3) and (2,4), the ABCD matrix for the shunt admittance can be determined.

$$\begin{bmatrix} V_{in} \\ I_{in} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ Y_{eq} & 1 \end{bmatrix} \begin{bmatrix} V_{out} \\ I_{out} \end{bmatrix} \tag{A.5}$$

Fig. A.2 shows a bridged tap branched out from the main line. Consider the general case where the bridged tap has mixed gauges, and the following ABCD matrix:

$$\begin{bmatrix} A_{composite} & B_{composite} \\ C_{composite} & D_{composite} \end{bmatrix} \tag{A.6}$$

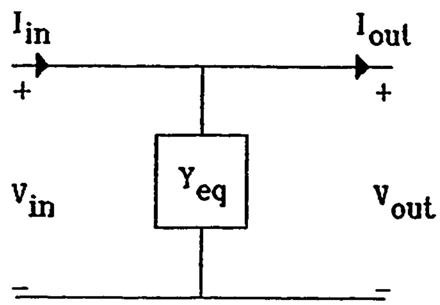


Fig. A.1 Two port network with a shunt admittance

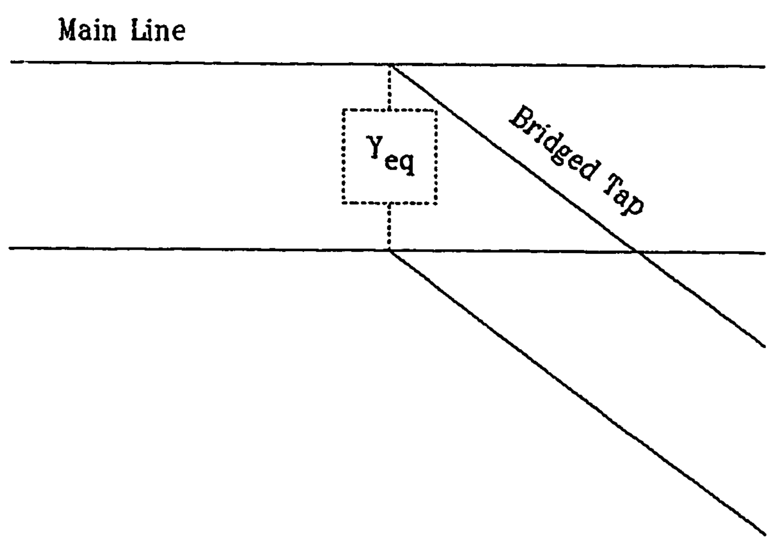


Fig. A.2 Bridged tap on a main line

The equivalent admittance of the bridged tap as seen from the main line is given by:

$$Y_{eq} = \frac{C_{composite} Z_L + D_{composite}}{A_{composite} Z_L + B_{composite}} \qquad (A.7)$$

when $Z_L \rightarrow \infty$ (bridged tap is an open circuit), then

$$Y_{eq} = C_{composite}/A_{composite} \qquad (A.8)$$

After substitution of (8) into (5), the ABCD matrix for a bridged tap is given by:

$$\begin{bmatrix} 1 & 0 \\ \dfrac{C_{composite}}{A_{composite}} & 1 \end{bmatrix} \qquad (A.9)$$

APPENDIX B

Derivation of the transfer function of the subscriber loop and hybrid

(a) Transfer function of the subscriber loop

Fig. B.1 shows the hybrid circuit configuration on the subscriber side and on the central office side. Consider the transfer function for transmission from the central office to the subscriber first. If we assume the line driver has a very low output impedance (ideally zero), then the equivalent impedance of the line as seen by the central office transmitter is

$$Z_{co} = \frac{V_1}{I_1} = \frac{AR_L + B}{CR_L + D} \qquad (B.1)$$

The transmitted signal V_1 at the line interface is

$$V_1 = GV_{co} \frac{Z_{co}}{R_L + Z_{co}} \qquad (B.2)$$

where G is the gain of the line driver, and R_L is the load resistance. The transfer function of the transmitter is:

$$\frac{V_1}{V_{co}} = \frac{GZ_{co}}{R_L + Z_{co}} \qquad (B.3)$$

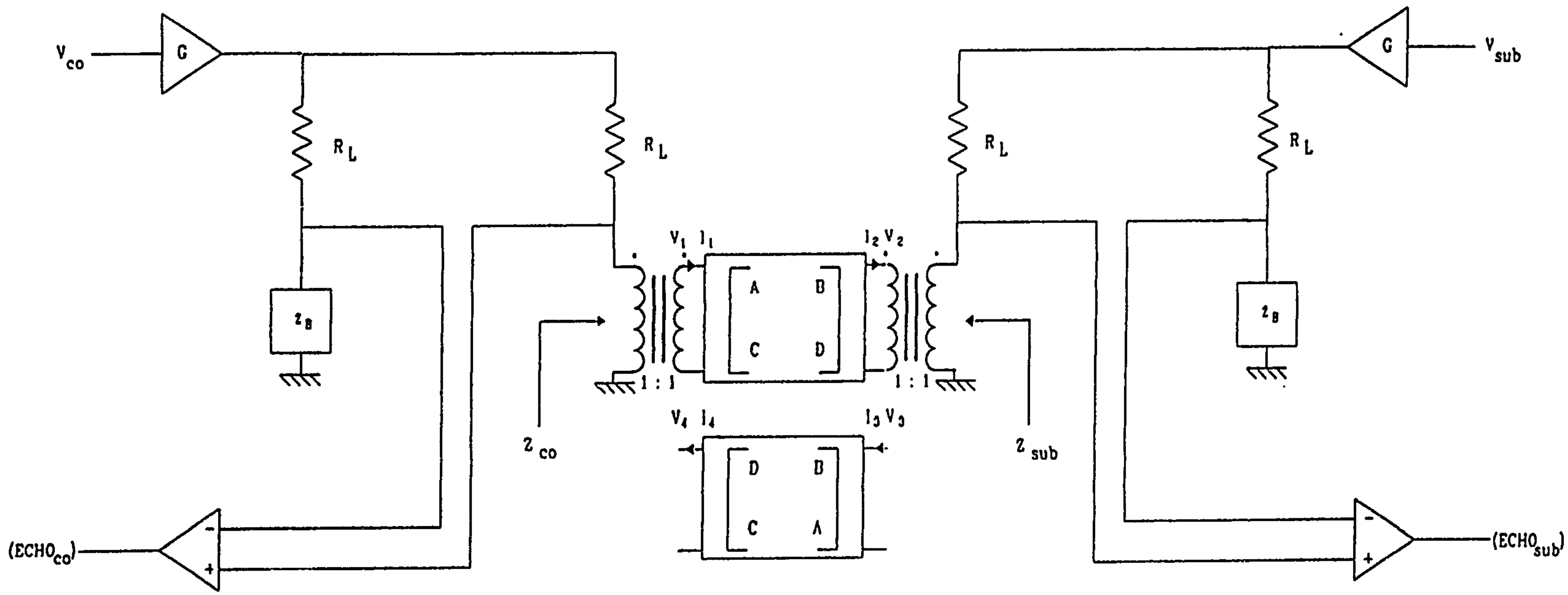


Fig. B.1 The hybrid circuit configuration

The received signal on the subscriber side is given by:

$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} V_2 \\ I_2 \end{bmatrix} \leftrightarrow \begin{bmatrix} V_2 \\ I_2 \end{bmatrix} = \frac{1}{AD - BC} \begin{bmatrix} D & -B \\ -C & A \end{bmatrix} \begin{bmatrix} V_1 \\ I_1 \end{bmatrix} \qquad (B.4)$$

The term AD - BC, which is the determinant of the ABCD matrix, is always equal to unity. We can prove this by first considering a homogenous line. The elements ABCD for a homogenous line are given by:

$$A = \text{Cosh}(\gamma L) \qquad B = Z_0\text{Sinh}(\gamma L)$$

$$C = \frac{\text{Sinh}(\gamma L)}{Z_0} \qquad D = \text{Cosh}(\gamma L) \qquad (B.5)$$

where γ is the propagation constant, Z_0 is the characteristic impedance, L is the length of the cable segment, sinh and cosh are hyperbolic sine and cosine respectively. The determinant of this ABCD matrix is

$$\begin{aligned} \text{Det}[ABCD] &= AD - BC \\ &= \text{Cosh}^2(\gamma L) - \text{Sinh}^2(\gamma L) \\ &= 1 \qquad (B.6) \end{aligned}$$

Therefore, the determinant of the ABCD matrix equals 1 for a homogenous line. Secondly, we will consider a line with two sections of cable. The composite ABCD matrix for this line is given by:

$$\begin{bmatrix} A_{composite} & B_{composite} \\ C_{composite} & D_{composite} \end{bmatrix} = \begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix} \begin{bmatrix} A_2 & B_2 \\ C_2 & D_2 \end{bmatrix} \qquad (B.7)$$

where the determinants of the matrices $ABCD_1$ and $ABCD_2$ equal 1. The determinant of the composite ABCD matrix is given by:

$$\mathrm{Det}[ABCD]_{com} = A_{com}D_{com} - B_{com}C_{com} \qquad (B.8)$$

where

$$A_{com} = A_1A_2 + B_1C_2 \qquad B_{com} = A_1B_2 + B_1D_2$$
$$C_{com} = C_1A_2 + D_1C_2 \qquad D_{com} = C_1B_2 + D_1D_2 \qquad (B.9)$$

Substitution of (9) into (8) yields the result $\mathrm{Det}[ABCD]_{com} = 1$ after some manipulations. Using a similar reasoning, we can extend the derived results to the most general case with N sections of cable. Therefore, the determinant of the ABCD matrix is equal to unity, which means the two-port is reciprocal.

The received signal V2 can then be simplified as:

$$\begin{aligned} V_2 &= DV_1 - BI_1 \\ &= DV_1 - BV_1/Z_{co} \end{aligned} \qquad (B.10)$$

and the transfer function of the line is

$$\frac{V_2}{V_1} = D - \frac{B}{Z_{co}} \qquad (B.11)$$

The transfer function V_2/V_{co} of the transmitter and the line is given by:

$$\frac{V_2}{V_{co}} = \frac{V_2}{V_1}\frac{V_1}{V_{co}} \qquad (B.12)$$

$$= \left[D - \frac{B}{Z_{co}} \right] \frac{GZ_{co}}{R_L + Z_{co}}$$

$$= \frac{G(DZ_{co} - B)}{R_L + Z_{co}} \qquad (B.13)$$

After substitution of (1) into (13), the transfer function of the subscriber loop is given by:

$$\frac{V_2}{V_{co}} = \frac{GR_L}{AR_L + B + CR_L{}^2 + DR_L} \qquad (B.14)$$

Fig. B.2 shows two ABCD matrices with opposite directions of transmission. The ABCD matrix for Fig. B.2a can be expressed as (equation B.4):

$$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} V_2 \\ I_2 \end{bmatrix} \leftrightarrow \begin{bmatrix} V_2 \\ I_2 \end{bmatrix} = \frac{1}{AD - BC} \begin{bmatrix} D & -B \\ -C & A \end{bmatrix} \begin{bmatrix} V_1 \\ I_1 \end{bmatrix} \qquad (B.15)$$

If we substitute:

$$V_2 = V_3 \qquad I_2 = -I_3$$

$$V_1 = V_4 \qquad I_1 = -I_4 \qquad (B.16)$$

into (15), we will get the ABCD matrix for the other direction of transmission:

$$\begin{bmatrix} V_3 \\ I_3 \end{bmatrix} = \begin{bmatrix} D & B \\ C & A \end{bmatrix} \begin{bmatrix} V_4 \\ I_4 \end{bmatrix} \qquad (B.17)$$

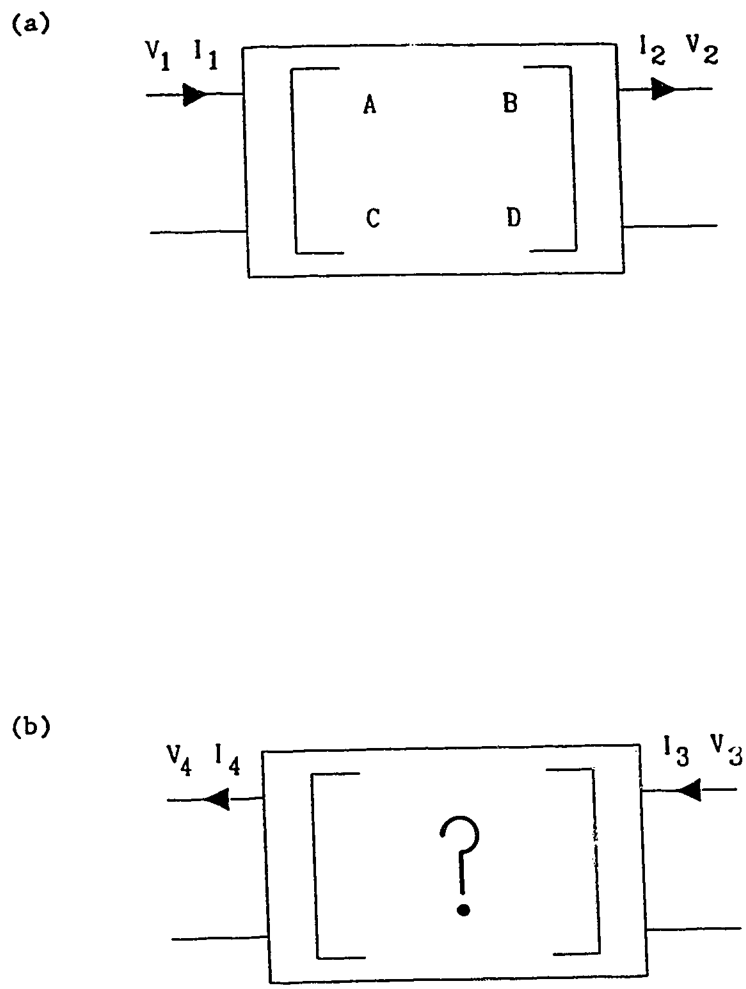


Fig. B.2 ABCD matrix in (a) one direction (b) the opposite direction

As a result, the A and D terms of the ABCD matrix for the other direction of transmission are simply interchanged, and the resultant ABCD matrix is shown in Fig. B.1. The transfer function for transmission from the subscriber to the central office is now derived. The equivalent impedance of the line as seen by the subscriber is Z_{sub}.

$$Z_{sub} = \frac{V_3}{I_3} = \frac{DR_L + B}{CR_L + A} \tag{B.18}$$

The transmitted signal V_3 at the network interface is

$$V_3 = GV_{sub} \frac{Z_{sub}}{R_L + Z_{sub}} \tag{B.19}$$

The transfer function of the transmitter is:

$$\frac{V_3}{V_{sub}} = \frac{GZ_{sub}}{R_L + Z_{sub}} \tag{B.20}$$

The received signal at the central office is given by:

$$\begin{bmatrix} V_3 \\ I_3 \end{bmatrix} = \begin{bmatrix} D & B \\ C & A \end{bmatrix} \begin{bmatrix} V_4 \\ I_4 \end{bmatrix} \leftrightarrow \begin{bmatrix} V_4 \\ I_4 \end{bmatrix} = \frac{1}{AD - BC} \begin{bmatrix} A & -B \\ -C & D \end{bmatrix} \begin{bmatrix} V_3 \\ I_3 \end{bmatrix} \tag{B.21}$$

$$\begin{aligned} \therefore \qquad V_4 &= AV_3 - BI_3 \\ &= AV_3 - BV_3/Z_{sub} \end{aligned} \tag{B.22}$$

and the transfer function is

$$\frac{V_4}{V_3} = A - \frac{B}{Z_{sub}} \tag{B.23}$$

The transfer function V_4/V_{sub} is given by:

$$\frac{V_4}{V_{sub}} = \frac{V_4}{V_3}\frac{V_3}{V_{sub}} \tag{B.24}$$

$$= \left[A - \frac{B}{Z_{sub}}\right]\frac{GZ_{sub}}{R_L + Z_{sub}}$$

$$= \frac{G(AZ_{sub} - B)}{R_L + Z_{sub}} \tag{B.25}$$

Substitution of (18) into (25) yields the transfer function:

$$\frac{V_4}{V_{sub}} = \frac{GR_L}{AR_L + B + CR_L^2 + DR_L} \tag{B.26}$$

The transfer functions for the two directions of transmission are the same due to the reciprocity of the ABCD matrix.

(b) The transhybrid transfer function

In the following section, the transhybrid transfer function will be derived. The local echo at the central office is given by:

$$Echo_{co} = GV_{co}\left[\frac{Z_{co}}{R_L + Z_{co}} - \frac{Z_B}{R_L + Z_B}\right] \qquad (B.27)$$

After substitution of (1) into (27), the transhybrid transfer function at the central office is given by:

$$\frac{Echo_{co}}{V_{co}} = G\left[\frac{AR_L + B}{AR_L + B + CR_L^2 + DR_L} - \frac{Z_B}{R_L + Z_B}\right] \qquad (B.28)$$

The echo on the subscriber side is given by:

$$Echo_{sub} = GV_{sub}\left[\frac{Z_{sub}}{R_L + Z_{sub}} - \frac{Z_B}{R_L + Z_B}\right] \qquad (B.29)$$

After substitution of (18) into (29), the transhybrid transfer function on the subscriber side is given by:

$$\frac{Echo_{sub}}{V_{sub}} = G\left[\frac{DR_L + B}{AR_L + B + CR_L^2 + DR_L} - \frac{Z_B}{R_L + Z_B}\right] \qquad (B.30)$$

APPENDIX C

Transfer functions of the transceiver

In this appendix, the transfer function of each of the transceiver modules will be shown.

The square-root raised cosine transmit and receive filter:

$$H_1(f) = \begin{cases} 1 & |f| \leq (1-\alpha)/2T \\ \left(0.5\left(1-\sin\left(\frac{\pi T}{\alpha}\left(|f| - \frac{1}{2T}\right)\right)\right)\right)^{0.5} & (1-\alpha)/2T < |f| \leq (1+\alpha)/2T \\ 0 & (1+\alpha)/2T < |f| \end{cases}$$

$$0 \leq \alpha \leq 1 \qquad \text{(C.1)}$$

The line driver:

$$H_{2co}(f) = G_1 \text{ (constant gain)} \qquad \text{(C.2a)}$$

$$H_{2sub}(f) = G_2 \qquad \text{(C.2b)}$$

The transhybrid transfer functions: (refer to equations B.28 and B.30)

$$H_{3co}(f) = \left(\frac{AR_L + B}{AR_L + B + CR_L^2 + DR_L} - \frac{Z_B}{R_L + Z_B}\right) \qquad \text{(C.3a)}$$

$$H_{3sub}(f) = \left(\frac{DR_L + B}{AR_L + B + CR_L^2 + DR_L} - \frac{Z_B}{R_L + Z_B}\right) \qquad \text{(C.3b)}$$

The hybrid voltage divider:

The hybrid voltage divider can be expressed as (refer to Fig. B.1):

$$V_1 = V_{co} \frac{Z_{co}}{R_L + Z_{co}} \tag{C.4a}$$

Substitution of equation (B.1) into (C.4a) gives the following result:

$$H_{4co}(f) = \frac{AR_L + B}{AR_L + B + CR_L^2 + DR_L} \tag{C.4b}$$

Similarly, the hybrid voltage divider on the subscriber side can be expressed as (refer to Fig. B.1):

$$V_3 = V_{sub} \frac{Z_{sub}}{R_L + Z_{sub}} \tag{C.4c}$$

Substitution of equation (B.18) into (C.4c) gives the following result:

$$H_{4sub}(f) = \frac{DR_L + B}{AR_L + B + CR_L^2 + DR_L} \tag{C.4d}$$

The subscriber loop: (refer to B.14 and B.26)

$$H_{5co}(f) = H_{5sub}(f) = \frac{R_L}{AR_L + B + CR_L^2 + DR_L} = H_5(f) \tag{C.5}$$

The MR fixed equalizer:

$$H_6(f) = 1 - z^{-1}\Big|_{z=e^{j\omega T}} = (1 - \cos(\omega T)) + j\sin(\omega T); \quad \omega = 2\pi f \tag{C.6}$$

The echo canceller:

$$H_7(f) = K \text{ (attenuation constant)} \tag{C.7}$$

The NEXT transfer function:

$$H_8(f) = (K_{NEXT} f^{1.5})^{0.5} \tag{C.8}$$

The composite transfer functions of the signal and different noises paths are shown below:

Transfer function of the signal path:

$$\begin{aligned} H_{co}(f) &= H_1^2(f)*H_{2co}(f)*H_5(f)*H_6(f) \\ H_{sub}(f) &= H_1^2(f)*H_{2sub}(f)*H_5(f)*H_6(f) \end{aligned} \tag{C.9}$$

Transfer function of the echo path:

$$\begin{aligned} H_{co}(f) &= H_1^2(f)*H_{2co}(f)*H_{3co}(f)*H_6(f)*H_7(f) \\ H_{sub}(f) &= H_1^2(f)*H_{2sub}(f)*H_{3sub}(f)*H_6(f)*H_7(f) \end{aligned} \tag{C.10}$$

Transfer function of the NEXT path:

$$\begin{aligned} H_{co}(f) &= H_1^2(f)*H_{2co}(f)*H_{4co}(f)*H_6(f)*H_8(f) \\ H_{sub}(f) &= H_1^2(f)*H_{2sub}(f)*H_{4sub}(f)*H_6(f)*H_8(f) \end{aligned} \tag{C.11}$$

Transfer function of the coloured noise path:

$$H_{co}(f) = H_{sub}(f) = H_1(f)*H_6(f) \tag{C.12}$$

Transfer function for power scaling at line and network termination:

$$H_{co}(f) = H_1(f)*H_{4co}(f)$$

$$H_{sub}(f) = H_1(f)*H_{4sub}(f) \quad (C.13)$$

APPENDIX D

Program listings

```
C*******************************************************************C
C                         PROGRAM    INTERPOL                       C
C*******************************************************************C
C                                                                   C
C  Purpose : This program will read in 4 sets of RLGC parameters for C
C            each of the 19, 22, 24 and 26 AWG cables. There are 38  C
C            data for each set of RLG parameters, and C is constant  C
C            at all frequencies. For the purpose of simulation,      C
C            intermediate RLGC parameters between frequencies        C
C            (128 in total) will be interpolated using the cubic     C
C            spline algorithm. The characteristic impedance (Z0) and C
C            propagation constant (Gamma) for each set of cables at  C
C            some pre-defined incremental frequency will be calculated C
C            using those RLGC parameters.                            C
C                                                                   C
C  Input file - Interpol.in                                         C
C  Log file   - Interpol.fil                                        C
C                                                                   C
C  Data required:                                                   C
C                      c - capacitance in 10^-6 F/mile              C
C                  nfreq - number of frequency components           C
C                   nrun - number of runs at different symbol rates C
C                numfile - number of files to be read               C
C                  indir - input file directory                     C
C              outdir(n) - a list of output-file directories        C
C                          ( n <= 100)                              C
C            baudrate(n) - a list of symbol rates ( n <= 100 )      C
C                    num - the index for which the outdir(n) and    C
C                          baudrate(n) will be used                 C
C                  fname - input file name                          C
C                          (1): 26 AWG at assigned temperature      C
C                          (2): 24 AWG at assigned temperature      C
C                          (3): 22 AWG at assigned temperature      C
C                          (4): 19 AWG at assigned temperature      C
C                                                                   C
C*******************************************************************C
      REAL FREQ(38),R(38),L(38),G(38),R2ND(38),L2ND(38),G2ND(38)
      REAL BAUDRATE(100)
      REAL RINT,LINT,GINT,FREQINT
      REAL DF,TPI,C
      INTEGER NFREQ,NRUN,NUMFILE,NDATA
      INTEGER NUM(100)
      COMPLEX Z0(256),GAMMA(256),CDEN,CNUM,CMPLX
      CHARACTER*30 NAME1,NAME2
      CHARACTER*12 FNAME(4)
```

```
      CHARACTER*7 INDIR
      CHARACTER*12 OUTDIR(100)
      TPI=6.283185308
      NDATA=38
C
C          READ INPUT DATA
C
      OPEN(1,FILE='INTERPOL.IN')
       READ(1,*) C
       READ(1,*) NFREQ,NRUN,NUMFILE
       READ(1,*) (NUM(I),I=1,NRUN)
       READ(1,*) INDIR
       READ(1,*) FNAME(1),FNAME(2)
       READ(1,*) FNAME(3),FNAME(4)
        DO 1 I=1,NUMFILE
1        READ(1,*) BAUDRATE(I),OUTDIR(I)
      CLOSE(1)
      OPEN(3,FILE='INTERPOL.FIL')
C
C          INTERPOLATE RLGC PARAMETERS TO CALCULATE Z0 AND GAMMA
C
      DO 5 II=1,NRUN
       WRITE(3,*)
       WRITE(3,*) 'RUN # ', II
       FREQINT=(BAUDRATE(NUM(II))/NFREQ)/1E3
        DO 5 KK=1,4
         NAME1=INDIR//FNAME(KK)
         NAME2=OUTDIR(NUM(II))//FNAME(KK)
         OPEN(1,FILE=NAME1)
         OPEN(2,FILE=NAME2)
         WRITE(3,*) NAME1,NAME2
          DO 10 I=1,NDATA
10         READ(1,*) FREQ(I),R(I),L(I),G(I)
C
C          CALCULATE THE SECOND DERIVATIVE FOR R, L, AND G. THE
C          SECOND DERIVATIVE AT BOTH ENDS IS SET TO ZERO, GIVING
C          THE NATURAL CUBIC SPLINE.
C
         CALL SPLINE(FREQ,R,NDATA,1E30,1E30,R2ND)
         CALL SPLINE(FREQ,L,NDATA,1E30,1E30,L2ND)
         CALL SPLINE(FREQ,G,NDATA,1E30,1E30,G2ND)
C
C          FOR I=1 (DC), Z0 AND GAMMA ARE NOT DEFINED. GAMMA IS
C          SET TO ZERO, AND Z0 IS SET AS THE DC RESISTANCE/MILE.
C
         Z0(1)=R(1)
         GAMMA(1)=0.0
C
C          Z0 AND GAMMA AT OTHER FREQUENCIES
C
         DO 15 I=1,NFREQ
          DF=FREQINT*I
C
C          INTERPOLATE RLGC PARAMETERS AT SPECIFIED FREQUENCY
```

```
C
         CALL SPLINT(FREQ,R,R2ND,NDATA,DF,RINT)
         CALL SPLINT(FREQ,L,L2ND,NDATA,DF,LINT)
         CALL SPLINT(FREQ,G,G2ND,NDATA,DF,GINT)
C
C         ZO AND GAMMA ARE CALCULATED
C
         CNUM=CMPLX(RINT,0)+CMPLX(0,DF*TPI*LINT)
         CDEN=CMPLX(GINT,0)+CMPLX(0,C*DF*TPI*1E3)
         ZO(I+1)=CSQRT(CNUM/CDEN)*1E3
         GAMMA(I+1)=CSQRT(CNUM*CDEN)*1E-3
15    CONTINUE
C
C         OUTPUT RESULTS
C
       DO 20 I=1,NFREQ+1
20     WRITE(2,*) ZO(I),GAMMA(I)
      CLOSE(1)
      CLOSE(2)
5     CONTINUE
      STOP
      END
C*********************************************************************C
C                                                                     C
C     THIS SUBROUTINE WILL CALCULATE THE SECOND DERIVATIVE OF THE     C
C     INTERPOLATING FUNCTION AT POINT Z1                              C
C                                                                     C
C        X = ARRAY OF LENGTH N ( X1 < X2 < ... XN )                   C
C        Y = F(X), ARRAY OF LENGTH N                                  C
C        N = DIMENSION OF ARRAY                                       C
C  YP1,YPN = FIRST DERIVATIVE OF THE INTERPOLATING FUNCTION AT POINT  C
C            1 AND POINT N. IF YP1 AND/OR YPN ARE EQUAL TO 1E30 OR    C
C            LARGER, THE SECOND DERIVATIVE IS SET TO ZERO ON THAT     C
C            BOUNDARY                                                 C
C       Y2 = ARRAY OF LENGTH N CONTAINING THE SECOND DERIVATIVES OF   C
C            THE INTERPOLATING FUNCTION                               C
C                                                                     C
C*********************************************************************C
      SUBROUTINE SPLINE(X,Y,N,YP1,YPN,Y2)
      PARAMETER (NMAX=100)
      DIMENSION X(N),Y(N),Y2(N),U(NMAX)
      IF (YP1.GT..99E30) THEN
        Y2(1)=0.
        U(1)=0.
      ELSE
        Y2(1)=-0.5
        U(1)=(3./(X(2)-X(1)))*((Y(2)-Y(1))/(X(2)-X(1))-YP1)
      ENDIF
      DO 11 I=2,N-1
        SIG=(X(I)-X(I-1))/(X(I+1)-X(I-1))
        P=SIG*Y2(I-1)+2.
        Y2(I)=(SIG-1.)/P
        U(I)=(6.*((Y(I+1)-Y(I))/(X(I+1)-X(I))-(Y(I)-Y(I-1))
     *      /(X(I)-X(I-1)))/(X(I+1)-X(I-1))-SIG*U(I-1))/P
```

```
11    CONTINUE
      IF (YPN.GT..99E30) THEN
        QN=0.
        UN=0.
      ELSE
        QN=0.5
        UN=(3./(X(N)-X(N-1)))*(YPN-(Y(N)-Y(N-1))/(X(N)-X(N-1)))
      ENDIF
      Y2(N)=(UN-QN*U(N-1))/(QN*Y2(N-1)+1.)
      DO 12 K=N-1,1,-1
        Y2(K)=Y2(K)*Y2(K+1)+U(K)
12    CONTINUE
      RETURN
      END
C***********************************************************************C
C                                                                       C
C       GIVEN POINT Xi, THIS ROUTINE WILL RETURN THE VALUE OF Yi USING  C
C       THE CUBIC-SPLINE INTERPOLATION.                                 C
C                                                                       C
C         XA = ARRAY OF LENGTH N                                        C
C         YA = ARRAY OF LENGTH N                                        C
C          N = DIMENSION OF ARRAY                                       C
C        Y2A = OUTPUT ARRAY OF LENGTH N FROM SUBROUTINE SPLINE          C
C          X = INPUT VALUE                                              C
C          Y = CUBIC SPLINE INTERPOLATED VALUE OF Y AT X                C
C                                                                       C
C***********************************************************************C
      SUBROUTINE SPLINT(XA,YA,Y2A,N,X,Y)
      DIMENSION XA(N),YA(N),Y2A(N)
      KLO=1
      KHI=N
1     IF (KHI-KLO.GT.1) THEN
        K=(KHI+KLO)/2
        IF(XA(K).GT.X)THEN
          KHI=K
        ELSE
          KLO=K
        ENDIF
      GOTO 1
      ENDIF
      H=XA(KHI)-XA(KLO)
      IF (H.EQ.0.) PAUSE 'Bad XA input.'
      A=(XA(KHI)-X)/H
      B=(X-XA(KLO))/H
      Y=A*YA(KLO)+B*YA(KHI)+
     *      ((A**3-A)*Y2A(KLO)+(B**3-B)*Y2A(KHI))*(H**2)/6.
      RETURN
      END
```

```
C*********************************************************************C
C                          PROGRAM      DTXLINE                       C
C*********************************************************************C
C                                                                     C
C          This program will generate the ABCD matrix of a subscriber C
C     loop. The loop can have any configuration including bridge tap  C
C     on a bridge tap.The ABCD matrix of a cable section is calculated C
C     from the characteristic impedance and propagation constant. The C
C     ABCD matrix of a loop is obtained by multiplying all these      C
C     ABCD matrices of the cable section.                             C
C                                                                     C
C     Input file - dtxline.in                                         C
C     Log file   - dtxline.fil                                        C
C                                                                     C
C     Data required:                                                  C
C                    nfreq - number of frequency components           C
C                  numfile - number of files to be read               C
C                    xloop - number of runs at different symbol rates C
C                 indir(n) - a list of input-data-file directories    C
C                            (n ≤ 100)                                C
C                outdir(n) - a list of output-file directories        C
C                            (n ≤ 100)                                C
C                      num - the index for which indir(n) and outdir(n) C
C                            will be used                             C
C                     name - input data file name                     C
C                    nfile - the path and file name for loop          C
C                            configuration                            C
C                                                                     C
C*********************************************************************C
      REAL*8 LENGTH
      INTEGER NFREQ,NDIM,TEMP,GAUGE,TYPE,NUMFILE,XLOOP
      INTEGER NUM(100)
      COMPLEX*8  CMPLX
      COMPLEX*16 DCMPLX
      COMPLEX*16 ABCD_FINAL(2,2,256),ABCD_T(2,2,256)
      CHARACTER*12 NAME
      CHARACTER*35 NFILE
      CHARACTER*12 INDIR(100)
      CHARACTER*14 OUTDIR(100)
C
C          READ INPUT DATA
C
      OPEN(1,FILE='DTXLINE.IN')
       READ(1,*) NFREQ,NUMFILE,XLOOP
       READ(1,*) (NUM(I),I=1,XLOOP)
       READ(1,*) NFILE
       READ(1,*) NAME
        DO 10 I=1,NUMFILE
10        READ(1,*) INDIR(I),OUTDIR(I)
      CLOSE(1)
C
      OPEN(2,FILE='DTXLINE.FIL')
      OPEN(3,FILE=NFILE)
C
```

```
C         CALCULATE ABCD MATRIX
C
      DO 20 II=1,XLOOP
       WRITE(2,*) 'RUN # ',II
       WRITE(2,*)
       WRITE(2,*) 'INPUT FILE :'
       WRITE(2,*)
C
C         INITIALIZATION
C
      NDIM=256
      ABCD_FINAL(1,1,1)=DCMPLX(0,0)
      ABCD_FINAL(1,2,1)=DCMPLX(0,0)
      ABCD_FINAL(2,1,1)=DCMPLX(0,0)
      ABCD_FINAL(2,2,1)=DCMPLX(0,0)
C
       DO 30 I=2,NFREQ+1
        ABCD_FINAL(1,1,I)=DCMPLX(1.,0)
        ABCD_FINAL(1,2,I)=DCMPLX(0,0)
        ABCD_FINAL(2,1,I)=DCMPLX(0,0)
30     ABCD_FINAL(2,2,I)=DCMPLX(1.,0)
C
C         READ IN LOOP CONFIGURATION
C            TYPE=0 : TERMINATION OF A CABLE SECTION
C            TYPE=1 : CABLE SECTION WITH DIFFERENT GAUGE
C            TYPE=2 : BRIDGED TAP
C
       DO 40 I=1,300
        READ(3,*) TYPE,GAUGE,TEMP,LENGTH
C
         IF(TYPE.EQ.0) THEN
C
C         OUTPUT RESULTS
C
          NFILE=OUTDIR(NUM(II))//NAME
          OPEN(1,FILE=NFILE)
           DO 50 J=1,NFREQ+1
           WRITE(1,*) CMPLX(ABCD_FINAL(1,1,J)),CMPLX(ABCD_FINAL(1,2,J))
50         WRITE(1,*) CMPLX(ABCD_FINAL(2,1,J)),CMPLX(ABCD_FINAL(2,2,J))
          CLOSE(1)
C
           WRITE(2,*)
           WRITE(2,*) 'OUTPUT FILE NAME = ',NFILE
           WRITE(2,*)
          GOTO 20
         ELSEIF(TYPE.EQ.2) THEN
          CALL BRIDGE(NDIM,NFREQ,ABCD_FINAL,INDIR(NUM(II)))
         ELSEIF(TYPE.EQ.1) THEN
          CALL GT_CONVERT(NFILE,TEMP,GAUGE,INDIR(NUM(II)))
          OPEN(4,FILE=NFILE)
          CALL DEFINE_ABCD(NDIM,NFREQ,ABCD_T,LENGTH)
          CALL XMAT(ABCD_FINAL,ABCD_T,NDIM,NFREQ)
         ELSE
          STOP 'INPUT ERROR'
```

```
        ENDIF
40     CONTINUE
20    REWIND(3)
      STOP
      END
C**********************************************************************C
C                                                                      C
C        This subroutine will use variables TEMP and GAUGE to          C
C     determine the input data file.                                   C
C                                                                      C
C      temp - temperature of cable (0, 70, 120 degrees F)              C
C     gauge - size of cable (19, 22, 24, 26 AWG)                       C
C       dir - the drive and directory of input file                    C
C         a - complete path and file name of input file                C
C                                                                      C
C**********************************************************************C
      SUBROUTINE GT_CONVERT(A,TEMP,GAUGE,DIR)
      CHARACTER*35 A
      CHARACTER*12 DIR
      INTEGER TEMP,GAUGE
       IF(GAUGE.EQ.26)GOTO 10
       IF(GAUGE.EQ.24)GOTO 20
       IF(GAUGE.EQ.22)GOTO 30
       IF(GAUGE.EQ.19)GOTO 40
      STOP 'INPUT ERROR'
 10    IF(TEMP.EQ.70) A=DIR//'G26T70.DAT'
       IF(TEMP.EQ.0) A=DIR//'G26T00.DAT'
       IF(TEMP.EQ.120) A=DIR//'G26T120.DAT'
        WRITE(2,*) A
         GO TO 50
 20    IF(TEMP.EQ.70) A=DIR//'G24T70.DAT'
       IF(TEMP.EQ.0) A=DIR//'G24T00.DAT'
       IF(TEMP.EQ.120) A=DIR//'G24T120.DAT'
        WRITE(2,*) A
         GO TO 50
 30    IF(TEMP.EQ.70) A=DIR//'G22T70.DAT'
       IF(TEMP.EQ.0) A=DIR//'G22T00.DAT'
       IF(TEMP.EQ.120) A=DIR//'G22T120.DAT'
        WRITE(2,*) A
         GO TO 50
 40    IF(TEMP.EQ.70) A=DIR//'G19T70.DAT'
       IF(TEMP.EQ.0) A=DIR//'G19T00.DAT'
       IF(TEMP.EQ.120) A=DIR//'G19T120.DAT'
        WRITE(2,*) A
 50   RETURN
      END
```

```
C**********************************************************************C
C                                                                      C
C          Two complex matrices ABCDF and ABCDT will be multiplied     C
C     together, and the resulting matrix will be put into ABCDF.       C
C                                                                      C
C     ndim - physical dimension of 2 by 2 array                        C
C     nsym - actual dimension of 2 by 2 array                          C
C                                                                      C
C**********************************************************************C
      SUBROUTINE XMAT(ABCDF,ABCDT,NDIM,NSYM)
      INTEGER NDIM,NSYM
      COMPLEX*16 ABCDF(2,2,NDIM),ABCDT(2,2,NDIM)
      COMPLEX*16 DCMPLX,TEMPOR
C
C       SPECIAL CASE (DC)
C
        ABCDF(1,1,1)=ABCDF(1,1,1)+ABCDT(1,1,1)
        ABCDF(1,2,1)=DCMPLX(0,0)
        ABCDF(2,1,1)=DCMPLX(0,0)
        ABCDF(2,2,1)=DCMPLX(0,0)
C
      DO 10 I=2,NSYM+1
        TEMPOR=ABCDF(1,1,I)*ABCDT(1,1,I)+ABCDF(1,2,I)*ABCDT(2,1,I)
        ABCDF(1,2,I)=ABCDF(1,1,I)*ABCDT(1,2,I)+ABCDF(1,2,I)*ABCDT(2,2,I)
        ABCDF(1,1,I)=TEMPOR
        TEMPOR=ABCDT(1,1,I)*ABCDF(2,1,I)+ABCDF(2,2,I)*ABCDT(2,1,I)
        ABCDF(2,2,I)=ABCDF(2,1,I)*ABCDT(1,2,I)+ABCDF(2,2,I)*ABCDT(2,2,I)
        ABCDF(2,1,I)=TEMPOR
10    CONTINUE
      RETURN
      END
C**********************************************************************C
C                                                                      C
C          This subroutine will calculate the ABCD parameters at each  C
C     frequency by using the characteristic impedance (ZO) and the     C
C     propagation constant (GAMMA).                                    C
C                                                                      C
C       ndim - physical dimension of a 2 by 2 array                    C
C       nsym - actual dimension of a 2 by 2 array                      C
C       abcd - 3 dimensional array (2 by 2 array of length nsym)       C
C     length - length of cable in feet                                 C
C                                                                      C
C**********************************************************************C
      SUBROUTINE DEFINE_ABCD(NDIM,NSYM,ABCD,LENGTH)
      INTEGER NDIM,NSYM
      COMPLEX*16 ABCD(2,2,NDIM)
      COMPLEX*16 DCMPLX,X,GAMMA,ZO
      REAL*8 LENGTH,R
      READ(4,*) ZO,GAMMA
C
C          SPECIAL CASE (DC), WHERE R IS THE DC RESISTANCE
C
       R=DREAL(ZO)*LENGTH/5280.0
       ABCD(1,1,1)=DCMPLX(R,0)
```

```
      ABCD(1,2,1)=DCMPLX(0,0)
      ABCD(2,1,1)=DCMPLX(0,0)
      ABCD(2,2,1)=DCMPLX(0,0)
C
C          AT OTHER FREQUENCIES
C
      DO 10 I=2,NSYM+1
      READ(4,*) Z0,GAMMA
       X=GAMMA*LENGTH/5280.0
       ABCD(1,1,I)=(CDEXP(X)+CDEXP(-X))/2.
       ABCD(1,2,I)=(CDEXP(X)-CDEXP(-X))*Z0/2.
       ABCD(2,1,I)=(CDEXP(X)-CDEXP(-X))/(2.*Z0)
       ABCD(2,2,I)=ABCD(1,1,I)
10    CONTINUE
      REWIND(4)
      RETURN
      END
C*******************************************************************C
C                                                                   C
C          This subroutine will calculate the ABCD parameters for a C
C     bridged tap. The ABCD matrix of the main loop is then multiplied C
C     with the ABCD matrix of the bridged tap.                     C
C                                                                   C
C      ndim = physical dimension of a 2 by 2 array                 C
C      nsym = actual dimension of a 2 by 2 array                   C
C     abcdf = abcd matrix of the main loop                          C
C       dir = the drive and directory name of input data file       C
C                                                                   C
C*******************************************************************C
      SUBROUTINE BRIDGE(NDIM,NSYM,ABCDF,DIR)
      INTEGER NDIM,NSYM,TYPE,GAUGE,TEMP
      COMPLEX*16 ABCDF(2,2,NDIM)
      COMPLEX*16 ABCD1(2,2,256),ABCD2(2,2,256)
      COMPLEX*16 DCMPLX
      REAL*8 LENGTH
      CHARACTER*35 NFILE
      CHARACTER*12 DIR
C
C          SPECIAL CASE (DC)
C
      ABCD1(1,1,1)=DCMPLX(0,0)
      ABCD1(1,2,1)=DCMPLX(0,0)
      ABCD1(2,1,1)=DCMPLX(0,0)
      ABCD1(2,2,1)=DCMPLX(0,0)
C
      DO 1 I=2,NSYM+1
       ABCD1(1,1,I)=DCMPLX(1.,0.)
       ABCD1(1,2,I)=DCMPLX(0.,0.)
       ABCD1(2,1,I)=DCMPLX(0.,0.)
1     ABCD1(2,2,I)=DCMPLX(1.,0.)
C
C          READ IN THE CONFIGURATION OF THE BRIDGED TAP
C            TYPE=0: A SINGLE GAUGE BRIDGED TAP
C            TYPE=1: A COMPOSITE GAUGES BRIDGED TAP
```

```
C           TYPE-2: BRIDGED TAP ON A BRIDGED TAP
C
      DO 10 I=1,100
       READ(3,*) TYPE,GAUGE,TEMP,LENGTH
        IF(TYPE.EQ.0) THEN
          ABCD2(1,1,1)=DCMPLX(0,0)
          ABCD2(1,2,1)=DCMPLX(0,0)
          ABCD2(2,1,1)=DCMPLX(0,0)
          ABCD2(2,2,1)=DCMPLX(0,0)
           DO 20 J=2,NSYM+1
            ABCD2(2,1,J)=ABCD1(2,1,J)/ABCD1(1,1,J)
            ABCD2(1,1,J)=DCMPLX(1.,0.)
            ABCD2(1,2,J)=DCMPLX(0.,0.)
            ABCD2(2,2,J)=DCMPLX(1.,0.)
20         CONTINUE
C
C          ABCD MATRIX OF MAIN LOOP AND BRIDGED TAP ARE MULTIPLIED
C
          CALL XMAT(ABCDF,ABCD2,NDIM,NSYM)
          RETURN
        ELSEIF(TYPE.EQ.2) THEN
         CALL BBRIDGE(NDIM,NSYM,ABCD1,DIR)
        ELSEIF(TYPE.EQ.1) THEN
         CALL GT_CONVERT(NFILE,TEMP,GAUGE,DIR)
         OPEN(4,FILE=NFILE)
         CALL DEFINE_ABCD(NDIM,NSYM,ABCD2,LENGTH)
         CALL XMAT(ABCD1,ABCD2,NDIM,NSYM)
        ELSE
         STOP 'INPUT ERROR'
       ENDIF
10    CONTINUE
      RETURN
      END
C*********************************************************************C
C                                                                     C
C          This subroutine will calculate the ABCD parameters for a   C
C     bridged tap on a bridged tap. The ABCD matrix of the bridged tap C
C     is then multiplied with the ABCD matrix of the bridged tap on a  C
C     bridged tap.                                                    C
C                                                                     C
C      ndim - physical dimension of a 2 by 2 array                   C
C      nsym - actual dimension of a 2 by 2 array                     C
C     abcdf - abcd matrix of the bridged tap                          C
C       dir - the drive and directory name of input data file         C
C                                                                     C
C*********************************************************************C
      SUBROUTINE BBRIDGE(NDIM,NSYM,ABCDF,DIR)
      INTEGER NDIM,NSYM,TYPE,GAUGE,TEMP
      COMPLEX*16 ABCDF(2,2,NDIM)
      COMPLEX*16 ABCD1(2,2,256),ABCD2(2,2,256)
      COMPLEX*16 DCMPLX
      REAL*8 LENGTH
      CHARACTER*35 NFILE
      CHARACTER*12 DIR
```

```
C
C         SPECIAL CASE (DC)
C
      ABCD1(1,1,1)=DCMPLX(0,0)
      ABCD1(1,2,1)=DCMPLX(0,0)
      ABCD1(2,1,1)=DCMPLX(0,0)
      ABCD1(2,2,1)=DCMPLX(0,0)
C
      DO 1 I=2,NSYM+1
       ABCD1(1,1,I)=DCMPLX(1.,0)
       ABCD1(1,2,I)=DCMPLX(0,0)
       ABCD1(2,1,I)=DCMPLX(0,0)
1     ABCD1(2,2,I)=DCMPLX(1.,0)
C
C      READ IN THE CONFIGURATION OF THE BRIDGED TAP
C           TYPE=0: A SINGLE GAUGE BRIDGED TAP
C           TYPE=1: A COMPOSITE GAUGES BRIDGED TAP
C
      DO 10 I=1,100
       READ(3,*) TYPE,GAUGE,TEMP,LENGTH
C
       IF(TYPE.EQ.0) THEN
         ABCD2(1,1,1)=DCMPLX(0,0)
         ABCD2(1,2,1)=DCMPLX(0,0)
         ABCD2(2,1,1)=DCMPLX(0,0)
         ABCD2(2,2,1)=DCMPLX(0,0)
          DO 20 J=2,NSYM+1
            ABCD2(2,1,J)=ABCD1(2,1,J)/ABCD1(1,1,J)
            ABCD2(1,1,J)=DCMPLX(1.,0.)
            ABCD2(1,2,J)=DCMPLX(0.,0.)
20        ABCD2(2,2,J)=DCMPLX(1.,0.)
C
C          ABCD MATRIX OF BRIDGED TAP AND BT ON BT ARE MULTIPLIED
C
         CALL XMAT(ABCDF,ABCD2,NDIM,NSYM)
         RETURN
       ELSEIF(TYPE.EQ.1) THEN
         CALL GT_CONVERT(NFILE,TEMP,GAUGE,DIR)
         OPEN(4,FILE=NFILE)
         CALL DEFINE_ABCD(NDIM,NSYM,ABCD2,LENGTH)
         CALL XMAT(ABCD1,ABCD2,NDIM,NSYM)
       ELSE
         STOP 'INPUT ERROR'
       ENDIF
10    CONTINUE
      RETURN
      END
```

```
C*********************************************************************C
C                         PROGRAM      DFEBER                         C
C*********************************************************************C
C                                                                     C
C          This program will calculate the bit-error rate of a system C
C     transmitting at a user assigned baud rate over a subscriber    C
C     loop. The system impulse response and the noise impulse        C
C     responses are calculated, and the received signal is obtained by C
C     convolving the different impulse responses with randomly gene- C
C     rated numbers or symbols. This received signal will then pass  C
C     through an adaptive decision feedback equalizer; the detected  C
C     symbols and the ideal transmitted symbols are compared to      C
C     obtain bit-error rate and symbol error rate. The signal to     C
C     noise ratio at the input of the quantiser of the DFE is also   C
C     calculated.                                                     C
C                                                                     C
C     Input file = dfeber.in                                          C
C     Subroutine needed for execution = bereq.for                     C
C                                                                     C
C     Data required:                                                  C
C                   name = file with ABCD matrix of the loop          C
C                  name1 = output file name                           C
C                symrate = transmit symbol rate (baud/s)              C
C                  alpha = roll off factor for raised cosine filter   C
C                          ( 0 ≤ x ≤ 1.0)                             C
C                  noise = white noise spectral density               C
C                  knext = near-end crosstalk coupling constant       C
C                     ec = attenuation provided by echo canceller (dB) C
C                    dbm = transmit power level in dBm at interface   C
C                 numsym = number of symbols to be transmitted        C
C                percent = ratio to decide the length of precursors   C
C                          and postcursors for impulse response       C
C                          (suggested value < 5e-3)                   C
C                 n3back = number of dfe taps                         C
C           delta,delta1 = step size for adaption (trial and error)   C
C               fraction = ratio to decide if dfe has converged       C
C                          (suggested value < 5e-3)                   C
C        ix,iy,iz,idum = seeds to initialize random number generator C
C                          (integer)                                  C
C                                                                     C
C*********************************************************************C
      INTEGER NPOINT,NSYM,PPSYM,START,NEWPPSYM,SAMPT1,SAMPT2,SAMPT3
      INTEGER N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12
      INTEGER IX,IY,IZ,IDUM,NUMSYM,PREMAX,POSTMAX
      REAL PI,TPI,ALPHA,DF,SYMRATE,COGAIN,SUBGAIN
      REAL WSD,XTSD,SYMVAR,NOISE,KNEXT,EC,DBM,RLOAD
      REAL DELTA,DELTA1,FRACTION,PERCENT
      REAL IMP(2052),MULTI1(2052),MULTI2(2052)
      REAL COCHANFIR(-20:40),SUBCHANFIR(-20:40)
      REAL COECFIR(-20:40),SUBECFIR(-20:40)
      REAL COXTFIR(-40:40),SUBXTFIR(-40:40)
      REAL WNFIR(-40:40)
      CHARACTER*30 NAME,NAME1
      COMMON /BLK1/ NPOINT,NSYM,PPSYM
```

```
      COMMON /BLK2/ ALPHA,DF
      COMMON /BLK3/ PI,TPI
      COMMON /BLK4/ START,NEWPPSYM
      COMMON /GAIN/ COGAIN,SUBGAIN
      COMMON /RAND/ IX,IY,IZ,IDUM
      COMMON /DFEBLK/ DELTA,DELTA1,FRACTION,NUMSYM
C
C          READ INPUT DATA
C
      OPEN(1,FILE='DFEBER.IN')
       READ(1,*) NAME,NAME1
       READ(1,*) SYMRATE,ALPHA,NOISE,KNEXT,EC,DBM
       READ(1,*) PERCENT,DELTA,DELTA1,FRACTION
       READ(1,*) N3BACK,NUMSYM
       READ(1,*) IX,IY,IZ,IDUM
      CLOSE(1)
      OPEN(8,FILE=NAME)
C
C          INITIALIZATION
C
      NPOINT=2048
      PPSYM=16
      NSYM=128
      DF=SYMRATE/NSYM
      PI=3.1415926536
      TPI=2.0*PI
      START=513
      NEWPPSYM=2
      RLOAD=135.0
      SYMVAR=(3**2+(-3)**2+1**2+(-1)**2)/4
      WSD=SQRT(NOISE*NEWPPSYM*SYMRATE*RLOAD)
      XTSD=SQRT(SYMVAR*NEWPPSYM)
C
C          GENERATE A POSITIVE AND A NEGATIVE IMPULSE, AND TRANSFORM
C          INTO FREQUENCY DOMAIN.
C
      IMP(START)=PPSYM
      IMP(START+NPOINT/2)=-PPSYM
       CALL REALFT(IMP,NPOINT/2,1)
C
      DO 5 I=1,NPOINT
        MULTI1(I)=IMP(I)
5     MULTI2(I)=IMP(I)
C
C          PROVIDE PROPER SCALING AT THE LINE AND NETWORK INTERFACE
C
      CALL VSCALE(COGAIN,SUBGAIN,DBM,MULTI1,MULTI2)
C
C          DETERMINE THE COLOURED NOISE IMPULSE RESPONSE, SAMPLED
C          AND FED THE COEFFICIENTS INTO THE FIR FILTER. THE LENGTH
C          REQUIRED FOR PRECURSORS AND POSTCURSORS ARE CALCULATED.
C
      DO 10 I=1,NPOINT
10      MULTI1(I)=IMP(I)
```

```
C
      CALL WN(WSD,MULTI1)
      CALL SAMPPHASE(START,1,SAMPT1,MULTI1)
      CALL IMPLENGTH(SAMPT1,N3,N4,PERCENT,MULTI1,20,20)
       DO 15 I=-N3*2,N4*2
15       WNFIR(I)=MULTI1(SAMPT1+I*PPSYM/2)
C
      WRITE(6,*) 'COLOURED NOISE :  PRECURSOR = ',N3
      WRITE(6,*) '                 POSTCURSOR = ',N4
C
C          DETERMINE THE NEAR-END CROSSTALK IMPULSE RESPONSE, SAMPLE
C          AND FEED THE COEFFICIENTS INTO THE FIR FILTER. THE LENGTHS
C          REQUIRED FOR PRECURSORS AND POSTCURSORS ARE CALCULATED.
C
      DO 20 I=1,NPOINT
        MULTI1(I)=IMP(I)
20    MULTI2(I)=IMP(I)
      CALL XT(KNEXT,XTSD,MULTI1,MULTI2)
C
C          C.O. SIDE
C
      CALL SAMPPHASE(START,1,SAMPT1,MULTI1)
      CALL IMPLENGTH(SAMPT1,N5,N6,PERCENT,MULTI1,20,20)
C
C          SUBSCRIBER SIDE
C
      CALL SAMPPHASE(START,1,SAMPT2,MULTI2)
      CALL IMPLENGTH(SAMPT2,N7,N8,PERCENT,MULTI2,20,20)
C
C          LENGTH REQUIRED FOR PRECURSORS AND POSTCURSORS AT BOTH
C          ENDS ARE CHOSEN SUCH THAT THE LONGER ONE WILL BE USED.
C
       WRITE(6,*) 'NEXT NOISE:  CO PRECURSOR = ',N5,' POSTCURSOR = ',N6
       WRITE(6,*) '           : SUB PRECURSOR = ',N7,' POSTCURSOR = ',N8
      N5=MAX(N5,N7)
      N6=MAX(N6,N8)
       WRITE(6,*) ' FINAL LENGTH : PRECURSOR = ',N5,' POSTCURSOR = ',N6
C
      DO 25 I=-N5*2,N6*2
        COXTFIR(I)=MULTI1(SAMPT1+I*PPSYM/2)
25    SUBXTFIR(I)=MULTI2(SAMPT2+I*PPSYM/2)
C
C          DETERMINE THE ECHO IMPULSE RESPONSE, SAMPLED AT THE WORST
C          SAMPLING PHASE ( MAXIMIZE ECHO ). THE LENGTHS REQUIRED
C          FOR PRECURSORS AND POSTCURSORS ARE CALCULATED.
C
       OPEN(8,FILE=NAME)
C
      DO 30 I=1,NPOINT
        MULTI1(I)=IMP(I)
30    MULTI2(I)=IMP(I)
      CALL ECHO(EC,MULTI1,MULTI2)
C
C          C.O. SIDE
```

```
C
      CALL SAMPPHASE(START,1,SAMPT1,MULTI1)
      CALL IMPLENGTH(SAMPT1,N9,N10,PERCENT,MULTI1,20,40)
C
C          SUBSCRIBER SIDE
C
      CALL SAMPPHASE(START,1,SAMPT2,MULTI2)
      CALL IMPLENGTH(SAMPT2,N11,N12,PERCENT,MULTI2,20,40)
C
C          DETERMINE THE CHANNEL IMPULSE RESPONSE, SAMPLED AT THE
C          PHASE TO MINIMIZE FIRST PRECURSOR. THE LENGTHS REQUIRED
C          FOR PRECURSORS AND POSTCURSORS ARE CALCULATED.
C
      OPEN(8,FILE-NAME)
      CALL IMPULSE(IMP)
C
      CALL SAMPPHASE(START,2,SAMPT3,IMP)
      CALL IMPLENGTH(SAMPT3,N1,N2,PERCENT,IMP,20,40)
C
C          LENGTHS REQUIRED FOR PRECURSORS AND POSTCURSORS AT BOTH
C          ENDS FOR THE CHANNEL AND ECHO IMPULSE RESPONSES ARE CHOSEN
C          SUCH THAT THE LONGER ONE WILL BE USED.
C
      WRITE(6,*) 'ECHO NOISE: CO PRECURSOR - ',N9,' POSTCURSOR - ',N10
      WRITE(6,*) '           :SUB PRECURSOR - ',N11,' POSTCURSOR - ',N12
      WRITE(6,*) 'CHANNEL    : CO PRECURSOR - ',N1,' POSTCURSOR - ',N2
C
      N1-MAX(N1,N9,N11)
      N2-MAX(N2,N10,N12)
      WRITE(6,*) ' FINAL LENGTH : PRECURSOR - ',N1,' POSTCURSOR - ',N2
C
      DO 35 I--N1,N2
        COCHANFIR(I)-IMP(SAMPT3+I*PPSYM)*COGAIN
        SUBCHANFIR(I)-IMP(SAMPT3+I*PPSYM)*SUBGAIN
        COECFIR(I)-MULTI1(SAMPT1+I*PPSYM)
35    SUBECFIR(I)-MULTI2(SAMPT2+I*PPSYM)
C
C          EQUALIZED BY DFE
C
      CALL DFE(COCHANFIR,SUBCHANFIR,COECFIR,SUBECFIR,COXTFIR,
     +      SUBXTFIR,WNFIR,N1,N2,N3,N4,N5,N6,N3BACK,NAME,NAME1)
C
      STOP
      END
```

```
C***********************************************************************C
C                                                                       C
C        This subroutine uses the LMS algorithm to adaptively          C
C    equalize the signal by a decision feedback equalizer. The          C
C    dfe has n3back taps, and a one tap linear transversal filter       C
C    at the front which acts as a automatic gain control device.        C
C    The dfe is first set in the training mode. If the equalizer        C
C    has converged, it is automatically set in the adaptive mode        C
C    and BER calculation begins. The signal and noise are               C
C    generated from their impulse responses by convolving with          C
C    randomly generated numbers or symbols. The quantiser signal        C
C    to noise ratio is also calculated.                                 C
C                                                                       C
C     cochanfir - co -> sub channel impulse response                    C
C    subchanfir - sub -> co channel impulse response                    C
C       coecfir - co side echo impulse response                         C
C      subecfir - sub side echo impulse response                        C
C       coxtfir - co side near-end crosstalk impulse response           C
C      subxtfir - sub side near-end crosstalk impulse response          C
C         wnfir - coloured zero-mean gaussian noise impulse response    C
C                 (same for both ends)                                  C
C         n1/n2 - number of symbol periods required to represent the    C
C                 pre/postcursor of channel and echo impulse response   C
C         n3/n4 - number of symbol periods required to represent the    C
C                 pre/postcursor of coloured noise impulse response     C
C         n5/n6 - number of symbol periods required to represent the    C
C                 pre/postcursor of near-end crosstalk impulse          C
C                 response                                              C
C        n3back - number of dfe taps ( maximum=50, can increase by      C
C                 changing the code )                                   C
C          name - loop file name                                        C
C         name1 - output file name ( unit 1 is used for writing )       C
C                                                                       C
C***********************************************************************C
      SUBROUTINE DFE(COCHANFIR,SUBCHANFIR,COECFIR,SUBECFIR,COXTFIR,
     +           SUBXTFIR,WNFIR,N1,N2,N3,N4,N5,N6,N3BACK,NAME,NAME1)
      INTEGER N1,N2,N3,N4,N5,N6,N3BACK,PREMAX,POSTMAX
      INTEGER COFLAG,SUBFLAG,SW,COUNT,INC
      INTEGER TRAINSYM,NUMSYM,SYMA,SYMB
      INTEGER SYMGEN,DECISION,IX,IY,IZ,IDUM
      INTEGER COBER(2),SUBBER(2)
      INTEGER COISYM(4),SUBISYM(4)
      INTEGER COREG(-20:40),SUBREG(-20:40)
      INTEGER COA(-50:-1),SUBA(-50:-1)
      REAL COGAIN,SUBGAIN,GCO,GSUB,COERROR,SUBERROR
      REAL COTFTAP,SUBTFTAP,CODFEYO,SDFEYO,COTFYO,SUBTFYO
      REAL COOLDAVG,SUBOLDAVG,CONEWAVG,SUBNEWAVG
      REAL DELTA,DELTA1,FRACTION,GASDEV,RAN1
      REAL COWNOISE,SUBWNOISE,COECNOISE,SUBECNOISE,COXTNOISE,SUBXTNOISE
      REAL CONOISEP,SUBNOISEP,COSIGP,SUBSIGP
      REAL COOUTSYM,SUBOUTSYM
      REAL CODFETAP(50),SUBDFETAP(50)
      REAL WNREG1(-40:40),WNREG2(-40:40)
      REAL XTREG1(-40:40),XTREG2(-40:40)
```

```
      REAL COCHANFIR(-20:40),SUBCHANFIR(-20:40)
      REAL COECFIR(-20:40),SUBECFIR(-20:40)
      REAL COXTFIR(-40:40),SUBXTFIR(-40:40)
      REAL WNFIR(-40:40)
      COMMON /GAIN/ COGAIN,SUBGAIN
      COMMON /RAND/ IX,IY,IZ,IDUM
      COMMON /DFEBLK/ DELTA,DELTA1,FRACTION,NUMSYM
      CHARACTER*30 NAME,NAME1
C
C          INITIALIZATION
C
       SW=1
       INC=500
       COUNT=INC
       GCO=1.0/(COCHANFIR(0))
       GSUB=1.0/(SUBCHANFIR(0))
       COTFTAP=1.0
       SUBTFTAP=1.0
       COOLDAVG=0.0
       SUBOLDAVG=0.0
       CONEWAVG=0.0
       SUBNEWAVG=0.0
       COFLAG=0
       SUBFLAG=0
       COBER(1)=0
       COBER(2)=0
       SUBBER(1)=0
       SUBBER(2)=0
C
C          FILLED THE FIR DATA REGISTERS AND DFE DATA REGISTERS
C
       DO 5 I=-20,40
         COREG(I)=SYMGEN(COISYM)
5      SUBREG(I)=SYMGEN(SUBISYM)
       DO 10 I=-40,40
         WNREG1(I)=GASDEV(IDUM)
         WNREG2(I)=GASDEV(IDUM)
         XTREG1(I)=GASDEV(IDUM)
10     XTREG2(I)=GASDEV(IDUM)
       DO 15 I=1,50
         COA(-I)=0.0
15     SUBA(-I)=0.0
C
C          DFE EQUALIZER: SW=1 (TRAINING MODE)
C                          =2 (ADAPTIVE MODE)
C
       DO 20 II=1,NUMSYM
C
         COECNOISE=0.0
         SUBECNOISE=0.0
         COWNOISE=0.0
         SUBWNOISE=0.0
         COXTNOISE=0.0
         SUBXTNOISE=0.0
```

```
         COOUTSYM-0.0
         SUBOUTSYM-0.0
C
C          SIGNAL AND NOISE AT THE SAMPLING INSTANT IS CALCULATED
C
        DO 25 J--N1,N2
          COECNOISE-COECNOISE+COREG(J)*COECFIR(J)
          SUBECNOISE-SUBECNOISE+SUBREG(J)*SUBECFIR(J)
          COOUTSYM-COOUTSYM+SUBREG(J)*SUBCHANFIR(J)
25      SUBOUTSYM-SUBOUTSYM+COREG(J)*COCHANFIR(J)
C
        DO 30 J--N3*2,N4*2
          COWNOISE-COWNOISE+WNFIR(J)*WNREG1(J)
30      SUBWNOISE-SUBWNOISE+WNFIR(J)*WNREG2(J)
C
        DO 35 J--N5*2,N6*2
          COXTNOISE-COXTNOISE+COXTFIR(J)*XTREG1(J)
35      SUBXTNOISE-SUBXTNOISE+SUBXTFIR(J)*XTREG2(J)
C
C          THE RECEIVED SIGNAL (SIGNAL+NOISE) AT THE CENTRAL OFFICE
C          AND THE SUBSCRIBERSIDE IS AMPLIFIED
C
        COOUTSYM-(COOUTSYM+COWNOISE+COXTNOISE+COECNOISE)*GSUB
        SUBOUTSYM-(SUBOUTSYM+SUBWNOISE+SUBXTNOISE+SUBECNOISE)*GCO
C
C          THE RECEIVED SIGNAL ARE MULTIPLIED BY A ONE TAP ADAPTIVE
C          LINEAR TRANSVERSAL FILTER. THE FILTER IS USED AS A
C          AUTOMATIC GAIN CONTROL DEVICE.
C
        COTFYO-COTFTAP*COOUTSYM
        SUBTFYO-SUBTFTAP*SUBOUTSYM
C
C          FEEDBACK FILTER (N3BACK TAPS) OUTPUT
C
        CODFEYO-0.0
        SDFEYO-0.0
        DO 40 I-1,N3BACK
          CODFEYO-CODFEYO+CODFETAP(I)*COA(-I)
40      SDFEYO-SDFEYO+SUBDFETAP(I)*SUBA(-I)
C
C          SW - 1 : THE EQUALIZED SIGNAL IS COMPARED TO THE IDEAL
C                   TRANSMITTED SYMBOL. THE DIFFERENCE IS USED TO
C                   UPDATE THE TAPS.
C
C             - 2 : THE EQUALIZED SIGNAL IS PASSED TO A SLICER.
C                   THE DIFFERENCE BETWEEN THE DETECTED SYMBOL
C                   AND THE RECEIVED SIGNAL IS USED TO UPDATE
C                   THE TAPS. ANY SYMBOL ERROR OCCURED WILL BE
C                   COUNTED.
C
        IF(SW.EQ.1) THEN
          COERROR-SUBREG(0)-(COTFYO-CODFEYO)
          SUBERROR-COREG(0)-(SUBTFYO-SDFEYO)
        ELSE
```

```
      SYMA=DECISION(COTFY0-CODFEY0)
      SYMB=DECISION(SUBTFY0-SDFEY0)
      COERROR=SYMA-(COTFY0-CODFEY0)
      SUBERROR=SYMB-(SUBTFY0-SDFEY0)
C
C          SIGNAL AND NOISE POWER ARE CALCULATED
C
      CONOISEP=CONOISEP+(SUBREG(0)-(COTFY0-CODFEY0))**2
      SUBNOISEP=SUBNOISEP+(COREG(0)-(SUBTFY0-SDFEY0))**2
      COSIGP=COSIGP+SUBREG(0)**2
      SUBSIGP=SUBSIGP+COREG(0)**2
        IF(SUBREG(0).NE.SYMA) THEN
          CALL BITSYMERROR(SUBREG(0)-SYMA,COBER)
        ENDIF
        IF(COREG(0).NE.SYMB) THEN
          CALL BITSYMERROR(COREG(0)-SYMB,SUBBER)
        ENDIF
      ENDIF
C
C          UPDATE TF TAP VALUE
C
      COTFTAP=COTFTAP+DELTA1*COERROR*COOUTSYM
      SUBTFTAP=SUBTFTAP+DELTA1*SUBERROR*SUBOUTSYM
C
C          UPDATE DFE TAP VALUES
C
      DO 45 I=1,N3BACK
        CODFETAP(I)=CODFETAP(I)-DELTA*COERROR*COA(-I)
45    SUBDFETAP(I)=SUBDFETAP(I)-DELTA*SUBERROR*SUBA(-I)
C
C          GENERATE NEW SYMBOLS AND SHIFT CONTENTS IN FIR
C          AND DFE DATA REGISTERS BY T (SYMBOL PERIOD)
C
      DO 50 K=1,2
        DO 55 J=-N4*2,-N3*2+1,-1
          WNREG1(J)=WNREG1(J-1)
55      WNREG2(J)=WNREG2(J-1)
        WNREG1(-N3*2)=GASDEV(IDUM)
50    WNREG2(-N3*2)=GASDEV(IDUM)
C
      DO 60 K=1,2
        DO 65 J=-N6*2,-N5*2+1,-1
          XTREG1(J)=XTREG1(J-1)
65      XTREG2(J)=XTREG2(J-1)
        XTREG1(-N5*2)=GASDEV(IDUM)
60    XTREG2(-N5*2)=GASDEV(IDUM)
C
      DO 70 I=-N3BACK,-2
        SUBA(I)=SUBA(I+1)
70    COA(I)=COA(I+1)
C
C          SW = 1: IDEAL TRANSMITTED SYMBOL WILL BE FED INTO
C                  DFE DATA REGISTERS.
C
```

```
C              2: DETECTED SYMBOLS FROM SLICER WILL BE FED
C                 INTO DFE DATA REGISTERS.
C
      IF(SW.EQ.1) THEN
        COA(-1)=SUBREG(0)
        SUBA(-1)=COREG(0)
      ELSE
        COA(-1)=SYMA
        SUBA(-1)=SYMB
      ENDIF
C
      DO 75 J=N2,-N1+1,-1
        SUBREG(J)=SUBREG(J-1)
75    COREG(J)=COREG(J-1)
      SUBREG(-N1)=SYMGEN(SUBISYM)
      COREG(-N1)=SYMGEN(COISYM)
C
C         DETERMINE WHEN TO SWITCH DFE FROM TRAINING MODE INTO
C         ADAPTIVE MODE. THE TAP VALUE OF THE LINEAR TRANSVERSAL
C         FILTER IS AVERAGED OVER 500 SYMBOLS. THIS AVERAGE TAP
C         VALUE IS COMPARED TO THE PREVIOUS AVERAGE TAP VALUE.
C         IF THE TWO VALUES ARE WITHIN CERTAIN ASSIGNED PERCENTAGE,
C         THE DFE IS AUTOMATICALLY SWITCHED INTO ADAPTIVE MODE AND
C         BEGINS TO CALCULATE THE BIT-ERROR RATE. COFLAG AND SUBFLAG
C         WILL BE SET TO 1 IF DFE'S AT BOTH ENDS ARE TRAINED AND
C         CONVERGED.
C
        IF(SW.EQ.1) THEN
          IF(COUNT.GT.II) THEN
            CONEWAVG=CONEWAVG+COTFTAP
            SUBNEWAVG=SUBNEWAVG+SUBTFTAP
          ELSEIF(COUNT.EQ.II) THEN
            COUNT=COUNT+INC
             IF(ABS(1.0-(COOLDAVG/CONEWAVG)).LE.FRACTION) COFLAG=1
             IF(ABS(1.0-(SUBOLDAVG/SUBNEWAVG)).LE.FRACTION) SUBFLAG=1
            COOLDAVG=CONEWAVG
            SUBOLDAVG=SUBNEWAVG
            CONEWAVG=0.0
            SUBNEWAVG=0.0
             IF(COFLAG.EQ.1.AND.SUBFLAG.EQ.1) THEN
               TRAINSYM=II
                DO 80 I=1,4
                  COISYM(I)=0
80              SUBISYM(I)=0
               SW=2
             ENDIF
          ENDIF
        ENDIF
20    CONTINUE
      NUMSYM=NUMSYM-TRAINSYM
```

```
C
C        OUTPUT RESULTS
C
      OPEN(1,FILE=NAME1)
       IF(SW.EQ.1) THEN
         WRITE(1,*) 'DFE HAS NOT CONVERGED'
         STOP
       ENDIF
       IF(SW.EQ.2) WRITE(1,*) 'C.O. AND SUB DFE HAVE CONVERGED'
      WRITE(1,*) NAME
      WRITE(1,*) 'TRAINING PERIOD (SYM) = ',TRAINSYM
      WRITE(1,*) 'TRANSMITTED SYMBOLS = ',NUMSYM
      WRITE(1,*)
      WRITE(1,*) 'CO :'
      WRITE(1,*)
      WRITE(1,*) ' SYMBOL  3 = ',SUBISYM(1),' SYMBOL  1 = ',SUBISYM(2)
      WRITE(1,*) ' SYMBOL -1 = ',SUBISYM(3),' SYMBOL -3 = ',SUBISYM(4)
      WRITE(1,*)
      WRITE(1,*) ' NUMBER OF BIT ERROR = ', COBER(1)
      WRITE(1,*) ' NUMBER OF SYM ERROR = ', COBER(2)
      WRITE(1,*) ' QSNR (dB)           = ', 10*LOG10(COSIGP/CONOISEP)
      WRITE(1,*) '      BIT ERROR RATE = ', COBER(1)/(NUMSYM*2.0)
      WRITE(1,*) '      SYM ERROR RATE = ', COBER(2)*1.0/NUMSYM
      WRITE(1,*)
      WRITE(1,*) 'SUB :'
      WRITE(1,*)
      WRITE(1,*) ' SYMBOL  3 = ',COISYM(1),' SYMBOL  1 = ',COISYM(2)
      WRITE(1,*) ' SYMBOL -1 = ',COISYM(3),' SYMBOL -3 = ',COISYM(4)
      WRITE(1,*)
      WRITE(1,*) ' NUMBER OF BIT ERROR = ', SUBBER(1)
      WRITE(1,*) ' NUMBER OF SYM ERROR = ', SUBBER(2)
      WRITE(1,*) ' QSNR (dB)           = ', 10*LOG10(SUBSIGP/SUBNOISEP)
      WRITE(1,*) '      BIT ERORR RATE = ', SUBBER(1)/(NUMSYM*2.0)
      WRITE(1,*) '      SYM ERORR RATE = ', SUBBER(2)*1.0/NUMSYM
      CLOSE(1)
      RETURN
      END
```

```
C*********************************************************************C
C                       PROGRAM      DFEEYE                           C
C*********************************************************************C
C                                                                     C
C          This program will plot the eye diagram of a system trans-  C
C     mitting at a user assigned baud rate over a subscriber loop.    C
C     The system impulse response and the noise impulse responses     C
C     are calculated, and the received signal is obtained by con-     C
C     volving the different impulse responses with randomly generated C
C     numbers or symbols. This received signal is equalized by        C
C     an adaptive decision feedback equalizer, and then the eye       C
C     diagram is plotted.                                             C
C                                                                     C
C     Input file = dfeeye.in                                          C
C    Output file = eyestat.dat, coeye.dat, subeye.dat                 C
C                                                                     C
C     Data required:                                                  C
C                  name = file with ABCD matrix of the loop           C
C                 name1 = output file name                            C
C               symrate = transmit symbol rate (baud/s)               C
C                 alpha = roll off factor for raised cosine filter    C
C                         ( 0 < x < 1.0)                              C
C                 noise = white noise spectral density                C
C                 knext = near-end crosstalk coupling constant        C
C                    ec = attenuation provided by echo canceller (dB) C
C                   dbm = transmit power level at interface           C
C                numsym = number of symbols to be transmitted         C
C               percent = ratio to decide the length of precursors    C
C                         and postcursors for impulse response        C
C                         (suggested value < 5e-3)                    C
C                n3back = number of dfe taps                          C
C          delta,delta1 = step size for adaption                      C
C              fraction = ratio to decide if dfe has converged        C
C                         (suggested value < 5e-3)                    C
C        ix,iy,iz,idum = seeds to initialize random number generator  C
C                         (integer)                                   C
C                                                                     C
C*********************************************************************C
      INTEGER NPOINT,NSYM,PPSYM,START,NEWPPSYM,SAMPT1,SAMPT2,SAMPT3
      INTEGER N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12
      INTEGER IX,IY,IZ,IDUM,NUMSYM,PREMAX,POSTMAX
      REAL PI,TPI,ALPHA,DF,SYMRATE,COGAIN,SUBGAIN
      REAL WSD,XTSD,SYMVAR,NOISE,KNEXT,EC,DBM,RLOAD
      REAL DELTA,DELTA1,FRACTION,PERCENT
      REAL IMP(2052),MULTI1(2052),MULTI2(2052)
      REAL COCHANFIR(-160:320),SUBCHANFIR(-160:320)
      REAL COECFIR(-160:320),SUBECFIR(-160:320)
      REAL COXTFIR(-160:160),SUBXTFIR(-160:160)
      REAL WNFIR(-160:160)
      CHARACTER*30 NAME
      COMMON /BLK1/ NPOINT,NSYM,PPSYM
      COMMON /BLK2/ ALPHA,DF
      COMMON /BLK3/ PI,TPI
      COMMON /BLK4/ START,NEWPPSYM
```

```
      COMMON /GAIN/ COGAIN,SUBGAIN
      COMMON /RAND/ IX,IY,IZ,IDUM
      COMMON /DFEBLK/ DELTA,DELTA1,FRACTION,NUMSYM
C
C          READ INPUT DATA
C
      OPEN(1,FILE='DFEEYE.IN')
       READ(1,*) NAME
       READ(1,*) SYMRATE,ALPHA,NOISE,KNEXT,EC,DBM
       READ(1,*) PERCENT,DELTA,DELTA1,FRACTION
       READ(1,*) N3BACK,NUMSYM
       READ(1,*) IX,IY,IZ,IDUM
      CLOSE(1)
      OPEN(8,FILE=NAME)
C
C          INITIALIZATION
C
      NPOINT=2048
      PPSYM=16
      NSYM=128
      DF=SYMRATE/NSYM
      PI=3.1415926536
      TPI=2.0*PI
      START=513
      NEWPPSYM=8
      RLOAD=135.0
      SYMVAR=(3**2+(-3)**2+1[illegible]**2)/4
      WSD=SQRT(NOISE*NEWPPSYM*SYMRATE*RLOAD)
      XTSD=SQRT(SYMVAR*NEWPPSYM)
C
C          GENERATE A POSITIVE AND A NEGATIVE IMPULSE, AND TRANSFORM
C          INTO FREQUENCY DOMAIN.
C
      IMP(START)=PPSYM
      IMP(START+NPOINT/2)=-PPSYM
       CALL REALFT(IMP,NPOINT/2,1)
C
      DO 5 I=1,NPOINT
        MULTI1(I)=IMP(I)
5     MULTI2(I)=IMP(I)
C
C          PROVIDE PROPER SCALING AT THE LINE AND NETWORK INTERFACE
C
      CALL VSCALE(COGAIN,SUBGAIN,DBM,MULTI1,MULTI2)
C
C          DETERMINE THE COLOURED NOISE IMPULSE RESPONSE, SAMPLE
C          AND FEED THE COEFFICIENTS INTO THE FIR FILTER. THE LENGTHS
C          REQUIRED FOR PRECURSORS AND POSTCURSORS ARE CALCULATED.
C
      DO 10 I=1,NPOINT
10      MULTI1(I)=IMP(I)
C
      CALL WN(WSD,MULTI1)
      CALL SAMPPHASE(START,1,SAMPT1,MULTI1)
```

```
      CALL IMPLENGTH(SAMPT1,N3,N4,PERCENT,MULTI1,20,20)
       DO 15 I=-N3*NEWPPSYM,N4*NEWPPSYM
15       WNFIR(I)=MULTI1(SAMPT1+I*PPSYM/NEWPPSYM)
C
      WRITE(6,*) 'COLOURED NOISE :  PRECURSOR = ',N3
      WRITE(6,*) '                 POSTCURSOR = ',N4
C
C          DETERMINE THE NEAR-END CROSSTALK IMPULSE RESPONSE, SAMPLE
C          AND FEED THE COEFFICIENTS INTO THE FIR FILTER. THE LENGTHS
C          REQUIRED FOR PRECURSORS AND POSTCURSORS ARE CALCULATED.
C
      DO 20 I=1,NPOINT
        MULTI1(I)=IMP(I)
20    MULTI2(I)=IMP(I)
      CALL XT(KNEXT,XTSD,MULTI1,MULTI2)
C
C          C.O. SIDE
C
      CALL SAMPPHASE(START,1,SAMPT1,MULTI1)
      CALL IMPLENGTH(SAMPT1,N5,N6,PERCENT,MULTI1,20,20)
C
C          SUBSCRIBER SIDE
C
      CALL SAMPPHASE(START,1,SAMPT2,MULTI2)
      CALL IMPLENGTH(SAMPT2,N7,N8,PERCENT,MULTI2,20,20)
C
C          LENGTH REQUIRED FOR PRECURSORS AND POSTCURSORS AT BOTH
C          ENDS ARE CHOSEN SUCH THAT THE LONGER ONE WILL BE USED.
C
       WRITE(6,*) 'NEXT NOISE:  CO PRECURSOR = ',N5,' POSTCURSOR = ',N6
       WRITE(6,*) '           : SUB PRECURSOR = ',N7,' POSTCURSOR = ',N8
      N5=MAX(N5,N7)
      N6=MAX(N6,N8)
       WRITE(6,*) ' FINAL LENGTH : PRECURSOR = ',N5,' POSTCURSOR = ',N6
C
      DO 25 I=-N5*NEWPPSYM,N6*NEWPPSYM
        COXTFIR(I)=MULTI1(SAMPT1+I*PPSYM/NEWPPSYM)
25    SUBXTFIR(I)=MULTI2(SAMPT2+I*PPSYM/NEWPPSYM)
C
C          DETERMINE THE ECHO IMPULSE RESPONSE, SAMPLE AT THE WORST
C          SAMPLING PHASE ( MAXIMIZE ECHO ). THE LENGTHS REQUIRED
C          FOR PRECURSORS AND POSTCURSORS ARE CALCULATED.
C
       OPEN(8,FILE=NAME)
C
      DO 30 I=1,NPOINT
        MULTI1(I)=IMP(I)
30    MULTI2(I)=IMP(I)
      CALL ECHO(EC,MULTI1,MULTI2)
C
C          C.O. SIDE
C
      CALL SAMPPHASE(START,1,SAMPT1,MULTI1)
      CALL IMPLENGTH(SAMPT1,N9,N10,PERCENT,MULTI1,20,40)
```

```
C
C          SUBSCRIBER SIDE
C
      CALL SAMPPHASE(START,1,SAMPT2,MULTI2)
      CALL IMPLENGTH(SAMPT2,N11,N12,PERCENT,MULTI2,20,40)
C
C          DETERMINE THE CHANNEL IMPULSE RESPONSE, SAMPLE AT THE
C          PHASE TO MINIMIZE FIRST PRECURSOR. THE LENGTHS REQUIRED
C          FOR PRECURSORS AND POSTCURSORS ARE CALCULATED.
C
      OPEN(8,FILE=NAME)
      CALL IMPULSE(IMP)
C
      CALL SAMPPHASE(START,2,SAMPT3,IMP)
      CALL IMPLENGTH(SAMPT3,N1,N2,PERCENT,IMP,20,40)
C
C          LENGTH REQUIRED FOR PRECURSORS AND POSTCURSORS AT BOTH
C          ENDS FOR THE CHANNEL AND ECHO IMPULSE RESPONSES ARE CHOSEN
C          SUCH THAT THE LONGER ONE WILL BE USED.
C
      WRITE(6,*) 'ECHO NOISE: CO PRECURSOR = ',N9,' POSTCURSOR = ',N10
      WRITE(6,*) '           :SUB PRECURSOR = ',N11,' POSTCURSOR = ',N12
      WRITE(6,*) 'CHANNEL    : CO PRECURSOR = ',N1,' POSTCURSOR = ',N2
C
      N1=MAX(N1,N9,N11)
      N2=MAX(N2,N10,N12)
      WRITE(6,*) ' FINAL LENGTH : PRECURSOR = ',N1,' POSTCURSOR = ',N2
C
      DO 35 I=-N1*NEWPPSYM,N2*NEWPPSYM
        COCHANFIR(I)=IMP(SAMPT3+I*PPSYM/NEWPPSYM)*COGAIN
        SUBCHANFIR(I)=IMP(SAMPT3+I*PPSYM/NEWPPSYM)*SUBGAIN
        COECFIR(I)=MULTI1(SAMPT1+I*PPSYM/NEWPPSYM)
35    SUBECFIR(I)=MULTI2(SAMPT2+I*PPSYM/NEWPPSYM)
C
C          EQUALIZED BY DFE
C
       CALL DFEEYE(COCHANFIR,SUBCHANFIR,COECFIR,SUBECFIR,COXTFIR,
     +              SUBXTFIR,WNFIR,N1,N2,N3,N4,N5,N6,N3BACK,NAME)
C
       STOP
       END
C**********************************************************************C
C                                                                      C
C          This subroutine uses the LMS algorithm to adaptively        C
C     equalize the signal by a decision feedback equalizer. This       C
C     dfe has n3back taps, and a one tap linear transversal filter     C
C     at the front which acts as a automatic gain control device.      C
C     The dfe is first set in the training mode. If the equalizer      C
C     has converged, it is automatically set in the adaptive mode      C
C     and plotting of the eye diagram begins. The signal and noise are C
C     generated from their impulse responses by convolving with        C
C     randomly generated numbers or symbols.                           C
C                                                                      C
C      cochanfir = co => sub channel impulse response                  C
```

```
C     subchanfir - sub -> co channel impulse response                  C
C        coecfir - co side echo impulse response                       C
C       subecfir - sub side echo impulse response                      C
C        coxtfir - co side near-end crosstalk impulse response         C
C       subxtfir - sub side near-end crosstalk impulse response        C
C          wnfir - coloured zero-mean gaussian noise impulse response  C
C                  (same for both ends)                                C
C          n1/n2 - number of symbol periods required to represent the  C
C                  pre/postcursor of channel and echo impulse response C
C          n3/n4 - number of symbol periods required to represent the  C
C                  pre/postcursor of coloured noise impulse response   C
C          n5/n6 - number of symbol periods required to represent the  C
C                  pre/postcursor of near-end crosstalk impulse        C
C                  response                                            C
C         n3back - number of dfe taps ( maximum=50, can increase by    C
C                  changing the code )                                 C
C           name -  loop file name                                     C
C                                                                      C
C**********************************************************************C
      SUBROUTINE DFEEYE(COCHANFIR,SUBCHANFIR,COECFIR,SUBECFIR,COXTFIR,
     +                   SUBXTFIR,WNFIR,N1,N2,N3,N4,N5,N6,N3BACK,NAME)
      INTEGER N1,N2,N3,N4,N5,N6,N3BACK
      INTEGER COFLAG,SUBFLAG,COUNT,INC
      INTEGER NUMSYM,SYMA,SYMB
      INTEGER SYMGEN,DECISION,IX,IY,IZ,IDUM
      INTEGER NPOINT,START,NEWPPSYM,INDEX,INDEX1
      INTEGER COISYM(4),SUBISYM(4)
      INTEGER*1 CODESIRE(256),SUBDESIRE(256)
      INTEGER*1 COREG(-20:40),SUBREG(-20:40)
      INTEGER*1 COA(-50:-1),SUBA(-50:-1)
      REAL COGAIN,SUBGAIN,GCO,GSUB,COERROR,SUBERROR
      REAL COTFTAP,SUBTFTAP,CODFEY0,SDFEY0,COTFY0,SUBTFY0
      REAL COOLDAVG,SUBOLDAVG,CONEWAVG,SUBNEWAVG
      REAL DELTA,DELTA1,FRACTION,GASDEV,RAN1
      REAL COWNOISE,SUBWNOISE,COECNOISE,SUBECNOISE,COXTNOISE,SUBXTNOISE
      REAL COOUTSYM,SUBOUTSYM
      REAL COEYE(2048),SUBEYE(2048)
      REAL CODFETAP(50),SUBDFETAP(50)
      REAL WNREG1(-160:160),WNREG2(-160:160)
      REAL XTREG1(-160:160),XTREG2(-160:160)
      REAL COCHANFIR(-160:320),SUBCHANFIR(-160:320)
      REAL COECFIR(-160:320),SUBECFIR(-160:320)
      REAL COXTFIR(-160:320),SUBXTFIR(-160:320)
      REAL WNFIR(-160:160)
      REAL COSYM(256),SUBSYM(256)
      COMMON /BLK4/ START,NEWPPSYM
      COMMON /GAIN/ COGAIN,SUBGAIN
      COMMON /RAND/ IX,IY,IZ,IDUM
      COMMON /DFEBLK/ DELTA,DELTA1,FRACTION,NUMSYM
      CHARACTER*30 NAME
C
C          INITIALIZATION
C
       NPOINT=2048
```

```
      INC=500
      COUNT=INC
      GCO=1.0/(COCHANFIR(0))
      GSUB=1.0/(SUBCHANFIR(0))
      COTFTAP=1.0
      SUBTFTAP=1.0
      COOLDAVG=0.0
      SUBOLDAVG=0.0
      CONEWAVG=0.0
      SUBNEWAVG=0.0
      COFLAG=0
      SUBFLAG=0
C
C        FILLED THE FIR DATA REGISTERS AND DFE DATA REGISTERS
C
      DO 5 I=-20,40
        COREG(I)=SYMGEN(COISYM)
5     SUBREG(I)=SYMGEN(SUBISYM)
      DO 10 I=-160,160
        WNREG1(I)=GASDEV(IDUM)
        WNREG2(I)=GASDEV(IDUM)
        XTREG1(I)=GASDEV(IDUM)
10    XTREG2(I)=GASDEV(IDUM)
      DO 15 I=1,50
        COA(-I)=0.0
15    SUBA(-I)=0.0
C
C         DFE EQUALIZER
C
      DO 20 II=1,NUMSYM
C
        COECNOISE=0.0
        SUBECNOISE=0.0
        COWNOISE=0.0
        SUBWNOISE=0.0
        COXTNOISE=0.0
        SUBXTNOISE=0.0
        COOUTSYM=0.0
        SUBOUTSYM=0.0
C
C         SIGNAL AND NOISE AT THE SAMPLING INSTANT IS CALCULATED
C
      DO 25 J=-N1,N2
        INDEX=J*NEWPPSYM
        COECNOISE=COECNOISE+COREG(J)*COECFIR(INDEX)
        SUBECNOISE=SUBECNOISE+SUBREG(J)*SUBECFIR(INDEX)
        COOUTSYM=COOUTSYM+SUBREG(J)*SUBCHANFIR(INDEX)
25    SUBOUTSYM=SUBOUTSYM+COREG(J)*COCHANFIR(INDEX)
C
      DO 30 J=-N3*NEWPPSYM,N4*NEWPPSYM,NEWPPSYM/2
        COWNOISE=COWNOISE+WNFIR(J)*WNREG1(J)
30    SUBWNOISE=SUBWNOISE+WNFIR(J)*WNREG2(J)
C
      DO 35 J=-N5*NEWPPSYM,N6*NEWPPSYM,NEWPPSYM/2
```

```
          COXTNOISE=COXTNOISE+COXTFIR(J)*XTREG1(J)
35        SUBXTNOISE=SUBXTNOISE+SUBXTFIR(J)*XTREG2(J)
C
C            THE RECEIVED SIGNAL (SIGNAL+NOISE) AT THE CENTRAL OFFICE
C            AND THE SUBSCRIBERSIDE IS AMPLIFIED
C
        COOUTSYM=(COOUTSYM+COWNOISE+COXTNOISE+COECNOISE)*GSUB
        SUBOUTSYM=(SUBOUTSYM+SUBWNOISE+SUBXTNOISE+SUBECNOISE)*GCO
C
C            THE RECEIVED SIGNAL IS MULTIPLIED BY A ONE TAP ADAPTIVE
C            LINEAR TRANSVERSAL FILTER. THE FILTER IS USED AS A
C            AUTOMATIC GAIN CONTROL DEVICE.
C
        COTFY0=COTFTAP*COOUTSYM
        SUBTFY0=SUBTFTAP*SUBOUTSYM
C
C            FEEDBACK FILTER (N3BACK TAPS) OUTPUT
C
        CODFEY0=0.0
        SDFEY0=0.0
        DO 40 I=1,N3BACK
          CODFEY0=CODFEY0+CODFETAP(I)*COA(-I)
40      SDFEY0=SDFEY0+SUBDFETAP(I)*SUBA(-I)
C
C            THE EQUALIZED SIGNAL IS COMPARED TO THE IDEAL
C            TRANSMITTED SYMBOL. THE DIFFERENCE IS USED TO
C            UPDATE THE TAPS.
C
          COERROR=SUBREG(0)-(COTFY0-CODFEY0)
          SUBERROR=COREG(0)-(SUBTFY0-SDFEY0)
C
C            UPDATE TF TAP VALUE
C
        COTFTAP=COTFTAP+DELTA1*COERROR*COOUTSYM
        SUBTFTAP=SUBTFTAP+DELTA1*SUBERROR*SUBOUTSYM
C
C            UPDATE DFE TAP VALUES
C
        DO 45 I=1,N3BACK
          CODFETAP(I)=CODFETAP(I)-DELTA*COERROR*COA(-I)
45      SUBDFETAP(I)=SUBDFETAP(I)-DELTA*SUBERROR*SUBA(-I)
C
C            GENERATE NEW SYMBOLS AND SHIFT CONTENTS IN FIR
C            AND DFE DATA REGISTERS BY T (SYMBOL PERIOD)
C
        DO 50 K=1,2
          DO 55 J=N4*NEWPPSYM,-N3*NEWPPSYM+NEWPPSYM/2,-NEWPPSYM/2
            WNREG1(J)=WNREG1(J-NEWPPSYM/2)
55        WNREG2(J)=WNREG2(J-NEWPPSYM/2)
          WNREG1(-N3*NEWPPSYM)=GASDEV(IDUM)
50      WNREG2(-N3*NEWPPSYM)=GASDEV(IDUM)
C
        DO 60 K=1,2
          DO 65 J=N6*NEWPPSYM,-N5*NEWPPSYM+NEWPPSYM/2,-NEWPPSYM/2
```

```
          XTREG1(J)=XTREG1(J-NEWPPSYM/2)
65       XTREG2(J)=XTREG2(J-NEWPPSYM/2)
         XTREG1(-N5*NEWPPSYM)=GASDEV(IDUM)
60     XTREG2(-N5*NEWPPSYM)=GASDEV(IDUM)
C
       DO 70 I=-N3BACK,-2
         SUBA(I)=SUBA(I+1)
70     COA(I)=COA(I+1)
C
C          IDEAL TRANSMITTED SYMBOL WILL BE FED INTO
C          DFE DATA REGISTERS.
C
         COA(-1)=SUBREG(0)
         SUBA(-1)=COREG(0)
C
       DO 75 J=N2,-N1+1,-1
         SUBREG(J)=SUBREG(J-1)
75     COREG(J)=COREG(J-1)
       SUBREG(-N1)=SYMGEN(SUBISYM)
       COREG(-N1)=SYMGEN(COISYM)
C
C          DETERMINE WHEN TO SWITCH DFE FROM TRAINING MODE INTO
C          ADAPTIVE MODE. THE TAP VALUE OF THE LINEAR TRANSVERSAL
C          FILTER IS AVERAGED OVER 500 SYMBOLS. THIS AVERAGE TAP
C          VALUE IS COMPARED TO THE PREVIOUS AVERAGE TAP VALUE.
C          IF THE TWO VALUES ARE WITHIN CERTAIN ASSIGNED PERCENTAGE,
C          THE DFE IS AUTOMATICALLY SWITCHED INTO ADAPTIVE MODE.
C
           IF(COUNT.GT.II) THEN
             CONEWAVG=CONEWAVG+COTFTAP
             SUBNEWAVG=SUBNEWAVG+SUBTFTAP
           ELSEIF(COUNT.EQ.II) THEN
             COUNT=COUNT+INC
              IF(ABS(1.0-(COOLDAVG/CONEWAVG)).LE.FRACTION) COFLAG=1
              IF(ABS(1.0-(SUBOLDAVG/SUBNEWAVG)).LE.FRACTION) SUBFLAG=1
             COOLDAVG=CONEWAVG
             SUBOLDAVG=SUBNEWAVG
             CONEWAVG=0.0
             SUBNEWAVG=0.0
              IF(COFLAG.EQ.1.AND.SUBFLAG.EQ.1) THEN
                DO 76 I=1,4
                  COISYM(I)=0
76              SUBISYM(I)=0
                GOTO 77
              ENDIF
           ENDIF
20     CONTINUE
77     INDEX=0
C
C          GENERATE SIGNAL WAVEFORM FOR EYE DIAGRAM (8 SAMPLES/SYMBOL)
C
       DO 80 II=1,NPOINT/NEWPPSYM
        DO 85 K=0,NEWPPSYM-1
C
```

```
      COECNOISE=0.0
      SUBECNOISE=0.0
      COWNOISE=0.0
      SUBWNOISE=0.0
      COXTNOISE=0.0
      SUBXTNOISE=0.0
      COOUTSYM=0.0
      SUBOUTSYM=0.0
      INDEX=INDEX+1
C
C       SIGNAL AND NOISE AT THE SAMPLING INSTANT ARE CALCULATED
C
      DO 90 J=-N1,N2-1
        INDEX1=J*NEWPPSYM+K
        COECNOISE=COECNOISE+COREG(J)*COECFIR(INDEX1)
        SUBECNOISE=SUBECNOISE+SUBREG(J)*SUBECFIR(INDEX1)
        COOUTSYM=COOUTSYM+SUBREG(J)*SUBCHANFIR(INDEX1)
90    SUBOUTSYM=SUBOUTSYM+COREG(J)*COCHANFIR(INDEX1)
C
      DO 95 J=-N3*NEWPPSYM,N4*NEWPPSYM
        COWNOISE=COWNOISE+WNFIR(J)*WNREG1(J)
95    SUBWNOISE=SUBWNOISE+WNFIR(J)*WNREG2(J)
C
      DO 100 J=-N5*NEWPPSYM,N6*NEWPPSYM
        COXTNOISE=COXTNOISE+COXTFIR(J)*XTREG1(J)
100   SUBXTNOISE=SUBXTNOISE+SUBXTFIR(J)*XTREG2(J)
C
C       THE RECEIVED SIGNAL (SIGNAL+NOISE) AT THE CENTRAL OFFICE
C       AND THE SUBSCRIBER SIDE IS AMPLIFIED
C
      COEYE(INDEX)=(COOUTSYM+COWNOISE+COXTNOISE+COECNOISE)
     +                                      *GSUB*COTFTAP
      SUBEYE(INDEX)=(SUBOUTSYM+SUBWNOISE+SUBXTNOISE+SUBECNOISE)
     +                                           *GCO*SUBTFTAP
C
C       GENERATE NEW SYMBOLS AND SHIFT CONTENTS IN FIR
C       AND DFE DATA REGISTERS BY T/8 (1/8 SYMBOL PERIOD)
C
        DO 105 J=N4*NEWPPSYM,-N3*NEWPPSYM+1
          WNREG1(J)=WNREG1(J-1)
105     WNREG2(J)=WNREG2(J-1)
        WNREG1(-N3*NEWPPSYM)=GASDEV(IDUM)
        WNREG2(-N3*NEWPPSYM)=GASDEV(IDUM)
C
        DO 110 J=N6*NEWPPSYM,-N5*NEWPPSYM+1
          XTREG1(J)=XTREG1(J-1)
110     XTREG2(J)=XTREG2(J-1)
        XTREG1(-N5*NEWPPSYM)=GASDEV(IDUM)
        XTREG2(-N5*NEWPPSYM)=GASDEV(IDUM)
85      CONTINUE
C
        CODESIRE(II)=SUBREG(0)
        SUBDESIRE(II)=COREG(0)
C
```

```
      DO 115 J=N2,-N1+1,-1
        SUBREG(J)=SUBREG(J-1)
115     COREG(J)=COREG(J-1)
      SUBREG(-N1)=SYMGEN(SUBISYM)
80    COREG(-N1)=SYMGEN(COISYM)
C
C        GENERATE EYE DIAGRAM RIGHT BEFORE THE SLICER OF DFE.
C        A 2048 POINT ARRAY IS USED TO STORE THE SIGNAL WAVEFORM,
C        WHICH COMES OUT TO BE 256 TRACES(SYMBOLS) ON THE EYE
C        DIAGRAM (2048 SAMPLES/ 8 SAMPLES/SYMBOL = 256 SYMBOLS).
C
       SDFEYO=0.0
       CODFEYO=0.0
       DO 120 I=1,N3BACK
        SDFEYO=SDFEYO+SUBDFETAP(I)*SUBA(-I)
120    CODFEYO=CODFEYO+CODFETAP(I)*COA(-I)
       DO 125 I=1,NEWPPSYM
        COEYE(I)=COEYE(I)-CODFEYO
125    SUBEYE(I)=SUBEYE(I)-SDFEYO
C
       DO 130 J=1,NPOINT/NEWPPSYM-1
C
       DO 135 I=-N3BACK,-2
       COA(I)=COA(I+1)
135    SUBA(I)=SUBA(I+1)
       COA(-1)=CODESIRE(J)
       SUBA(-1)=SUBDESIRE(J)
       CODFEYO=0.0
       SDFEYO=0.0
       DO 140 I=1,N3BACK
        SDFEYO=SDFEYO+SUBDFETAP(I)*SUBA(-I)
140     CODFEYO=CODFEYO+CODFETAP(I)*COA(-I)
C
       DO 145 I=-NEWPPSYM/2+1,NEWPPSYM/2
        INDEX=J*NEWPPSYM+I
        COEYE(INDEX)=COEYE(INDEX)-CODFEYO
145    SUBEYE(INDEX)=SUBEYE(INDEX)-SDFEYO
130    CONTINUE
C
       DO 150 I=1,NPOINT/NEWPPSYM
        COSYM(I)=COEYE((I-1)*NEWPPSYM+1)
150    SUBSYM(I)=SUBEYE((I-1)*NEWPPSYM+1)
       DO 155 I=1,NEWPPSYM/2
       COEYE(I)=0.0
       COEYE(NPOINT-I+1)=0.0
       SUBEYE(I)=0.0
155    SUBEYE(NPOINT-I+1)=0.0
C
       NSYM=NPOINT/NEWPPSYM
C
C         CALCULATE EYE STATISTICS (EYE OPENING)
C
      OPEN(1,FILE='EYESTAT.DAT')
      WRITE(1,*) NAME
```

```
      CALL EYEOPEN(COSYM,CODESIRE,NSYM,1)
      CALL EYEOPEN(SUBSYM,SUBDESIRE,NSYM,2)
      CLOSE(1)
C
C         EYE DIAGRAM PLOT
C
      OPEN(3,FILE='COEYE.DAT')
      CALL EYE(NEWPPSYM/2+1,1,NPOINT,NEWPPSYM,COEYE)
      OPEN(3,FILE='SUBEYE.DAT')
      CALL EYE(NEWPPSYM/2+1,1,NPOINT,NEWPPSYM,SUBEYE)
     CLOSE(3)
     RETURN
     END
```

```
C**********************************************************************C
C                        PROGRAM    FSDFEBER                           C
C**********************************************************************C
C                                                                      C
C         This program will calculate the bit-error rate of a system   C
C     transmitting at a user assigned baud rate over a subscriber      C
C     loop. The bit and symbol error rate are estimated by Monte-Carlo C
C     method (error counting). The system impulse response and three   C
C     different noise impulse responses are calculated, and the re-    C
C     ceived signal is obtained by convolving the different impulse    C
C     responses with randomly generated numbers or symbols. This re-   C
C     ceived signal will then feed through an adaptive decision feed-  C
C     back equalizer with a fractionally-spaced linear equalizer at    C
C     the front, the detected symbols and the ideal transmitted        C
C     symbols are compared to obtain bit-error rate and symbol error   C
C     rate. The signal to noise ratio at the quantiser of the          C
C     fractionally-spaced DFE is also calculated.                      C
C                                                                      C
C     Input file - fsdfeber.in                                         C
C     Subroutine needed for execution - bereq.for                      C
C                                                                      C
C     Data required:                                                   C
C                     name - file with ABCD matrix of the loop         C
C                    name1 - output file name                          C
C                  symrate - transmit symbol rate (baud/s)             C
C                    alpha - roll off factor for raised cosine filter  C
C                            ( 0 ≤ x ≤ 1.0)                            C
C                    noise - white noise spectral density              C
C                    knext - near-end crosstalk coupling constant      C
C                       ec - attenuation provided by echo canceller (dB) C
C                      dbm - transmit power level in dBm at interface  C
C                   numsym - number of symbols to be transmitted       C
C                  percent - ratio to decide the lengths of precursors C
C                            and postcursors for impulse response      C
C                            (suggested value < 5e-3)                  C
C                   n3back - number of DFE taps                        C
C                      n1f - number of non-casual taps of FSLE (even)  C
C                      n2b - number of casual taps of FSLE (even)      C
C             delta,delta1 - step size for adaptation                  C
C                            delta: DFE        delta1: FSLE            C
C                 fraction - ratio to decide if equalizer has          C
C                            converged (suggested value < 5e-3)        C
C          ix,iy,iz,idum - seeds to initialize random number generator C
C                            (integer)                                 C
C                                                                      C
C**********************************************************************C
      INTEGER NPOINT,NSYM,PPSYM,START,NEWPPSYM,SAMPT1,SAMPT2,SAMPT3
      INTEGER N1,N2,N3,N4,N5,N6,N7,N8,N9,N10,N11,N12,N1F,N2B,N3BACK
      INTEGER IX,IY,IZ,IDUM,NUMSYM
      REAL PI,TPI,ALPHA,DF,SYMRATE,COGAIN,SUBGAIN
      REAL WSD,XTSD,SYMVAR,NOISE,KNEXT,EC,DBM,RLOAD
      REAL DELTA,DELTA1,FRACTION,PERCENT
      REAL IMP(2052),MULTI1(2052),MULTI2(2052)
      REAL COCHANFIR(-40:80),SUBCHANFIR(-40:80)
```

```
      REAL COECFIR(-40:80),SUBECFIR(-40:80)
      REAL COXTFIR(-40:40),SUBXTFIR(-40:40)
      REAL WNFIR(-40:40)
      CHARACTER*30 NAME,NAME1
      COMMON /BLK1/ NPOINT,NSYM,PPSYM
      COMMON /BLK2/ ALPHA,DF
      COMMON /BLK3/ PI,TPI
      COMMON /BLK4/ START,NEWPPSYM
      COMMON /GAIN/ COGAIN,SUBGAIN
      COMMON /RAND/ IX,IY,IZ,IDUM
      COMMON /DFEBLK/ DELTA,DELTA1,FRACTION,NUMSYM
C
C          READ INPUT DATA
C
      OPEN(1,FILE='FSDFEBER.IN')
       READ(1,*) NAME,NAME1
       READ(1,*) SYMRATE,ALPHA,NOISE,KNEXT,EC,DBM
       READ(1,*) PERCENT,DELTA,DELTA1,FRACTION
       READ(1,*) N1F,N2B,N3BACK,NUMSYM
       READ(1,*) IX,IY,IZ,IDUM
      CLOSE(1)
      OPEN(8,FILE=NAME)
C
C          INITIALIZATION
C
      NPOINT=2048
      PPSYM=16
      NSYM=128
      DF=SYMRATE/NSYM
      PI=3.1415926536
      TPI=2.0*PI
      START=513
      NEWPPSYM=2
      RLOAD=135.0
      SYMVAR=(3**2+(-3)**2+1**2+(-1)**2)/4
      WSD=SQRT(NOISE*NEWPPSYM*SYMRATE*RLOAD)
      XTSD=SQRT(SYMVAR*NEWPPSYM)
C
C          GENERATE A POSITIVE AND A NEGATIVE IMPULSE, AND TRANSFORM
C          INTO FREQUENCY DOMAIN.
C
      IMP(START)=PPSYM
      IMP(START+NPOINT/2)=-PPSYM
       CALL REALFT(IMP,NPOINT/2,1)
C
      DO 5 I=1,NPOINT
        MULTI1(I)=IMP(I)
5     MULTI2(I)=IMP(I)
C
C          PROVIDE PROPER SCALING AT THE LINE AND NETWORK INTERFACE
C
      CALL VSCALE(COGAIN,SUBGAIN,DBM,MULTI1,MULTI2)
C
C          DETERMINE THE COLOURED NOISE IMPULSE RESPONSE, SAMPLE
```

```
C          AND FEED THE COEFFICIENTS INTO THE FIR FILTER. THE LENGTHS
C          REQUIRED FOR PRECURSORS AND POSTCURSORS ARE CALCULATED.
C
      DO 10 I=1,NPOINT
10      MULTI1(I)=IMP(I)
C
      CALL WN(WSD,MULTI1)
      CALL SAMPPHASE(START,1,SAMPT1,MULTI1)
      CALL IMPLENGTH(SAMPT1,N3,N4,PERCENT,MULTI1,20,20)
       DO 15 I=-N3*2,N4*2
15       WNFIR(I)=MULTI1(SAMPT1+I*PPSYM/2)
C
      WRITE(6,*) 'COLOURED NOISE :  PRECURSOR = ',N3
      WRITE(6,*) '                 POSTCURSOR = ',N4
C
C          DETERMINE THE NEAR-END CROSSTALK IMPULSE RESPONSE, SAMPLE
C          AND FEED THE COEFFICIENTS INTO THE FIR FILTER. THE LENGTHS
C          REQUIRED FOR PRECURSORS AND POSTCURSORS ARE CALCULATED.
C
      DO 20 I=1,NPOINT
        MULTI1(I)=IMP(I)
20    MULTI2(I)=IMP(I)
      CALL XT(KNEXT,XTSD,MULTI1,MULTI2)
C
C          C.O. SIDE
C
      CALL SAMPPHASE(START,1,SAMPT1,MULTI1)
      CALL IMPLENGTH(SAMPT1,N5,N6,PERCENT,MULTI1,20,20)
C
C          SUBSCRIBER SIDE
C
      CALL SAMPPHASE(START,1,SAMPT2,MULTI2)
      CALL IMPLENGTH(SAMPT2,N7,N8,PERCENT,MULTI2,20,20)
C
C          LENGTHS REQUIRED FOR PRECURSORS AND POSTCURSORS AT BOTH
C          ENDS ARE CHOSEN SUCH THAT THE LONGER ONE WILL BE USED.
C
       WRITE(6,*) 'NEXT NOISE:  CO PRECURSOR = ',N5,' POSTCURSOR = ',N6
       WRITE(6,*) '           : SUB PRECURSOR = ',N7,' POSTCURSOR = ',N8
      N5=MAX(N5,N7)
      N6=MAX(N6,N8)
       WRITE(6,*) ' FINAL LENGTH : PRECURSOR = ',N5,' POSTCURSOR = ',N6
C
      DO 25 I=-N5*2,N6*2
        COXTFIR(I)=MULTI1(SAMPT1+I*PPSYM/2)
25    SUBXTFIR(I)=MULTI2(SAMPT2+I*PPSYM/2)
C
C          DETERMINE THE ECHO IMPULSE RESPONSE, SAMPLE AT THE WORST
C          SAMPLING PHASE ( MAXIMIZE ECHO ). THE LENGTHS REQUIRED
C          FOR PRECURSORS AND POSTCURSORS ARE CALCULATED.
C
       OPEN(8,FILE=NAME)
C
      DO 30 I=1,NPOINT
```

```
        MULTI1(I)=IMP(I)
30    MULTI2(I)=IMP(I)
      CALL ECHO(EC,MULTI1,MULTI2)
C
C          C.O. SIDE
C
      CALL SAMPPHASE(START,1,SAMPT1,MULTI1)
      CALL IMPLENGTH(SAMPT1,N9,N10,PERCENT,MULTI1,20,40)
C
C          SUBSCRIBER SIDE
C
      CALL SAMPPHASE(START,1,SAMPT2,MULTI2)
      CALL IMPLENGTH(SAMPT2,N11,N12,PERCENT,MULTI2,20,40)
C
C          DETERMINE THE CHANNEL IMPULSE RESPONSE, SAMPLE AT THE
C          PHASE TO MINIMIZE FIRST PRECURSOR. THE LENGTHS REQUIRED
C          FOR PRECURSORS AND POSTCURSORS ARE CALCULATED.
C
      OPEN(8,FILE=NAME)
      CALL IMPULSE(IMP)
C
      CALL SAMPPHASE(START,2,SAMPT3,IMP)
      CALL IMPLENGTH(SAMPT3,N1,N2,PERCENT,IMP,20,40)
C
C          LENGTH REQUIRED FOR PRECURSORS AND POSTCURSORS AT BOTH
C          ENDS FOR THE CHANNEL AND ECHO IMPULSE RESPONSES ARE CHOSEN
C          SUCH THAT THE LONGER ONE WILL BE USED.
C
      WRITE(6,*) 'ECHO NOISE: CO PRECURSOR = ',N9,' POSTCURSOR = ',N10
      WRITE(6,*) '           :SUB PRECURSOR = ',N11,' POSTCURSOR = ',N12
      WRITE(6,*) 'CHANNEL    : CO PRECURSOR = ',N1,' POSTCURSOR = ',N2
C
      N1=MAX(N1,N9,N11)
      N2=MAX(N2,N10,N12)
      WRITE(6,*) 'FINAL LENGTH : PRECURSOR = ',N1,' POSTCURSOR = ',N2
C
      DO 35 I=-N1*2,N2*2
        COCHANFIR(I)=IMP(SAMPT3+I*PPSYM/2)*COGAIN
        SUBCHANFIR(I)=IMP(SAMPT3+I*PPSYM/2)*SUBGAIN
        COECFIR(I)=MULTI1(SAMPT1+I*PPSYM/2)
35    SUBECFIR(I)=MULTI2(SAMPT2+I*PPSYM/2)
C
C          EQUALIZED BY FSLE & DFE
C
       CALL FSDFE(COCHANFIR,SUBCHANFIR,COECFIR,SUBECFIR,COXTFIR,
     + SUBXTFIR,WNFIR,N1,N2,N3,N4,N5,N6,N1F,N2B,N3BACK,NAME,NAME1)
C
       STOP
       END
C**********************************************************************C
C                                                                      C
C          This subroutine uses the LMS algorithm to adaptively       C
C     equalize the signal by a decision feedback equalizer with a     C
C     fractionally-spaced linear equalizer. The dfe has n3back taps,  C
```

```
C     and the FSLE has n1f non-casual and n2b casual taps.            C
C     The equalizer is first set in the training mode. If the         C
C     equalizer taps have converged, it is automatically set in the   C
C     adaptive mode and BER calculation begins. The signal and        C
C     noise are generated from their impulse responses by convolving  C
C     with randomly generated numbers or symbols. The quantiser signal C
C     to noise ratio is also calculated.                              C
C                                                                     C
C      cochanfir - co -> sub channel impulse response                 C
C     subchanfir - sub -> co channel impulse response                 C
C        coecfir - co side echo impulse response                      C
C       subecfir - sub side echo impulse response                     C
C        coxtfir - co side near-end crosstalk impulse response        C
C       subxtfir - sub side near-end crosstalk impulse response       C
C          wnfir - coloured zero-mean gaussian noise impulse response C
C                  (same for both ends)                               C
C          n1/n2 - number of symbol periods required to represent the C
C                  pre/postcursor of channel and echo impulse response C
C          n3/n4 - number of symbol periods required to represent the C
C                  pre/postcursor of coloured noise impulse response  C
C          n5/n6 - number of symbol periods required to represent the C
C                  pre/postcursor of near-end crosstalk impulse       C
C                  response                                           C
C         n3back - number of dfe taps ( maximum - 50, can increase by C
C                  changing the code )                                C
C        n1f/n2b - number of non-causal/casual taps for FSLE          C
C                  ( maximum - 80 for both cases, even integer )      C
C                  loop file name                                     C
C                  utput file name ( unit 1 is used for writing )     C
C                                                                     C
C***                ***************************************************C
                    FE(COCHANFIR,SUBCHANFIR,COECFIR,SUBECFIR,COXTFIR,
                    .N1,N2,N3,N4,N5,N6,N1F,N2B,N3BACK,NAME,NAME1)
                     ,N4,N5,N6,N1F,N2B,N3BACK
                     BFLAG,SW,COUNT,INC
                      NUMSYM,SYMA,SYMB
                      CISION,IX,IY,IZ,IDUM
                  (2),SUBBER(2)
             COISYM(4),SUBISYM(4)
      GER COREG(-20:40),SUBREG(-20:40)
      INTEGER COA(-50:-1),SUBA(-50:-1)
      REAL COGAIN,SUBGAIN,GCO,GSUB,COERROR,SUBERROR
      REAL CODFEY0,SDFEY0,COTFY0,SUBTFY0
      REAL COOLDAVG,SUBOLDAVG,CONEWAVG,SUBNEWAVG
      REAL DELTA,DELTA1,FRACTION,GASDEV
      REAL COWNOISE,SUBWNOISE,COECNOISE,SUBECNOISE,COXTNOISE,SUBXTNOISE
      REAL CONOISEP,SUBNOISEP,COSIGP,SUBSIGP
      REAL COOUTSYM,SUBOUTSYM
      REAL CODFETAP(50),SUBDFETAP(50)
      REAL COTFTAP(-80:80),SUBTFTAP(-80:80)
      REAL COIN(-80:80),SUBIN(-80:80)
      REAL WNREG1(-40:40),WNREG2(-40:40)
      REAL XTREG1(-40:40),XTREG2(-40:40)
      REAL COCHANFIR(-40:80),SUBCHANFIR(-40:80)
```

```
      REAL COECFIR(-40:80),SUBECFIR(-40:80)
      REAL COXTFIR(-40:40),SUBXTFIR(-40:40)
      REAL WNFIR(-40:40)
      COMMON /GAIN/ COGAIN,SUBGAIN
      COMMON /RAND/ IX,IY,IZ,IDUM
      COMMON /DFEBLK/ DELTA,DELTA1,FRACTION,NUMSYM
      CHARACTER*30 NAME,NAME1
C
C           INITIALIZATION
C
        SW=1
        INC=500
        COUNT=INC
        GCO=1.0/COCHANFIR(0)
        GSUB=1.0/SUBCHANFIR(0)
        COTFTAP(0)=0.5
        SUBTFTAP(0)=0.5
        COOLDAVG=0.0
        SUBOLDAVG=0.0
        CONEWAVG=0.0
        SUBNEWAVG=0.0
        COFLAG=0
        SUBFLAG=0
        COBER(1)=0
        COBER(2)=0
        SUBBER(1)=0
        SUBBER(2)=0
C
C           FILLED THE FIR DATA REGISTERS AND FSLE + DFE DATA REGISTERS
C
        DO 1 I=-80,80
          COIN(I)=0.0
1       SUBIN(I)=0.0
        DO 5 I=-20,40
          COREG(I)=SYMGEN(COISYM)
5       SUBREG(I)=SYMGEN(SUBISYM)
        DO 10 I=-40,40
          WNREG1(I)=GASDEV(IDUM)
          WNREG2(I)=GASDEV(IDUM)
          XTREG1(I)=GASDEV(IDUM)
10      XTREG2(I)=GASDEV(IDUM)
        DO 15 I=1,50
          COA(-I)=0.0
15      SUBA(-I)=0.0
C
C           FSLE + DFE EQUALIZER: SW=1 (TRAINING MODE)
C                                   =2 (ADAPTIVE MODE)
C
        DO 20 II=1,NUMSYM
         DO 25 KK=0,1
C
          COECNOISE=0.0
          SUBECNOISE=0.0
          COWNOISE=0.0
```

```
      SUBWNOISE=0.0
      COXTNOISE=0.0
      SUBXTNOISE=0.0
      COOUTSYM=0.0
      SUBOUTSYM=0.0
C
C       SIGNAL AND NOISE AT nT-T/2 AND nT ARE CALCULATED
C
      DO 30 J=-N3*2,N4*2
        COWNOISE=COWNOISE+WNFIR(J)*WNREG1(J)
30    SUBWNOISE=SUBWNOISE+WNFIR(J)*WNREG2(J)
C
      DO 35 J=-N5*2,N6*2
        COXTNOISE=COXTNOISE+COXTFIR(J)*XTREG1(J)
35    SUBXTNOISE=SUBXTNOISE+SUBXTFIR(J)*XTREG2(J)

       IF(KK.EQ.0) THEN
         DO 40 J=-N1*2+1,N2*2-1,2
           COECNOISE=COECNOISE+COREG((J+1)/2)*COECFIR(J)
           SUBECNOISE=SUBECNOISE+SUBREG((J+1)/2)*SUBECFIR(J)
           COOUTSYM=COOUTSYM+SUBREG((J+1)/2)*SUBCHANFIR(J)
40       SUBOUTSYM=SUBOUTSYM+COREG((J+1)/2)*COCHANFIR(J)
         COOUTSYM=(COOUTSYM+COWNOISE+COXTNOISE+COECNOISE)*GSUB
         SUBOUTSYM=(SUBOUTSYM+SUBWNOISE+SUBXTNOISE+SUBECNOISE)*GCO
       ELSE
         DO 45 J=-N1*2,N2*2,2
           COECNOISE=COECNOISE+COREG(J/2)*COECFIR(J)
           SUBECNOISE=SUBECNOISE+SUBREG(J/2)*SUBECFIR(J)
           COOUTSYM=COOUTSYM+SUBREG(J/2)*SUBCHANFIR(J)
45       SUBOUTSYM=SUBOUTSYM+COREG(J/2)*COCHANFIR(J)
         COOUTSYM=(COOUTSYM+COWNOISE+COXTNOISE+COECNOISE)*GSUB
         SUBOUTSYM=(SUBOUTSYM+SUBWNOISE+SUBXTNOISE+SUBECNOISE)*GCO
       ENDIF
C
C       SHIFT THE AMPLIFIED SIGNAL INTO THE FSLE DATA REGISTERS
c       BY HALF SYMBOL PERIOD (T/2)
C
      DO 50 I=-N2B,N1F-1
        COIN(I)=COIN(I+1)
50    SUBIN(I)=SUBIN(I+1)
      COIN(N1F)=COOUTSYM
      SUBIN(N1F)=SUBOUTSYM
C
C       SHIFT CONTENTS IN FIR DATA REGISTERS BY T/2
C
       DO 60 J=N4*2,-N3*2+1,-1
         WNREG1(J)=WNREG1(J-1)
60     WNREG2(J)=WNREG2(J-1)
       WNREG1(-N3*2)=GASDEV(IDUM)
       WNREG2(-N3*2)=GASDEV(IDUM)
C
       DO 70 J=N6*2,-N5*2+1,-1
         XTREG1(J)=XTREG1(J-1)
70     XTREG2(J)=XTREG2(J-1)
```

```
          XTREG1(-N5*2)=GASDEV(IDUM)
          XTREG2(-N5*2)=GASDEV(IDUM)
25      CONTINUE
C
C          THE RECEIVED SIGNAL IS EQUALIZED BY THE ADAPTIVE
C          FRACTIONALLY-SPACED LINEAR EQUALIZER
C
        COTFY0=0.0
        SUBTFY0=0.0
        DO 75 I=-N1F,N2B
          COTFY0=COTFY0+COTFTAP(I)*COIN(-I)
75      SUBTFY0=SUBTFY0+SUBTFTAP(I)*SUBIN(-I)
C
C          FEEDBACK EQUALIZER OUTPUT
C
        CODFEY0=0.0
        SDFEY0=0.0
        DO 80 I=1,N3BACK
          CODFEY0=CODFEY0+CODFETAP(I)*COA(-I)
80      SDFEY0=SDFEY0+SUBDFETAP(I)*SUBA(-I)
C
C          SW = 1 : THE EQUALIZED SIGNAL IS COMPARED TO THE IDEAL
C                   TRANSMITTED SYMBOL. THE DIFFERENCE IS USED TO
C                   UPDATE THE TAPS.
C
C             = 2 : THE EQUALIZED SIGNAL IS PASSED TO A SLICER.
C                   THE DIFFERENCE BETWEEN THE DETECTED SYMBOL
C                   AND THE RECEIVED SIGNAL IS USED TO UPDATE
C                   THE TAPS. THE BIT AND SYMBOL ERRORS WILL BE
C                   COUNTED.
C
        IF(SW.EQ.1) THEN
          COERROR=SUBREG(N1F/2)-(COTFY0-CODFEY0)
          SUBERROR=COREG(N1F/2)-(SUBTFY0-SDFEY0)
        ELSE
          SYMA=DECISION(COTFY0-CODFEY0)
          SYMB=DECISION(SUBTFY0-SDFEY0)
          COERROR=SYMA-(COTFY0-CODFEY0)
          SUBERROR=SYMB-(SUBTFY0-SDFEY0)
          CONOISEP=CONOISEP+(SUBREG(N1F/2)-(COTFY0-CODFEY0))**2
          SUBNOISEP=SUBNOISEP+(COREG(N1F/2)-(SUBTFY0-SDFEY0))**2
          COSIGP=COSIGP+SUBREG(N1F/2)**2
          SUBSIGP=SUBSIGP+COREG(N1F/2)**2
            IF(SUBREG(N1F/2).NE.SYMA) THEN
              CALL BITSYMERROR(SUBREG(N1F/2)-SYMA,COBER)
            ENDIF
            IF(COREG(N1F/2).NE.SYMB) THEN
              CALL BITSYMERROR(COREG(N1F/2)-SYMB,SUBBER)
            ENDIF
        ENDIF
C
C          UPDATE FSLE TAP VALUE
C
        DO 85 I=-N1F,N2B
```

```
      COTFTAP(I)=COTFTAP(I)+DELTA1*COERROR*COIN(-I)
85    SUBTFTAP(I)=SUBTFTAP(I)+DELTA1*SUBERROR*SUBIN(-I)
C
C         UPDATE DFE TAP VALUE
C
      DO 90 I=1,N3BACK
        CODFETAP(I)=CODFETAP(I)-DELTA*COERROR*COA(-I)
90    SUBDFETAP(I)=SUBDFETAP(I)-DELTA*SUBERROR*SUBA(-I)
C
C         GENERATE NEW SYMBOLS AND SHIFT CONTENTS IN
C         FSLE + DFE DATA REGISTERS BY T (SYMBOL PERIOD)
C
      DO 95 I=-N3BACK,-2
        SUBA(I)=SUBA(I+1)
95    COA(I)=COA(I+1)
C
C         SW = 1: IDEAL TRANSMITTED SYMBOL WILL BE FED INTO
C                 DFE DATA REGISTERS.
C
C              2: DETECTED SYMBOLS FROM SLICER WILL BE FED
C                 INTO DFE DATA REGISTERS.
C
      IF(SW.EQ.1) THEN
        COA(-1)=SUBREG(N1F/2)
        SUBA(-1)=COREG(N1F/2)
      ELSE
        COA(-1)=SYMA
        SUBA(-1)=SYMB
      ENDIF
C
      DO 100 J=N2,-N1+1,-1
        SUBREG(J)=SUBREG(J-1)
100   COREG(J)=COREG(J-1)
      SUBREG(-N1)=SYMGEN(SUBISYM)
      COREG(-N1)=SYMGEN(COISYM)
C
C         DETERMINE WHEN TO SWITCH FSLE + DFE FROM TRAINING MODE INTO
C         ADAPTIVE MODE. THE MAIN TAP VALUE OF THE FSLE IS AVERAGED
C         OVER 500 SYMBOLS. THIS AVERAGE TAP VALUE IS COMPARED TO THE
C         PREVIOUS AVERAGE TAP VALUE. IF THE TWO VALUES ARE WITHIN
C         CERTAIN ASSIGNED PERCENTAGE, THE FSLE + DFE IS AUTOMATICALLY
C         SWITCHED INTO ADAPTIVE MODE AND BEGINS TO CALCULATE THE
C         BIT-ERROR RATE. COFLAG AND SUBFLAG WILL BE SET TO 1 IF
C         FSLE + DFE AT BOTH ENDS ARE TRAINED AND CONVERGED.
C
         IF(SW.EQ.1) THEN
           IF(COUNT.GT.II) THEN
             CONEWAVG=CONEWAVG+COTFTAP(0)
             SUBNEWAVG=SUBNEWAVG+SUBTFTAP(0)
           ELSEIF(COUNT.EQ.II) THEN
             COUNT=COUNT+INC
              IF(ABS(1.0-(COOLDAVG/CONEWAVG)).LE.FRACTION) COFLAG=1
              IF(ABS(1.0-(SUBOLDAVG/SUBNEWAVG)).LE.FRACTION) SUBFLAG=1
             COOLDAVG=CONEWAVG
```

```
          SUBOLDAVG=SUBNEWAVG
          CONEWAVG=0.0
          SUBNEWAVG=0.0
           IF(COFLAG.EQ.1.AND.SUBFLAG.EQ.1) THEN
             TRAINSYM=II
              DO 105 I=1,4
                COISYM(I)=0
105           SUBISYM(I)=0
             SW=2
           ENDIF
        ENDIF
      ENDIF
20    CONTINUE
      NUMSYM=NUMSYM-TRAINSYM
C
C        OUTPUT RESULTS
C
      OPEN(1,FILE=NAME1)
       IF(SW.EQ.1) THEN
         WRITE(1,*) 'EQUALIZER HAS NOT CONVERGED'
         STOP
       ENDIF
       IF(SW.EQ.2) WRITE(1,*) 'C.O. AND SUB FSLE + DFE HAVE CONVERGED'
      WRITE(1,*) NAME
      WRITE(1,*) 'TRAINING PERIOD (SYM) = ',TRAINSYM
      WRITE(1,*) 'TRANSMITTED SYMBOLS = ',NUMSYM
      WRITE(1,*)
      WRITE(1,*) 'CO :'
      WRITE(1,*)
      WRITE(1,*) ' SYMBOL  3 = ',SUBISYM(1),' SYMBOL  1 = ',SUBISYM(2)
      WRITE(1,*) ' SYMBOL -1 = ',SUBISYM(3),' SYMBOL -3 = ',SUBISYM(4)
      WRITE(1,*)
      WRITE(1,*) ' NUMBER OF BIT ERROR = ', COBER(1)
      WRITE(1,*) ' NUMBER OF SYM ERROR = ', COBER(2)
      WRITE(1,*) ' QSNR (dB)           = ', 10*LOG10(COSIGP/CONOISEP)
      WRITE(1,*) '       BIT ERROR RATE = ', COBER(1)/(NUMSYM*2.0)
      WRITE(1,*) '       SYM ERROR RATE = ', COBER(2)*1.0/NUMSYM
      WRITE(1,*)
      WRITE(1,*) 'SUB :'
      WRITE(1,*)
      WRITE(1,*) ' SYMBOL  3 = ',COISYM(1),' SYMBOL  1 = ',COISYM(2)
      WRITE(1,*) ' SYMBOL -1 = ',COISYM(3),' SYMBOL -3 = ',COISYM(4)
      WRITE(1,*)
      WRITE(1,*) ' NUMBER OF BIT ERROR = ', SUBBER(1)
      WRITE(1,*) ' NUMBER OF SYM ERROR = ', SUBBER(2)
      WRITE(1,*) ' QSNR (dB)           = ', 10*LOG10(SUBSIGP/SUBNOISEP)
      WRITE(1,*) '       BIT ERORR RATE = ', SUBBER(1)/(NUMSYM*2.0)
      WRITE(1,*) '       SYM ERORR RATE = ', SUBBER(2)*1.0/NUMSYM
      CLOSE(1)
      RETURN
      END
```

```
C<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<C
C                                                                     C
C           The subroutines in this program BEREQ.FOR are needed      C
C      for the execution of the program DFEBER.FOR, DFEEYE.FOR, and   C
C      FSDFEBER.FOR. Subroutines in BEREQ.FOR need to be compiled     C
C      only once and can link to any other main program for execution. C
C                                                                     C
C>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>C
C
C
C*********************************************************************C
C                                                                     C
C           This subroutine will scale the power at the line and the  C
C      network termination to x dbm.                                  C
C                                                                     C
C            dbm = desired power level in dbm                         C
C        scaleco = scaling factor at the central office               C
C       scalesub = scaling factor at the subscriber side              C
C     spco/spsub = array with the spectrum of two isolated impulses   C
C                  (+ve and -ve)                                      C
C                                                                     C
C*********************************************************************C
      SUBROUTINE VSCALE(SCALECO,SCALESUB,DBM,SPCO,SPSUB)
      INTEGER NPOINT,NSYM,PPSYM
      REAL SPCO(NPOINT),SPSUB(NPOINT)
      REAL RS,RM,COSUM,SUBSUM,SCALESUB,SCALECO,DBM,VLEVEL
      REAL SQRC,PI,TPI,ALPHA,DF
      COMPLEX NUMSUB,NUMCO
      COMPLEX V,A,B,C,D,DEN
      COMPLEX CMPLX
      COMMON /BLK1/ NPOINT,NSYM,PPSYM
      COMMON /BLK2/ ALPHA,DF
      COMMON /BLK3/ PI,TPI
C
C         RS AND RM ARE THE TERMINATION RESISTANCE
C
      RS=135.0
      RM=135.0
C
C         NO DC RESPONSE ON THE LINE
C
         READ(8,*) A,B
         READ(8,*) C,D
          SPSUB(1)=0.0
          SPSUB(2)=0.0
          SPCO(1)=0.0
          SPCO(2)=0.0
C
C         AC RESPONSE
C
         DO 20 I=1,NSYM
          READ(8,*) A,B
          READ(8,*) C,D
           DEN=A*RS+B+C*RS*RM+D*RM
```

```
            IF((I*1.0)/NSYM.LT.(1.0-ALPHA)/2.0) THEN
               NUMSUB=(D*RM+B)/DEN
                NUMCO=(A*RS+B)/DEN
            ELSEIF((I*1.0)/NSYM.GT.(1.0+ALPHA)/2.0) THEN
               NUMSUB=0.0
                NUMCO=0.0
            ELSE
               SQRC=SQRT(0.5*(1.0-SIN((PI/ALPHA)*((I*1.0)/NSYM-0.5))))
               NUMSUB=(D*RM+B)*SQRC/DEN
                NUMCO=(A*RS+B)*SQRC/DEN
            ENDIF
C
C        SPCO IS THE VOLTAGE APPEARS AT THE LINE TERMINATION (CO)
C
          V=CMPLX(SPCO(2*I+1),SPCO(2*I+2))*NUMCO
          SPCO(2*I+1)=REAL(V)
          SPCO(2*I+2)=AIMAG(V)
C
C        SPSUB IS THE VOLTAGE APPEARS AT THE NETWORK TERMINATION (SUB)
C
          V=CMPLX(SPSUB(2*I+1),SPSUB(2*I+2))*NUMSUB
          SPSUB(2*I+1)=REAL(V)
20        SPSUB(2*I+2)=AIMAG(V)
C
C        RESPONSE AT HIGHER FREQUENCIES IS ZERO
C
        DO 25 I=2*NSYM+3,NPOINT
         SPCO(I)=0.0
25       SPSUB(I)=0.0
C
         REWIND(8)
C
C         RESPONSE IN TIME DOMAIN
C
        CALL REALFT(SPSUB,NPOINT/2,-1)
        CALL REALFT(SPCO,NPOINT/2,-1)
C
C         SCALING OF IFFT
C
        DO 30 I=1,NPOINT
          SPSUB(I)=SPSUB(I)/(NPOINT/2)
30      SPCO(I)=SPCO(I)/(NPOINT/2)
C
C         SET THE POWER TO DESIRED LEVEL AND DETERMINE THE SCALING
FACTOR
C
        COSUM=0.0
        SUBSUM=0.0
        DO 35 I=1,NPOINT
          COSUM=COSUM+SPCO(I)**2
35      SUBSUM=SUBSUM+SPSUB(I)**2
        VLEVEL=SQRT(0.135/5.0*10**(DBM/10.0))
        SCALESUB=VLEVEL*SQRT(32.0/SUBSUM)
        SCALECO=VLEVEL*SQRT(32.0/COSUM)
```

```
C
       WRITE(6,*) 'CO SCALING FACTOR = ',SCALECO
       WRITE(6,*) 'SUB SCALING FACTOR = ',SCALESUB
      RETURN
      END
C*********************************************************************C
C                                                                     C
C         This subroutine will calculate the composite impulse        C
C     response of the line, the transmit and receive filters. The     C
C     response is the same on the central office side and the        C
C     subscriber side.                                                C
C                                                                     C
C     impresp = array with the composite impulse response             C
C                                                                     C
C*********************************************************************C
      SUBROUTINE IMPULSE(IMPRESP)
      INTEGER NPOINT,NSYM,PPSYM
      REAL ALPHA,DF,PI,TPI,RC
      REAL IMPRESP(NPOINT)
      REAL RS,RM
      COMPLEX A,B,C,D,DEN,TRANSF,CSIG,EMPHASIS
      COMPLEX CMPLX
      COMMON /BLK1/ NPOINT,NSYM,PPSYM
      COMMON /BLK2/ ALPHA,DF
      COMMON /BLK3/ PI,TPI
C
C         TERMINATION IMPEDANCE
C
      RS=135.0
      RM=135.0
C
C         DC RESPONSE = 0.0
C
      READ(8,*) A,B
      READ(8,*) C,D
       IMPRESP(1)=0.0
       IMPRESP(2)=0.0
C
C         AC RESPONSE
C
       DO 20 I=1,NSYM
         READ(8,*) A,B
         READ(8,*) C,D
C
C         THE LINE, RAISED COSINE FILTER, AND PRE-EMPHASIS (1-Z**-1)
C         ARE USED FOR THE SYSTEM
C
         DEN=A*RS+B+C*RS*RM+D*RM
         TRANSF=RS/DEN
         CSIG=CMPLX(IMPRESP(2*I+1),IMPRESP(2*I+2))
         EMPHASIS=CMPLX(1.0-COS(I*TPI/NSYM),SIN(I*TPI/NSYM))
C
         IF((I*1.0)/NSYM.LT.(1.0-ALPHA)/2.0) THEN
           CSIG=CSIG*TRANSF*EMPHASIS
```

```
            IMPRESP(2*I+1)=REAL(CSIG)
            IMPRESP(2*I+2)=AIMAG(CSIG)
          ELSEIF((I*1.0)/NSYM.GT.(1.0+ALPHA)/2.0) THEN
            IMPRESP(2*I+1)=0.0
            IMPRESP(2*I+2)=0.0
          ELSE
            RC=(1.0-SIN((PI/ALPHA)*((I*1.0)/NSYM-0.5)))/2.0
            CSIG=CSIG*TRANSF*RC*EMPHASIS
            IMPRESP(2*I+1)=REAL(CSIG)
            IMPRESP(2*I+2)=AIMAG(CSIG)
          ENDIF
20      CONTINUE
C
C           RESPONSE AT HIGHER FREQUENCIES IS ZERO
C
        DO 25 I=2*NSYM+3,NPOINT
25        IMPRESP(I)=0.0
C
C           RESPONSE IN TIME DOMAIN
C
        CALL REALFT(IMPRESP,NPOINT/2,-1)
C
C           SCALING OF IFFT
C
        DO 40 I=1,NPOINT
40        IMPRESP(I)=IMPRESP(I)/(NPOINT/2.0)
C
       CLOSE(8)
       RETURN
       END
C**********************************************************************C
C                                                                      C
C          This subroutine will calculate the white noise impulse      C
C      response. Response is the same on the central office side and   C
C      the subscriber side.                                            C
C                                                                      C
C     wnresp = array with the white noise impulse response             C
C       wsd = power level scaling factor                               C
C                                                                      C
C**********************************************************************C
       SUBROUTINE WN(WSD,WNRESP)
       INTEGER NPOINT,PPSYM,NSYM,START,NEWPPSYM
       REAL ALPHA,DF,PI,TPI,TEMP,WSD
       REAL WNRESP(NPOINT)
       COMPLEX EMPHASIS,VS,VOUT,CMPLX
       COMMON /BLK1/ NPOINT,NSYM,PPSYM
       COMMON /BLK2/ ALPHA,DF
       COMMON /BLK3/ PI,TPI
       COMMON /BLK4/ START,NEWPPSYM
C
C           DC RESPONSE = 0
C
         WNRESP(1)=0.0
         WNRESP(2)=0.0
```

```
C
C          AC RESPONSE
C
        DO 10 I-1,NSYM
         VS-CMPLX(WNRESP(2*I+1),WNRESP(2*I+2))
         EMPHASIS-CMPLX(1-COS(I*TPI/NSYM),SIN(I*TPI/NSYM))
          IF((I*1.0)/NSYM.LT.(1.0-ALPHA)/2.0) THEN
            VOUT-VS*EMPHASIS*WSD
            WNRESP(2*I+1)-REAL(VOUT)
            WNRESP(2*I+2)-IMAG(VOUT)
          ELSEIF((I*1.0)/NSYM.GT.(1.0+ALPHA)/2.0) THEN
            WNRESP(2*I+1)-0.0
            WNRESP(2*I+2[illegible]
          ELSE
            TEMP-SQRT[illegible]N((PI/ALPHA)*((I*1.0)/NSYM-0.5)))*0.5)
            VOUT-VS*EM[illegible]S*TEMP*WSD
            WNRESP(2*I+1)-REAL(VOUT)
            WNRESP(2*I+2)-IMAG(VOUT)
          ENDIF
10      CONTINUE
C
C          RESPONSE AT HIGHER FREQUENCIES IS ZERO
C
        DO 15 I-2*NSYM+3,NPOINT
15       WNRESP(I)-0.0
C
C          RESPONSE IN TIME DOMAIN
C
         CALL REALFT(WNRESP,NPOINT/2,-1)
C
C          SCALING OF IFFT
C
        DO 20 I-1,NPOINT
20       WNRESP(I)-WNRESP(I)/(NPOINT*0.5*NEWPPSYM)
      RETURN
      END
C**********************************************************************C
C                                                                      C
C          This subroutine will calculate the near-end crosstalk       C
C     impulse response on the central office side and the subscriber  C
C     side.                                                            C
C                                                                      C
C      coxt - array with the C.O. NEXT impulse response                C
C     subxt - array with the SUB. NEXT impulse response                C
C     knext - cable-dependent constant                                 C
C      xtsd - power level scaling factor                               C
C                                                                      C
C**********************************************************************C
      SUBROUTINE XT(KNEXT,XTSD,COXT,SUBXT)
      INTEGER NPOINT,NSYM,PPSYM,START,NEWPPSYM
      REAL ALPHA,DF,PI,TPI,COGAIN,SUBGAIN,TEMP,FREQ,KNEXT,XTSD
      REAL RS,RM,COXTTF,SUBXTTF
      REAL COXT(NPOINT),SUBXT(NPOINT)
      REAL COZ,SUBZ
```

```
      COMPLEX COHYB,SUBHYB
      COMPLEX CSIG1,CSIG2,EMPHASIS,A,B,C,D
      COMPLEX CMPLX
      COMMON /BLK1/ NPOINT,NSYM,PPSYM
      COMMON /BLK2/ ALPHA,DF
      COMMON /BLK3/ PI,TPI
      COMMON /BLK4/ START,NEWPPSYM
      COMMON /GAIN/ COGAIN,SUBGAIN
C
C          TERMINATION IMPEDANCE
C
      RS=135.0
      RM=135.0
C
C          DC RESPONSE
C
      READ(8,*) A,B
      READ(8,*) C,D
      COXT(1)=0.0
      COXT(2)=0.0
      SUBXT(1)=0.0
      SUBXT(2)=0.0
C
C          AC RESPONSE
C
       DO 20 I=1,NSYM
         READ(8,*) A,B
         READ(8,*) C,D
C
C          VOLTAGE DIVISION AT THE HYBRID AND INPUT IMPEDANCE OF THE
C          LINE AS SEEN AT THE HYBRID
C
         SUBHYB=(D*RM+B)/(A*RS+B+C*RS*RM+D*RM)
         SUBZ=REAL((D*RM+B)/(C*RM+A))
         COHYB=(A*RS+B)/(A*RS+B+C*RS*RM+D*RM)
         COZ=REAL((A*RS+B)/(C*RS+D))
C
C          NEXT IMPULSE RESPONSE
C
         CSIG1=CMPLX(COXT(2*I+1),COXT(2*I+2))
         CSIG2=CMPLX(SUBXT(2*I+1),SUBXT(2*I+2))
         EMPHASIS=CMPLX(1.0-COS(I*TPI/NSYM),SIN(I*TPI/NSYM))
         FREQ=I*DF
         COXTTF=SQRT(RM/COZ*KNEXT*FREQ**1.5)
         SUBXTTF=SQRT(RS/SUBZ*KNEXT*FREQ**1.5)
          IF((I*1.0)/NSYM.LT.(1.0-ALPHA)/2.0) THEN
            CSIG1=CSIG1*COGAIN*COHYB*COXTTF*EMPHASIS*XTSD
            CSIG2=CSIG2*SUBGAIN*SUBHYB*SUBXTTF*EMPHASIS*XTSD
            COXT(2*I+1)=REAL(CSIG1)
            COXT(2*I+2)=AIMAG(CSIG1)
            SUBXT(2*I+1)=REAL(CSIG2)
            SUBXT(2*I+2)=AIMAG(CSIG2)
          ELSEIF((I*1.0)/NSYM.GT.(1.0+ALPHA)/2.0) THEN
            COXT(2*I+1)=0.0
```

```
          COXT(2*I+2)=0.0
          SUBXT(2*I+1)=0.0
          SUBXT(2*I+2)=0.0
        ELSE
          TEMP=(1.0-SIN((PI/ALPHA)*((I*1.0)/NSYM-0.5)))/2.0
          CSIG1=CSIG1*COGAIN*COHYB*COXTTF*EMPHASIS*TEMP*XTSD
          CSIG2=CSIG2*SUBGAIN*SUBHYB*SUBXTTF*EMPHASIS*TEMP*XTSD
          COXT(2*I+1)=REAL(CSIG1)
          COXT(2*I+2)=AIMAG(CSIG1)
          SUBXT(2*I+1)=REAL(CSIG2)
          SUBXT(2*I+2)=AIMAG(CSIG2)
       ENDIF
20    CONTINUE
C
C         RESPONSE AT HIGHER FREQUENCIES IS ZERO
C
      DO 25 I=2*NSYM+3,NPOINT
        COXT(I)=0.0
25    SUBXT(I)=0.0
C
C         RESPONSE IN TIME DOMAIN
C
      CALL REALFT(COXT,NPOINT/2,-1)
      CALL REALFT(SUBXT,NPOINT/2,-1)
C
C         SCALING OF IFFT
C
      DO 30 I=1,NPOINT
       COXT(I)=COXT(I)/(NPOINT*0.5*NEWPPSYM)
30    SUBXT(I)=SUBXT(I)/(NPOINT*0.5*NEWPPSYM)
C
      CLOSE(8)
      RETURN
      END
C*********************************************************************C
C                                                                     C
C          This subroutine will calculate the echo impulse response on C
C     the central office side and the subscriber side.                C
C                                                                     C
C      coecresp = array with the C.O. echo impulse response           C
C     subecresp = array with the SUB. echo impulse response           C
C            ec = the desired echo cancellation in dB                 C
C                                                                     C
C*********************************************************************C
      SUBROUTINE ECHO(EC,COECRESP,SUBECRESP)
      INTEGER NPOINT,NSYM,PPSYM
      REAL ALPHA,DF,PI,TPI,EC,COGAIN,SUBGAIN,TEMP
      REAL RS,RM,R1,R2,R3,C2,C3
      REAL COECRESP(NPOINT),SUBECRESP(NPOINT)
      COMPLEX COEC,SUBEC,VDIV1,VDIV2,DEN,A,B,C,D
      COMPLEX CSIG1,CSIG2,EMPHASIS
      COMPLEX ZB,Y1,Y2,Y3
      COMPLEX CMPLX
      COMMON /BLK1/ NPOINT,NSYM,PPSYM
```

```
      COMMON /BLK2/ ALPHA,DF
      COMMON /BLK3/ PI,TPI
      COMMON /GAIN/ COGAIN,SUBGAIN
C
C         COMPONENT VALUES OF THE BALANCE NETWORK AND
C         TERMINATION IMPEDANCE
C
      RS=135.0
      RM=135.0
      R1=680.0
      R2=330.0
      R3=220.0
      C2=47E-9
      C3=15E-9
C
C         CONVERT EC FROM dB INTO DECIMAL
C
      EC=10**(-EC/20.0)
C
C         DC RESPONSE
C
      READ(8,*) A,B
      READ(8,*) C,D
      COECRESP(1)=0.0
      COECRESP(2)=0.0
      SUBECRESP(1)=0.0
      SUBECRESP(2)=0.0
C
C         AC RESPONSE
C
      DO 20 I=1,NSYM
      READ(8,*) A,B
      READ(8,*) C,D
C
C         THE BALANCE IMPEDANCE IS CALCULATED
C
      Y1=1.0/CMPLX(R1,0)
      Y2=1.0/CMPLX(R2,-1.0/(TPI*I*DF*C2))
      Y3=1.0/CMPLX(R3,-1.0/(TPI*I*DF*C3))
      ZB=1.0/(Y1+Y2+Y3)
C
C         HYBRID ECHO LOSS TRANSFER FUNCTION IS CALCULATED
C
      VDIV1=(D*RM+B)/(A*RS+B+C*RS*RM+D*RM)
      VDIV2=(ZB/(ZB+RS))
      SUBEC=VDIV1-VDIV2
C
      VDIV1=(A*RS+B)/(A*RS+B+C*RS*RM+D*RM)
      VDIV2=(ZB/(ZB+RM))
      COEC=VDIV1-VDIV2
C
C         ECHO IMPULSE RESPONSE
C
      CSIG1=CMPLX(COECRESP(2*I+1),COECRESP(2*I+2))
```

```
      CSIG2=CMPLX(SUBECRESP(2*I+1),SUBECRESP(2*I+2))
      EMPHASIS=CMPLX(1.0-COS(I*TPI/NSYM),SIN(I*TPI/NSYM))
        IF((I*1.0)/NSYM.LT.(1.0-ALPHA)/2.0) THEN
          CSIG1=CSIG1*COEC*EMPHASIS*EC*COGAIN
          CSIG2=CSIG2*SUBEC*EMPHASIS*EC*SUBGAIN
          COECRESP(2*I+1)=REAL(CSIG1)
          COECRESP(2*I+2)=AIMAG(CSIG1)
          SUBECRESP(2*I+1)=REAL(CSIG2)
          SUBECRESP(2*I+2)=AIMAG(CSIG2)
        ELSEIF((I*1.0)/NSYM.GT.(1.0+ALPHA)/2.0) THEN
          COECRESP(2*I+1)=0.0
          COECRESP(2*I+2)=0.0
          SUBECRESP(2*I+1)=0.0
          SUBECRESP(2*I+2)=0.0
        ELSE
          TEMP=(1.0-SIN((PI/ALPHA)*((I*1.0)/NSYM-0.5)))/2.0
          CSIG1=CSIG1*COEC*EMPHASIS*EC*COGAIN*TEMP
          CSIG2=CSIG2*SUBEC*EMPHASIS*EC*SUBGAIN*TEMP
          COECRESP(2*I+1)=REAL(CSIG1)
          COECRESP(2*I+2)=AIMAG(CSIG1)
          SUBECRESP(2*I+1)=REAL(CSIG2)
          SUBECRESP(2*I+2)=AIMAG(CSIG2)
        ENDIF
20    CONTINUE
C
C          RESPONSE AT HIGHER FREQUENCIES IS ZERO
C
      DO 25 I=2*NSYM+3,NPOINT
        COECRESP(I)=0.0
25    SUBECRESP(I)=0.0
C
C          RESPONSE IN TIME DOMAIN
C
       CALL REALFT(COECRESP,NPOINT/2,-1)
       CALL REALFT(SUBECRESP,NPOINT/2,-1)
C
C          SCALING OF IFFT
C
      DO 30 I=1,NPOINT
        COECRESP(I)=COECRESP(I)/(NPOINT/2.0)
30    SUBECRESP(I)=SUBECRESP(I)/(NPOINT/2.0)
C
      CLOSE(8)
      RETURN
      END
```

```
C*********************************************************************C
C                                                                     C
C        This subroutine will choose the best sampling phase to       C
C    minimize the precursor, or sample at the maximum value (cursor). C
C                                                                     C
C      data - input array containing the spectrum of two              C
C             isolated impulses (+ve & -ve)                           C
C     start - index when the +ve impulse is located                   C
C    method - 1 : maximize the sample value                           C
C           - 2 : maximize the ratio of sample value to               C
C                 precursor value                                     C
C    samptime - desired sample time                                   C
C                                                                     C
C*********************************************************************C
      SUBROUTINE SAMPPHASE(START,METHOD,SAMPTIME,DATA)
      INTEGER NPOINT,NSYM,PPSYM,INDEX,SAMPTIME,NUM,START,DELAY,METHOD
      REAL DATA(NPOINT)
      REAL MAX,RATIO,MAXRATIO
      COMMON /BLK1/ NPOINT,NSYM,PPSYM
       MAX=-100.0
       MAXRATIO=-1.0
C
C         ESTIMATE THE RANGE WHERE THE +VE IMPULSE SHOULD LOCATE
C         AND SEARCH THE POINT WITH THE MAXIMUM VALUE.
C
        DO 10 I=1,NPOINT*5/8
          IF(DATA(I).GT.MAX) THEN
            MAX=DATA(I)
            INDEX=I
          ENDIF
10      CONTINUE
       WRITE(6,*) 'CENTER SAMPLING POINT = ',INDEX,MAX
       DELAY=INDEX-START
       WRITE(6,*) 'DELAY = ',DELAY
C
C         IF METHOD=1 THEN THE POINT WITH MAXIMUM VALUE IS THE
C         DESIRED SAMPLING PHASE
C
C         IF METHOD=2 THEN SEARCH THE POINT WITH THE MAXIMUM RATIO
C         OF CURSOR/PRECURSOR VALUE IS THE DESIRED SAMPLING PHASE
C
        IF(METHOD.EQ.1) THEN
          SAMPTIME=INDEX
        ELSEIF(METHOD.EQ.2) THEN
          DO 15 I=1,PPSYM-1
            IF(ABS(DATA(INDEX-I)/DATA(INDEX)).LT.SQRT(.5)) THEN
              NUM=I
              GOTO 17
            ENDIF
15        CONTINUE
17          DO 20 I=0,NUM
              IF(ABS(DATA(INDEX-PPSYM-I)).GT.0) THEN
                RATIO=ABS(DATA(INDEX-I)/DATA(INDEX-I-PPSYM))
                  IF(RATIO.GT.MAXRATIO) THEN
```

```
                    MAXRATIO=RATIO
                    SAMPTIME=INDEX-I
                  ENDIF
              ELSE
                MAXRATIO=1E30
                SAMPTIME=INDEX-I
                WRITE(6,*) 'SAMPLE VALUE CLOSE TO ZERO'
              ENDIF
20          CONTINUE
C
          MAXRATIO=20*LOG10(MAXRATIO)
          WRITE(6,*) 'SAMPLING POINT TO MINIMIZE PRECURSOR = ',SAMPTIME
          WRITE(6,*) 'RATIO OF SAMPLE VALUE TO PRECURSOR VALUE = ',
     +                                           MAXRATIO,' dB'
        ELSE
          WRITE(6,*) 'ERROR IN SAMPLING PHASE'
        ENDIF
      RETURN
      END
C*********************************************************************C
C                                                                     C
C          This subroutine will estimate the numbers of front and back C
C     taps that are required to approximate an impulse response. The  C
C     area(area1) under the main cursor within a symbol period is     C
C     compared to the area under the precursor/postcursor(area2)      C
C     within a symbol period. When area2 < area1*fraction is met,     C
C     then this is the number of front/back taps that is required.    C
C                                                                     C
C     samptime = sampling instant where the main cursor is located    C
C            x = array of the impulse response                        C
C     fraction = criteria to determine the number of taps required    C
C                ( suggested value < 5e-3 )                           C
C    precursor = number of front taps that is required                C
C   postcursor = number of back taps that is required                 C
C       premax = maximum number of front taps that is allowed         C
C      postmax = maximum number of back taps that is allowed          C
C                                                                     C
C*********************************************************************C
      SUBROUTINE IMPLENGTH(SAMPTIME,PRECURSOR,POSTCURSOR,FRACTION,X,
     +                                           PREMAX,POSTMAX)
      INTEGER PRECURSOR,POSTCURSOR,PREMAX,POSTMAX
      INTEGER NPOINT,NSYM,PPSYM,INDEX,REM,SAMPTIME
      REAL X(NPOINT)
      REAL FRACTION,AREA1,AREA2
      COMMON /BLK1/ NPOINT,NSYM,PPSYM
C
C          AREA UNDER THE MAIN CURSOR
C
      AREA1=0.0
      DO 20 J=0,PPSYM-1
       INDEX=REM(SAMPTIME-J,NPOINT)
20    AREA1=AREA1+ABS(X(INDEX))
      AREA1=AREA1*FRACTION
C
```

```
C         AREA UNDER THE PRECURSOR AT I PERIOD AWAY
C
      DO 30 I=1,PREMAX
       AREA2=0.0
       DO 40 J=0,PPSYM-1
        INDEX=REM(SAMPTIME-J-I*PPSYM,NPOINT)
40     AREA2=AREA2+ABS(X(INDEX))
      IF(AREA2.LT.AREA1) THEN
        PRECURSOR=I
        GOTO 35
      ENDIF
30    CONTINUE
C
C         IF CRITERIA IS NOT MET, FRONT TAPS REQUIRED IS SET TO PREMAX
C
35    IF(PRECURSOR.EQ.0) THEN
        WRITE(6,*) 'MAY NEED MORE POINTS FOR PRECURSOR'
        INDEX=SAMPTIME-PREMAX*PPSYM
        DB=20.0*LOG10(ABS(X(INDEX)/X(SAMPTIME)))
        WRITE(6,*) 'THE RATIO OF PRECURSOR/CURSOR IS',DB,' dB DOWN '
        PRECURSOR=PREMAX
      ENDIF
C
C         AREA UNDER THE POSTCURSOR AT I PERIOD AWAY
C
      DO 50 I=1,POSTMAX
       AREA2=0.0
       DO 60 J=0,PPSYM-1
        INDEX=REM(SAMPTIME-J+I*PPSYM,NPOINT)
60     AREA2=AREA2+ABS(X(INDEX))
      IF(AREA2.LT.AREA1) THEN
        POSTCURSOR=I
        GOTO 55
      ENDIF
50    CONTINUE
C
C         IF CRITERIA IS NOT MET, BACK TAPS REQUIRED IS SET TO POSTMAX
C
55    IF(POSTCURSOR.EQ.0) THEN
        WRITE(6,*) 'MAY NEED MORE POINTS FOR POSTCURSOR'
        INDEX=SAMPTIME+POSTMAX*PPSYM
        DB=20*LOG10(ABS(X(INDEX)/X(SAMPTIME)))
        WRITE(6,*) 'THE RATIO OF POSTCURSOR/CURSOR IS', DB,' dB DOWN '
        POSTCURSOR=POSTMAX
      ENDIF
      RETURN
      END
```

```
C*********************************************************************C
C                                                                     C
C          This integer function will generate random quaternary      C
C     symbols (3,1,-1,-3). The number of symbols of each value        C
C     generated will be counted and stored in array isym.             C
C                                                                     C
C     isym = (1): counter for symbol +3                               C
C            (2): counter for symbol +1                               C
C            (3): counter for symbol -1                               C
C            (4): counter for symbol -3                               C
C                                                                     C
C*********************************************************************C
      INTEGER FUNCTION SYMGEN(ISYM)
      INTEGER ISYM(4)
      INTEGER IX,IY,IZ
      REAL RANDOM
      COMMON /RAND/ IX,IY,IZ,IDUM
      IX=MOD(171*IX,30269)
      IY=MOD(172*IY,30307)
      IZ=MOD(170*IZ,30323)
      RANDOM=AMOD(FLOAT(IX)/30269.0+FLOAT(IY)/30307.0+
     +                               FLOAT(IZ)/30323.0,1.0)
      IF(RANDOM.GE.0.5) THEN
        IF(RANDOM.GE.0.75) THEN
          SYMGEN=3
          ISYM(1)=ISYM(1)+1
        ELSE
          SYMGEN=-3
          ISYM(4)=ISYM(4)+1
        ENDIF
      ELSE
        IF(RANDOM.GE.0.25) THEN
          SYMGEN=1
          ISYM(2)=ISYM(2)+1
        ELSE
          SYMGEN=-1
          ISYM(3)=ISYM(3)+1
        ENDIF
      ENDIF
      RETURN
      END
C*********************************************************************C
C                                                                     C
C          This function will generate random number with magnitude   C
C     between 0.0 and 1.0. Set idum to any negative value to initial- C
C     ize or reinitialize the sequence.                               C
C                                                                     C
C     idum = seed to activate the random number generator (integer)   C
C                                                                     C
C*********************************************************************C
      FUNCTION RAN1(IDUM)
      DIMENSION R(97)
      PARAMETER (M1=259200,IA1=7141,IC1=54773,RM1=3.8580247E-6)
      PARAMETER (M2=134456,IA2=8121,IC2=28411,RM2=7.4373773E-6)
```

```
      PARAMETER (M3=243000,IA3=4561,IC3=51349)
      DATA IFF /0/
      IF (IDUM.LT.0.OR.IFF.EQ.0) THEN
        IFF=1
        IX1=MOD(IC1-IDUM,M1)
        IX1=MOD(IA1*IX1+IC1,M1)
        IX2=MOD(IX1,M2)
        IX1=MOD(IA1*IX1+IC1,M1)
        IX3=MOD(IX1,M3)
        DO 11 J=1,97
          IX1=MOD(IA1*IX1+IC1,M1)
          IX2=MOD(IA2*IX2+IC2,M2)
          R(J)=(FLOAT(IX1)+FLOAT(IX2)*RM2)*RM1
11      CONTINUE
        IDUM=1
      ENDIF
      IX1=MOD(IA1*IX1+IC1,M1)
      IX2=MOD(IA2*IX2+IC2,M2)
      IX3=MOD(IA3*IX3+IC3,M3)
      J=1+(97*IX3)/M3
      IF(J.GT.97.OR.J.LT.1)PAUSE
      RAN1=R(J)
      R(J)=(FLOAT(IX1)+FLOAT(IX2)*RM2)*RM1
      RETURN
      END
C***********************************************************************C
C                                                                       C
C         This function will return a normally distributed deviate     C
C     with zero mean and unit variance, using RAN1(IDUM) as the source C
C     of uniform deviates.                                              C
C                                                                       C
C     idum = seed to initiate the function (integer)                    C
C                                                                       C
C***********************************************************************C
      FUNCTION GASDEV(IDUM)
      DATA ISET/0/
      IF (ISET.EQ.0) THEN
1       V1=2.*RAN1(IDUM)-1.
        V2=2.*RAN1(IDUM)-1.
        R=V1**2+V2**2
        IF(R.GE.1.)GO TO 1
        FAC=SQRT(-2.*LOG(R)/R)
        GSET=V1*FAC
        GASDEV=V2*FAC
        ISET=1
      ELSE
        GASDEV=GSET
        ISET=0
      ENDIF
      RETURN
      END
```

```
C*********************************************************************C
C        Modulo j operation except that when rem=0, rem is set to j   C
C        i cannot equal to zero  .                                     C
C*********************************************************************C
      INTEGER FUNCTION REM(I,J)
      REM=I-(INT(I/J)*J)
       IF(REM.EQ.0) REM=J
      RETURN
      END
C*********************************************************************C
C                                                                     C
C         This function subprogram will decide the symbol received   C
C         by comparing to the threshold level. The threshold levels  C
C         are -2, 0, and +2.                                          C
C                                                                     C
C         vin = the received signal level                             C
C                                                                     C
C*********************************************************************C
      INTEGER FUNCTION DECISION(VIN)
      REAL VIN
       IF(VIN.GT.2) THEN
         DECISION=3
       ELSEIF(VIN.LE.-2) THEN
         DECISION=-3
       ELSEIF(VIN.GT.0.AND.VIN.LE.2) THEN
         DECISION=1
       ELSEIF(VIN.LE.0.AND.VIN.GT.-2) THEN
         DECISION=-1
       ELSE
         STOP 'ERROR IN SUBROUTINE DECISION '
       ENDIF
      RETURN
      END
C*********************************************************************C
C                                                                     C
C         This subroutine will keep track of the number of symbol    C
C     errors and bit errors.                                          C
C                                                                     C
C       x = the difference between the ideal symbol and               C
C           detected symbol                                           C
C     ber = (1): counter for bit errors                               C
C           (2): counter for symbol errors                            C
C                                                                     C
C*********************************************************************C
      SUBROUTINE BITSYMERROR(X,BER)
      INTEGER X
      INTEGER BER(2)
       IF(IABS(X).EQ.4) THEN
         BER(1)=BER(1)+2
         BER(2)=BER(2)+1
       ELSEIF(IABS(X).EQ.2) THEN
         BER(1)=BER(1)+1
         BER(2)=BER(2)+1
       ELSEIF(IABS(X).EQ.6) THEN
```

```
          BER(1)=BER(1)+1
          BER(2)=BER(2)+1
        ELSE
          STOP 'ERROR IN SUBROUTINE BITSYMERROR'
        ENDIF
      RETURN
      END
C**********************************************************************C
C                                                                      C
C          Calculate the Fourier transform of a set of 2n real-valued  C
C     data points. Replace this data (stored in data) by the positive  C
C     frequency half of its complex Fourier transform. The real-valued C
C     first and last components of the complex transform are returned  C
C     as elements data(1) and data(2) respectively. N must be a power  C
C     of 2. This routine also calculate the inverse transform of a     C
C     complex data array if it is the transform of real data. (result  C
C     in this case must be multiplied by 1/n)                          C
C                                                                      C
C     isign =1: fft   -1: ifft                                         C
C                                                                      C
C**********************************************************************C
      SUBROUTINE REALFT(DATA,N,ISIGN)
      REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
      INTEGER N
      DIMENSION DATA(2*N+2)
      THETA=-6.28318530717959D0/2.0D0/DBLE(N)
      WR=1.0D0
      WI=0.0D0
      C1=0.5
      IF (ISIGN.EQ.1) THEN
        C2=-0.5
        CALL FOUR1(DATA,N,-1)
        DATA(2*N+1)=DATA(1)
        DATA(2*N+2)=DATA(2)
      ELSE
        C2=0.5
        THETA=-THETA
        DATA(2*N+1)= DATA(2)
        DATA(2*N+2)=0.0
        DATA(2)=0.0
      ENDIF
      WPR=-2.0D0*DSIN(0.5D0*THETA)**2
      WPI=DSIN(THETA)
      N2P3=2*N+3
      DO 11 I=1,N/2+1
        I1=2*I-1
        I2=I1+1
        I3=N2P3-I2
        I4=I3+1
        WRS=SNGL(WR)
        WIS=SNGL(WI)
        H1R=C1*(DATA(I1)+DATA(I3))
        H1I=C1*(DATA(I2)-DATA(I4))
        H2R=-C2*(DATA(I2)+DATA(I4))
```

```
          H2I=C2*(DATA(I1)-DATA(I3))
          DATA(I1)=H1R+WRS*H2R-WIS*H2I
          DATA(I2)=H1I+WRS*H2I+WIS*H2R
          DATA(I3)=H1R-WRS*H2R+WIS*H2I
          DATA(I4)=-H1I+WRS*H2I+WIS*H2R
          WTEMP=WR
          WR=WR*WPR-WI*WPI+WR
          WI=WI*WPR+WTEMP*WPI+WI
11      CONTINUE
        IF (ISIGN.EQ.1) THEN
          DATA(2)=DATA(2*N+1)
        ELSE
          CALL FOUR1(DATA,N,1)
        ENDIF
        RETURN
        END
C**********************************************************************C
C                                                                      C
C       fft subroutine                                                 C
C                                                                      C
C        data = array to be transformed                                C
C          nn = (dimension of data)/2                                  C
C       isign = -1: fft    1:ifft                                      C
C                                                                      C
C**********************************************************************C
        SUBROUTINE FOUR1(DATA,NN,ISIGN)
        REAL*8 WR,WI,WPR,WPI,WTEMP,THETA
        DIMENSION DATA(*)
        N=2*NN
        J=1
        DO 11 I=1,N,2
          IF(J.GT.I)THEN
            TEMPR=DATA(J)
            TEMPI=DATA(J+1)
            DATA(J)=DATA(I)
            DATA(J+1)=DATA(I+1)
            DATA(I)=TEMPR
            DATA(I+1)=TEMPI
          ENDIF
          M=N/2
1         IF ((M.GE.2).AND.(J.GT.M)) THEN
            J=J-M
            M=M/2
          GO TO 1
          ENDIF
          J=J+M
11      CONTINUE
        MMAX=2
2       IF (N.GT.MMAX) THEN
          ISTEP=2*MMAX
          THETA=6.28318530717959D0/(ISIGN*MMAX)
          WPR=-2.D0*DSIN(0.5D0*THETA)**2
          WPI=DSIN(THETA)
          WR=1.D0
```

```
        WI=0.DO
        DO 13 M=1,MMAX,2
          DO 12 I=M,N,ISTEP
            J=I+MMAX
            TEMPR=SNGL(WR)*DATA(J)-SNGL(WI)*DATA(J+1)
            TEMPI=SNGL(WR)*DATA(J+1)+SNGL(WI)*DATA(J)
            DATA(J)=DATA(I)-TEMPR
            DATA(J+1)=DATA(I+1)-TEMPI
            DATA(I)=DATA(I)+TEMPR
            DATA(I+1)=DATA(I+1)+TEMPI
12        CONTINUE
          WTEMP=WR
          WR=WR*WPR-WI*WPI+WR
          WI=WI*WPR+WTEMP*WPI+WI
13      CONTINUE
        MMAX=ISTEP
      GO TO 2
      ENDIF
      RETURN
      END
C**********************************************************************C
C                                                                      C
C       Generate eye diagram by importing samples of the traces into  C
C     LOTUS 123.                                                       C
C                                                                      C
C      offset = number of samples to be shifted                        C
C    timebase = number of periods to be plotted                        C
C     totalpt = dimension of array A                                   C
C      ptpert = number of points per period                            C
C                                                                      C
C**********************************************************************C
      SUBROUTINE EYE(OFFSET,TIMEBASE,TOTALPT,PTPERT,A)
      INTEGER OFFSET,TIMEBASE,TOTALPT,PTPERT,TRACEPT,MTRACE,TEST,REM
      REAL A(TOTALPT)
      TRACEPT=TIMEBASE*PTPERT
      MTRACE=TOTALPT/TRACEPT+1
       DO 20 I=1,MTRACE
         DO 30 J=0,TRACEPT
          TEST=(I-1)*TRACEPT+J+OFFSET
   30     WRITE(3,100) (J*1.0)/TRACEPT,A(REM(TEST,TOTALPT))
        WRITE(3,100)  1.0, -9.0
        WRITE(3,100)  0.0, -9.0
   20  CONTINUE
  100 FORMAT(F8.5,F8.5)
      CLOSE(3)
      RETURN
      END
```

```
C*********************************************************************C
C                                                                     C
C          This routine will calculate the eye opening of the eye    C
C     diagram. The thresholds are at -2, 0, and 2 Volts.              C
C                                                                     C
C        sym = detected symols                                        C
C     desire = ideal symbols                                          C
C       nsym = dimension of array sym and desire                      C
C         sw = switch to select the direction of transmission         C
C              C.O -> Sub.  or  Sub. -> C.O.                          C
C**********************************************************************
      SUBROUTINE EYEOPEN(SYM,DESIRE,NSYM,SW)
      INTEGER NSYM,SW,NERR,DECISION
      REAL MIN3,MIN1,MAX1,MINN1,MAXN1,MAXN3,EYETOP,EYEMID,EYEBOT,AVG
      INTEGER*1 DESIRE(NSYM)
      REAL SYM(NSYM)
C
      MIN3=100.0
      MIN1=100.0
      MINN1=100.0
      MAX1=-100.0
      MAXN1=-100.0
      MAXN3=-100.0
      NERR=0
C
       DO 10 I=1,NSYM
         IF(DESIRE(I).EQ.DECISION(SYM(I))) THEN
           IF(DESIRE(I).EQ.3) THEN
             MIN3=MIN(MIN3,SYM(I))
           ELSEIF(DESIRE(I).EQ.1) THEN
             MAX1=MAX(MAX1,SYM(I))
             MIN1=MIN(MIN1,SYM(I))
           ELSEIF(DESIRE(I).EQ.-1) THEN
             MAXN1=MAX(MAXN1,SYM(I))
             MINN1=MIN(MINN1,SYM(I))
           ELSEIF(DESIRE(I).EQ.-3) THEN
             MAXN3=MAX(MAXN3,SYM(I))
           ELSE
             STOP 'ERROR IN EYE OPENING CALCULATION'
           ENDIF
         ELSE
           NERR=NERR+1
         ENDIF
10    CONTINUE
      EYETOP=(MIN3-MAX1)*50.0
      EYEMID=(MIN1-MAXN1)*50.0
      EYEBOT=(MINN1-MAXN3)*50.0
      AVG=(EYETOP+EYEMID+EYEBOT)/3.0
       IF(SW.EQ.1) THEN
         WRITE(1,*) 'CENTRAL OFFICE :'
       ELSE
         WRITE(1,*) 'SUBSCRIBER SIDE :'
       ENDIF
      WRITE(1,*)
```

```
WRITE(1,*) '    NUMBER OF ERRORS = ',NERR
WRITE(1,*) '      TOP EYE OPENING = ',EYETOP
WRITE(1,*) '   MIDDLE EYE OPENING = ',EYEMID
WRITE(1,*) '   BOTTOM EYE OPENING = ',EYEBOT
WRITE(1,*) ' AVERAGE EYE OPENING = ',AVG
WRITE(1,*)
RETURN
END
```