VALIDATION AND ENHANCEMENT OF TELETEXT SYSTEM SIMULATION FINAL REPORT

MCS File No: 8526
DSS File No: 12ST.36001-4-3095
DSS Contract No: OST84-00458
Date: March 18, 1985

Prepared by: R. M. Armstrong, M. Moher

K. W. Moreland
K. W. Moreland

Approved by: S. Crozier

SUBMITTED BY:

MCS MILLER COMMUNICATIONS SYSTEMS LTD.
300 Legget Drive,
Kanata, Ontario,
Canada K2K 1Y5

(2)

VALIDATION AND ENHANCEMENT OF TELETEXT SYSTEM SIMULATION, FINAL REPORT

MCS File No: 8526
DSS File No: 12ST.36001-4-3095
DSS Contract No: OST84-00458
Date: March 18, 1985

Prepared by: M. Moher.
(1) R. M. Armstrong, M. Moher

K. W. Moreland
K. W. Moreland

Approved by:
S. Crozier

SUBMITTED BY:

MCS MILLER COMMUNICATIONS SYSTEMS LTD.
300 Legget Drive,
Kanata, Ontario,
Canada K2K 1Y5

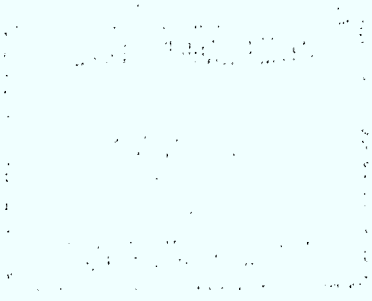


TABLE OF CONTENTS

1.0 INTRODUCTION

This report is a summary of the work carried out to enhance and validate the existing teletext simulation system. The existing simulation system provided an end to end channel simulation of a Telidon channel including a coding analysis of the output data stream. This simulation was developed under a previous contract by MCS using the CRC Honeywell CP-6 computer.

The present work was broken down into five tasks, and these five tasks provide a general outline of the report. The first task was the transfer of the existing software from the CRC CP-6 computer to the MCS VAX-11/750 computer. This was accomplished and the program was further enhanced by coding computationally demanding parts to use the MCS FPS-5105 array processor. The VAX/array processor combination provided a factor of approximately fifteen improvement in execution time over the original CP-6 version.

The second task was a validation of the coding analysis. The proposed coding technique for the Telidon signal format was developed at the Department of Mathematics and Statistics of Carleton University. This task included a subcontract with Dr. Brian Mortimer to verify that the proposed coding techniques had been correctly implemented in software. A comparison of predicted performance with simulation results over ideal channels provided proof that the software implementation was correct.

The third task was the development and simulation of a suitable adaptive slicing level algorithm. Such an adaptive slicer was developed but the instances in which a noticeable improvement in performance is obtained are

limited. The reason being that the instances where the slicing level is affected form only a subset of the possible multipath channels. It is only on this subset that performance can be improved by this method.

The fourth task was the development and simulation of an improved symbol synchronization scheme. An near optimal bit timing recovery scheme was suggested and implemented in software. This technique was shown to have better performance than existing techniques and a possible hardware implementation was suggested.

The fifth task was the incorporation of measured characteristics of various components of the teletext system in the simulation and to validate the simulation based on these measurements. This task was limited because of the limited amount and quality of the data provided and although there are some differences between simulation and measured results, there are also quite reasonable explanations for these differences.

In general, the software simulation was shown to be valid, and the incorporation of a new bit timing recovery scheme and slicing algorithm have illustrated some ways in which the existing teletext system can be improved.

2.0 SOFTWARE TRANSFER AND MODIFICATIONS

The software simulation of the Telidon system was originally written for use on a Honeywell CP-6 computer sited at CRC Ottawa. The source code of this software, as well as some data files containing filter specifications and simulation results, were transferred in January.

The facilities available at MCS include a DEC VAX/750 serving as host to a FPS-5105 Array Processor. Once minor changes to the existing software had been made to accommodate the different I/O conventions of the DEC equipment and the differences in the FORTRAN compilers used (see Section 2.1 below), a modified version of the software was produced which used the Array Processor to perform the most time-consuming of the simulation calculations.

This modified version of the simulation software was on the order of sixty times faster than the VAX version of the same software, and 15 times faster than the CP-6. This saving in execution time made it possible to perform a much larger number of simulations than could otherwise have been contemplated. The deliverable software will include both this AP version of the software and a VAX-only version.

2.1 Changes to the Host Program

Most of the changes made to the version of the simulation utilizing the host alone were necessitated by the different I/O conventions and file-naming conventions of the Honeywell and DEC equipment. The most important of these is that Version 3.7 of VMS allows file names to have at most 13 characters (nine characters, followed by '.', followed by a three-character extension), while the CP-6 operating system allowed much longer file names, and

allowed non-alphanumeric characters to be used in these names. Additionally, the FORTRAN-77 compiler used on the VAX required several parameters to be present in OPEN statement (such as file type), which the CP-6 compiler did not demand. These changes, as well as some minor changes to READ and WRITE statements, and changes in values used as logical I/O unit numbers, were made easily, and did not affect the operational characteristics of the simulation.

2.2 Array Processor Utilization

The Array Processor greatly increases the speed of repetitive computations performed upon large vectors of values. For example, replacing the usual method of multiplying corresponding elements of matrices through use of a DO-loop construct with a call to the VMUL subroutine supplied in the AP math library typically cuts computation time by an order of magnitude or more. The execution time improvements associated with more complex operations, such as Fast Fourier Transforms, are even more dramatic.

The two disadvantages of the AP when compared to the VAX are (1) its inability to do disk I/O directly, and (2) the somewhat more primitive nature of the operations available on it. The first is a minor consideration; values can be transferred to and from the VAX host for output to disk, or for printing, with a minimum of difficulty. To keep execution times to a minimum, however, minimizing the number of such transfers performed is desirable. In aid of this, intermediate values calculated during the execution of the program which need be output are left in the AP during program execution. This practice makes debugging the AP version of the program somewhat more difficult, but the increase in software development time was more than

compensated for by the improved execution times noted above.

The second point forced a slight change in programming style. Vector values in the AP are referred to by the Advanced Math Library routines supplied by FPS by their base address and increment values. Throughout the AP version of the simulation these base addresses are given symbolic names (i.e. LSIGNAL refers to the location in AP memory of the base address of the SIGNAL vector). A brief summary of the Advanced Math Library routines primarily used in this simulation is provided in Table 2.1.

2.3 Summary of New Features

Three major functional changes, and several minor ones, have been made to the simulation. The first, and most important, is the implementation of a new decoding scheme. A new bit-timing recovery scheme, and a new slicing level determination scheme, are included in this. The correlation method of bit-timing recovery has also been implemented in the new simulation, as has an adaptive slicing level determination.

Among the less important alterations, a third filter file format has been included. Setting the filter file format specifier to 2 causes the filter values corresponding to each frequency to be read from the file as complex notation (i.e. the three real numbers in each record of the filter file are read as frequency in MHz and real and imaginary components of the complex gain at that frequency). Also, a separation of random number generation seeds in the main program body has been effected, so that the data sequence generated for a given random seed is unique, and unaffected by the selection of bit-timing recovery method. This

TABLE 2.1 : FPS ADVANCED MATH LIBRARY SUBROUTINES USED

ROUTINE	RESULT
APGET	Gets a vector from the AP memory locations specified
APPSEL	Selects which page in AP memory subsequency operations will be performed on
APPUT	Puts a vector into the AP memory locations specified
APWD/APWR/ APWAIT	Timing routine required to ensure synchronization of processing between host and AP
BVMOV	Moves a vector from one page of memory to another
CFFT	Performs forward or inverse FFT on the complex vector specified
CFFTSC	Performs scaling on a complex vector after FFT operations
CVADD	Adds the two complex vectors specified
CVMUL	Multiplies the two complex vectors specified
RECT	Converts the complex vector values specified to rectangular form
VCLR	Fills the vector specified with zero values
VFIX	Converts the real vector specified to integer
VFLT	Converts the integer array specified to real values
VLIM	Clips a vector to specified upper and lower bounds
VLN	Performs the natural logarithm operation on the vector specified
VMOV	Moves a vector from one location within a page to another location in the same page
VMUL	Multiplies the two real vectors specified
VNEG	Negates the vector specified
VRAND	Generates a vector of random numbers uniformly distributed on [0.,1.]
VREAL	Extracts real parts of a complex vector
VSMSA	Multiplies vector elements by one scalar and adds a second scalar to each value (linear transformation).
VSMUL	Multiplies vector elements by a scalar
VSQRT	Performs the square root operation on the vector specified
VSUB	Subtracts one real vector from another

change was made to allow direct comparison of simulation runs performed with different bit-timing receovery schemes.

2.4 Support Programs

Several independent programs were converted or created to aid in simulation and in the analysis of simulation results.

2.4.1 SUPERPLOT

A sophisticated plotting utility, intended for use with a HP7470A plotter, was implemented. This program runs interactively, communicating with the user through a series of screen panels, and was used to plot such things as error rate curves and Telidon line signals.

2.4.2 CODING

This program is equivalent to the MCODING program created for the previous contract.

2.4.3 COMPRESS

This program, converted from the version written for the previous contract, takes error sequences generated by the TELSIM program and places them in a compressed format for use with the CODING program.

2.4.4 TAPEREAD

This program was written to convert the error data supplied by CRC from field tests into the compressed format required for the CODING program.

2.4.5 SUPPLY

This program takes a measured impulse response characteristic stored in a disk file, as well as a pulse shaping characteristic also stored ina disk file, and factors the former out of the latter to create an output disk file which can then be used by TELSIM as an input filter file desicription.

2.4.6 MC

This program separates the effects of word synchronization loss on the error rate expressed by a TELSIM error listing file. The program also averages error results from different simulation runs performed with the same data sequence to create 95% confidence intervals for the actual error rates.

2.4.7 EYETEL

This is a modified version of the TELSIM program, used to create eye diagrams. A single noiseless signal is produced in accordance with the parameters contained in a user-specified parameter file, and this burst used to construct a data file which is subsequently read by MAKEYE.

2.4.8 MAKEYE

This program plots an eye diagram based upon the values stored in the file created by EYETEL.

3.0 CODING

The validation of the coding analysis program has been separated into the following subtasks:

(a) Review of the coding analysis program by Dr. B.C. Mortimer;

(b) comparison of simulation results to theoretical results;

(c) evaluation of the performance of the proposed codes for computer simulated and field measured error sequences;

(d) possible explanation of the coding performance trends (this will include a rough characterization of the simulation runs).

Dr. Mortimer's contributions to the report may be found in Appendix A. The theoretical results of this appendix are used in the validation results that follow. Appendix B contains a pseudo-code description of the decoding algorithm implemented for the Code C and Bundle decoder.

3.1 Code Description and Terminology

3.1.1 The Codes

The coding analysis program evaluates the performance of 5 codes, viz:

1) Byte Parity
2) Product Code
3) Carleton 1-byte Code

4) Code C
5) Bundle code (encoded with Code C)

Before describing the codes, a brief description of the packet structure will be given

(i) Packet Structure

A line of Teliдon information is shown in Figure 3.1. The first three bytes are for synchronization purposes, therefore, this portion of the line is pertinent only to the 'Telidon channel', not the 'Telidon decoder'. At the beginning of each line, there are line prefix bytes and a variable number of header bytes. All of these prefix and header bytes have been specified to be Hamming (8, 4) encoded [6], thus enabling single bit/byte error correction and detection of 2 and possibly more bit errors in a byte. The prefix identifies the packet by way of a three byte address. It also indicates the structure of the remaining packet; for instance, the number of suffix bytes to be used is indicated in the prefix.

All bytes following the prefix, including the optional suffix bytes and encoded header bytes, form the data block. This block consists of 25 to 28 bytes. The last one, two or three bytes are optionally taken up as suffix bytes, used for error detection and correction.

The codes under investigation operate on the data block. The effect of the double encoding on the header bytes is not considered. For notational purposes we shall denote the prefix bytes, B_1 through B_5 and the data bytes, B_6 through B_{33}. The suffix bytes will be treated as data bytes, in that an error in the suffix bytes will count as an error in the data.

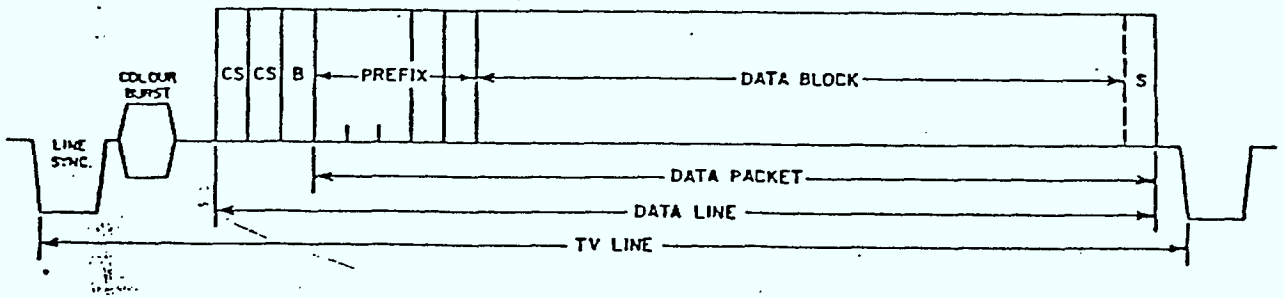


Figure 3.1 Packet Structure.

The codes proposed for the data block will now be described.

a) Byte Parity

A requirement under BS-14 [6] is that all of the bytes in the data block must have odd parity. The parity check bits allow for the detection of any error pattern that introduces an odd number of errors into a data byte. Byte parity alone has no error correction capabilities and it represents the minimum amount of coding allowed.

b) Product Code

The Product code is simply a row column parity check. It is capable of correcting one bit error in the 28 byte data block, and detecting any odd number of errors/byte or any error pattern confined to one byte.

The suffix byte, B_{33}, for the product code is defined as:

$$B_{33} = \sum_{i=6}^{32} B_i$$

The suffix byte is used to check column parity, while the row parity check is handled by the odd byte parity. The Product code was proposed for the Telidon system by Sablatash & Storey and is clearly described in [7].

(c) Carleton 1-byte code

The Carleton 1-byte code has the same correction capabilities as the Product code, however, it has greater error detection capabilities. The suffix byte, B_{33}, is given by

$$B_{33} = \alpha^{117} \sum_{i=6}^{32} B_i \alpha^{B_i}$$

where α^j is an element in a Galois field specified by the generator polynomial

$$x^8 + x^7 + x^4 + x^3 + x + 1 = 0$$

and the elements

α^0	00000001
α^1	00000010
α^2	00000100
α^3	00001000
α^4	00010000
α^5	00100000
α^6	01000000
α^7	10000000

A detailed description of the code is given in [8].

(d) Code C

The Code C is a combination of the Carleton one byte code and the Product code. It is capable of correcting any single byte error and any 2 byte

erasures (parity errors). The packet is encoded such that B_{33} meets the Product code requirements (1), and B_{32} is choosen such that B_{33} also meets the Carleton 1-byte codes requirements. This is accomplished as follows:

$$\text{Let } P_p = \sum_{i=6}^{31} B_i \ ,$$

$$P_c = \alpha^{117} \sum_{i=6}^{32} B_i \alpha^{B_i} \ ,$$

then we wish B_{32} and B_{33} such that:

$$P_c = \alpha^{117} B_{32} \alpha^{8(32)} \Big|_{\text{mod } 127} = B_{33}$$

$$P_p + B_{32} = B_{33}$$

Solving for B_{32} gives

$$B_{32} = \alpha^{14}(P_c + P_p)$$

A detailed description of the code is provided in [9].

(e) Bundle Code

A number of possible bundle strategies have been suggested for Telidon [9]. The approach considered here is that recommended by Mortimer and Moore [9]. A bundle of the Bundle code consists of 14 data blocks (packets). Thirteen of these are identical to packets encoded with the Carleton 2-byte code. The 14th packet contains 2 byte suffixes for packets formed by interleaving the bytes of the previous 13 data blocks. (See Figure 3.2). Suffix bytes in the 14th block are calculated on the basis of the 14th block alone. The choice of 14 lines (13 data lines and one line of vertical suffixes) allows the vertical codewords to be Code C codewords. The details of the Bundle code structure are given in [9].

3.1.2 The Terminology

Packet - A line of teletext data composed of 33 bytes

Prefix Bytes - The first 5 bytes in a packet. These bytes are encoded with a Hamming [8, 4] code.

Data Bytes - The last 28 bytes in a packet. These bytes include both the data and any bytes used for coding purposes.

Prefix Code - Hamming [8, 4] code applied to the prefix bytes.

Data Code - A generic term used to indicate the code applied to the data bytes.

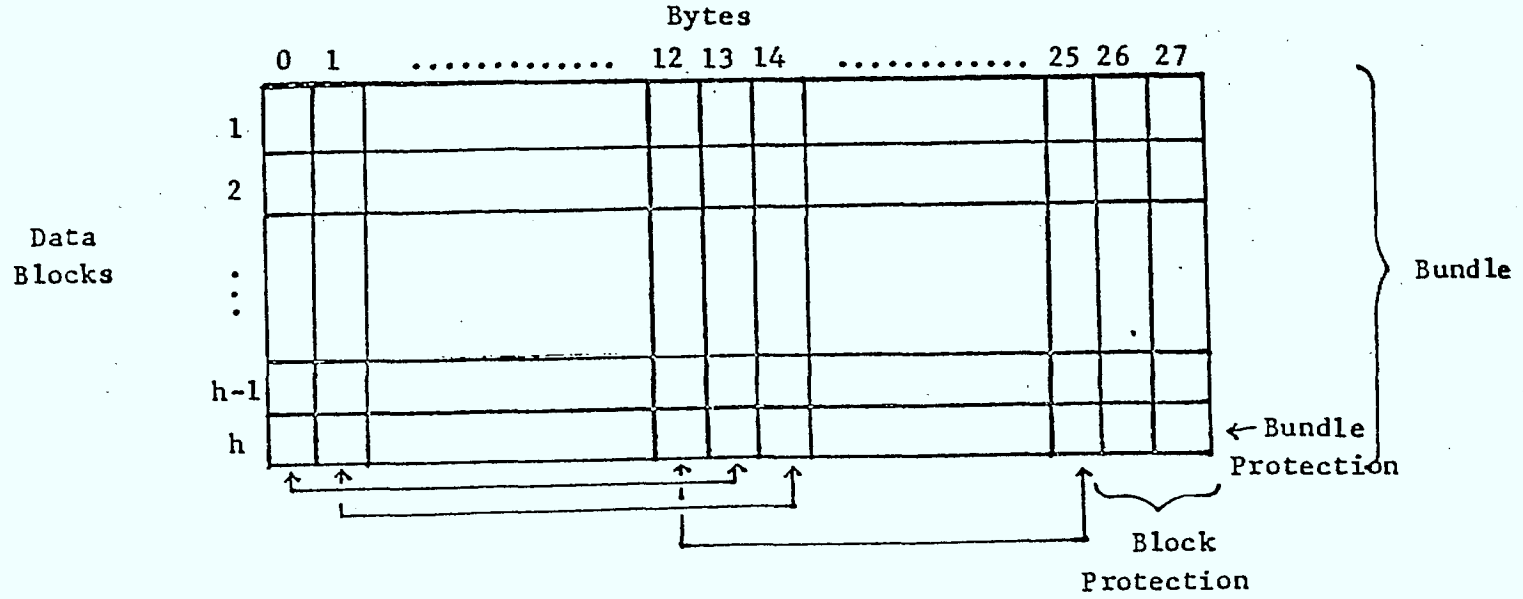


Figure 3.2 Bundle Structure.

Decoding Error (DE)	- An error condition in which the decoder either incorrectly decodes a packet or perceives an erroneous packet as error free.
Decoding Failure (DF)	- A detectable but not correctable error condition.
Correct Decoding (CD)	- An error condition (including the error-free condition) that results in an error-free packet after decoding.
Rejection Rate	- The probability of a decoding failure, P(DF).
Rejection	- A term used to indicate that a decoding failure has occured.
Corruption	- A term ised to indicate that a decoding error has occurred.
Smeared Packet	- A packet where greater than 30% of the bits are in error. This usally results when word-sync is lost in the decoder.
Output Bit Error Rate	- The total number of bit errors in the packets not rejected by the data and prefix codes, divided by the total number of bits in those packets.
P(x)	- The probability of the event x.

3.2 Validation Against Theory

In this section a comparison is made between the performance of the Product code, Code C and the Bundle code calculated by the coding evaluation program and the performance predicted by theory. To make a comparison, a program was written which generated independent errors at a desired probability of error. These error sequences were then used as input to the coding analysis program and the results were compared to theory.

For theoretical purposes, the most common measures of a codes performance are:

P(DF) - Probability of a decoding failure (Also known as the Rejection Rate)

P(DE) - Probability of a decoding error

P(CD) - Probability of correct decoding.

Note that for any coding scheme these three event form the set of all possible outcomes of decoding, hence,

$$P(DF) + P(DE) + P(CD) = 1. \qquad (3\text{-}1)$$

The coding analysis program evaluates the probability of a decoding falure and the Output Bit Error Rate (among other parameters). The Output Bit Error Rate (OBER) may be related to the above indices, by the following equation:

$$OBER = \frac{\sum_{i=1}^{\#\text{ of Bits/Codeword}} P_i(DE)}{1 - P(DF)} * \frac{i}{\#\text{ of Bits/Cordword}} \qquad (3\text{-}2)$$

where $P_i(DE)$ denotes the probability of a decoding error which results in i bit errors.

A summary of the results of this comparison follows.

(a) Product Code

The Product code is a single error correction code. The probability of correct decoding is given by

$$P(CD) = q^n + n*p*q^{(n-1)} \qquad (3-3)$$

where n is the number of bits in a codeword (n = 224 for the Telidon system). To determine the probability of a decoding error, the weight distribution of the code may be used. The weight distribution of the Product code was calculated in [8] and is reproduced below

	A_3	A_4	A_5	A_6	A_7
Product Code	0	10,584	0	1,100,736	0

where A_i denotes a codeword of weight i.

In [10] an expression for the probability of a decoding error, for a single error correcting code, is given as follows:

$$P(DE) = \sum_{k=2}^{n+1} ((k+1)A_{k+1} + A_k + (n-k+1)A_{k-1})p^k q^{n-k} \qquad (3-4)$$

rewriting

$$P(DE) = \sum_{}^{n+1} (k+1)A_{k+1}p^k q^{n-k} + \sum_{k=2}^{n+1} A_k p^k q^{n-k} + \sum_{k=2}^{n+1} (n-k+1)A_{k-1}p^k q^{n-k} \qquad (3-5)$$

The first summation in equation (3-5) is due to errors of weight k, where all k errors are coincident with the non-zero elements in a codeword of weight k+1. After Decoding the error sequence will be mapped into an error sequence of weight k+1.

The second summation in equation (3-5) corresponds to an error pattern of weight k that is also a codeword of weight k. In this case the decoder does not even detect the existence of the error. After decoding there will be k errors.

The final term in this equation corresponds to an error pattern of weight k, where k-1 of the errors are coincident with the non-zero elements in a codeword of weight k-1. The decoder will correct one of the k errors, resulting in k-1 errors after decoding.

Making note of the above, the probability of a decoding error which results in i bit errors after decoding is given by

$$P_i(DE) = A_i p^i q^{n-i} + i\, A_i p^{i-1} q^{n-i+1} + (n - i) A_i p^{i+1} q^{n-i-1} \qquad (3\text{-}6)$$

and the probability of a decoding failure may be determined using equations (3-1), (3-3) and (3-6).

Using the above equalities, the performance of the product code was calculated. A comparison of the performance of the different performance indices predicted by the above parameters, and the results

obtained using the evaluation program may be found in Figures 3.3 and 3.4. From these figures it is clear that the evaluation program is functioning correctly for the product code.

(b) Code C

In the Appendix A, the required probabilities are detemined by considering the contribution of error patterns of increasing weights until the contribution becomes negligible. This process is carried out for the decoding error and decoding failure conditions. The probability of correct decoding is then determine from these values. For the benchmark in this section, the calculation has been slightly altered. The alteration is simply that P(CD) is calculated explicitly

$$P(CD) = (q^8)^{28} + \binom{28}{1}(q^8)^{27}p_1 + \binom{28}{2}(q^8)^{26}p_2^2 \tag{3-7}$$

where $p_1 = 1 - q^8$,

$$p_2 = \binom{2}{1}q^7p + \binom{8}{3}q^5p^3 + \binom{8}{5}q^3p^5 + \binom{8}{7}p^7q$$

and using the P(DE) from the Appendix A, the P(DF) is determined. The reason for this alteration is that the bound on the probability of the decoding failure is not that tight until a great number of error patterns have been considered. In effect all that has been done is to tighten the bound of the P(DF).

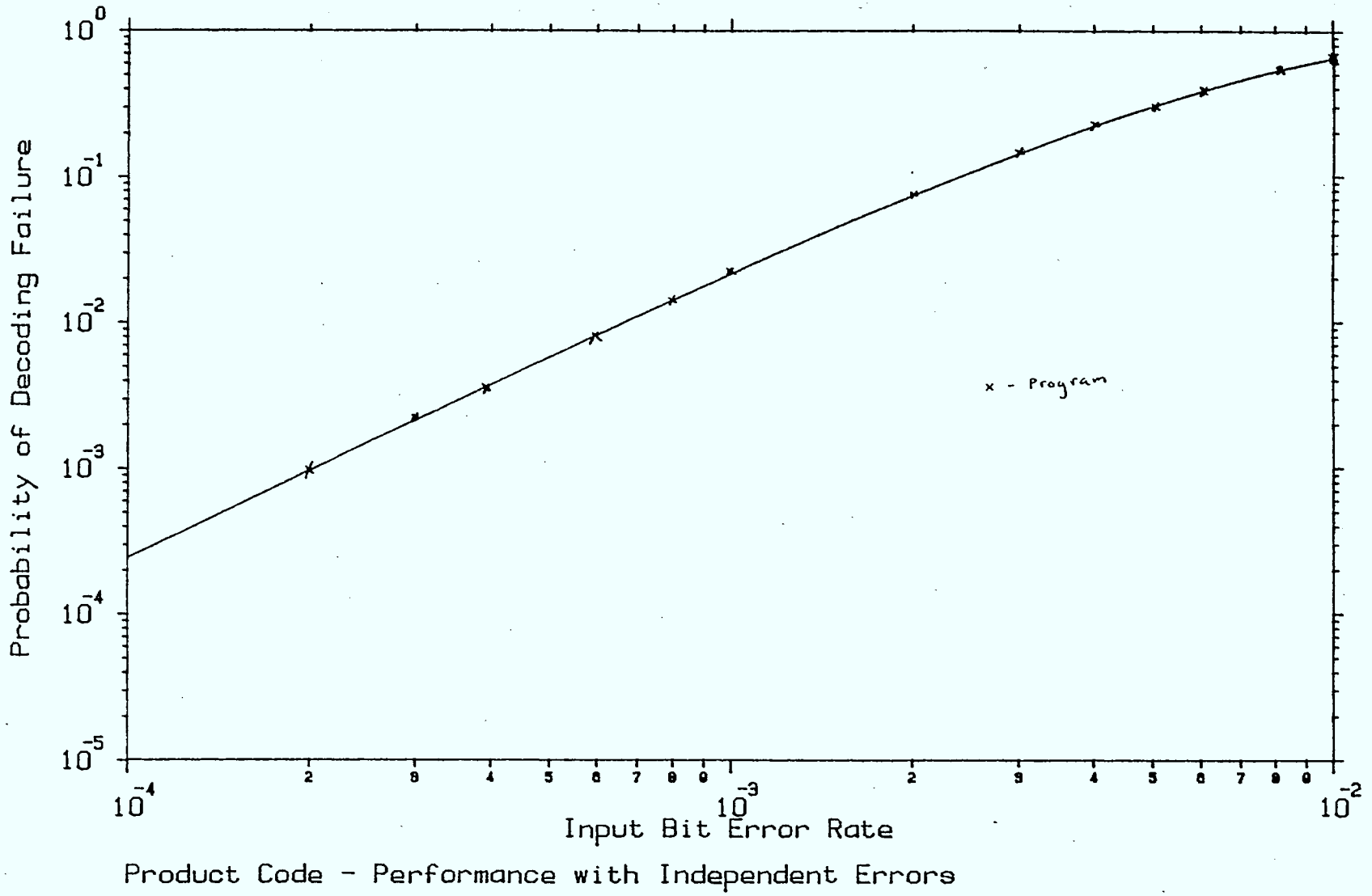
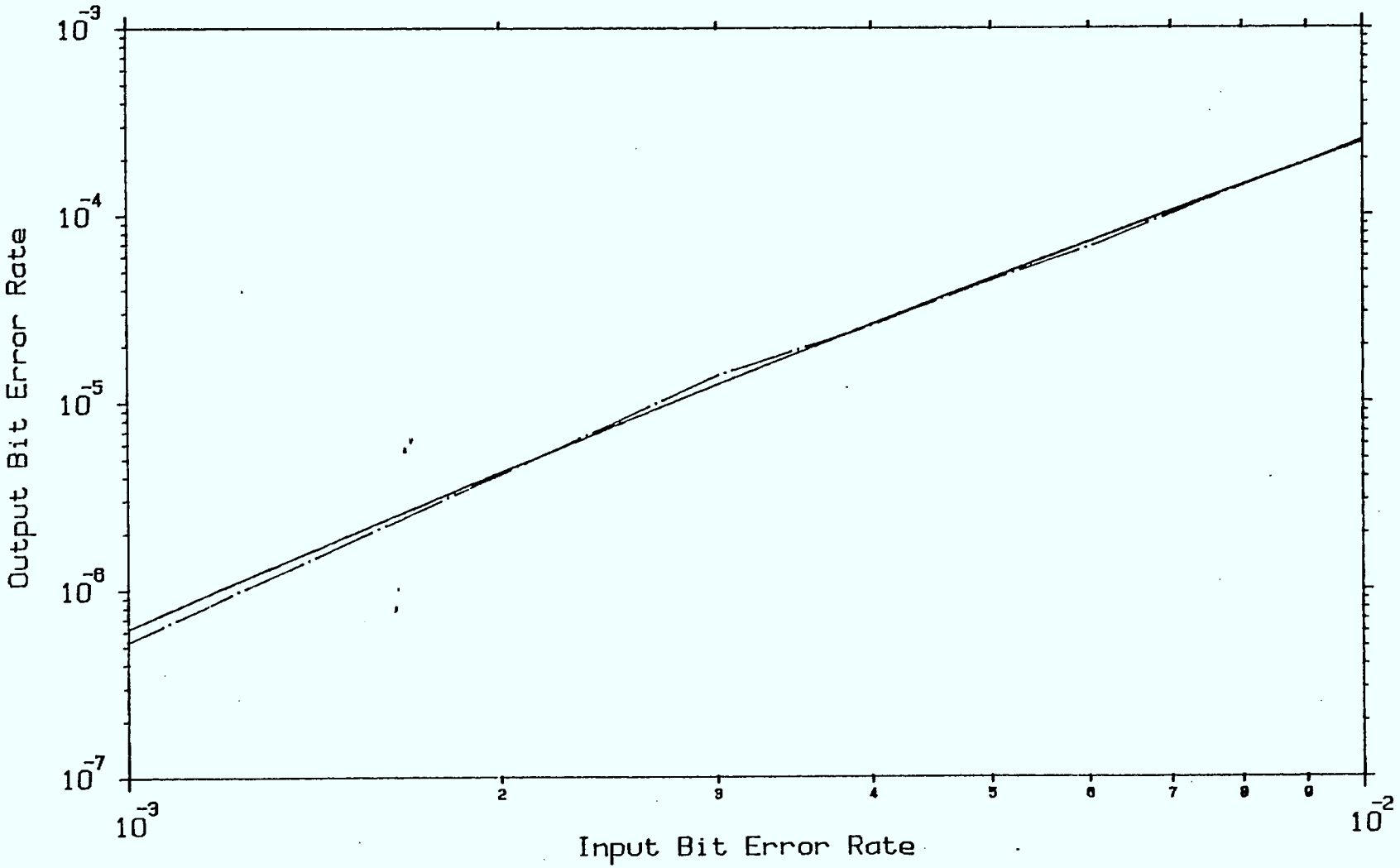

Probability of Decoding Failure
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
x - Program
10^{-4}
10^{-3}
10^{-2}
Input Bit Error Rate
Product Code - Performance with Independent Errors

Figure 3.3



Product Code - Performance with Independent Errors

Theory_____

Program_____ . _____ .

Figure 3.4

The results of the comparison for the performance indices described above, may be found in Figures 3.5 and 3.6. The points evaluated by the evaluation program have very small standard deviations (and the bounds determined are quite tight in this range) and as a result, extremely close agreement between the theory and the program is to be expected - and was obtained - indicating the algorithm is operating correctly.

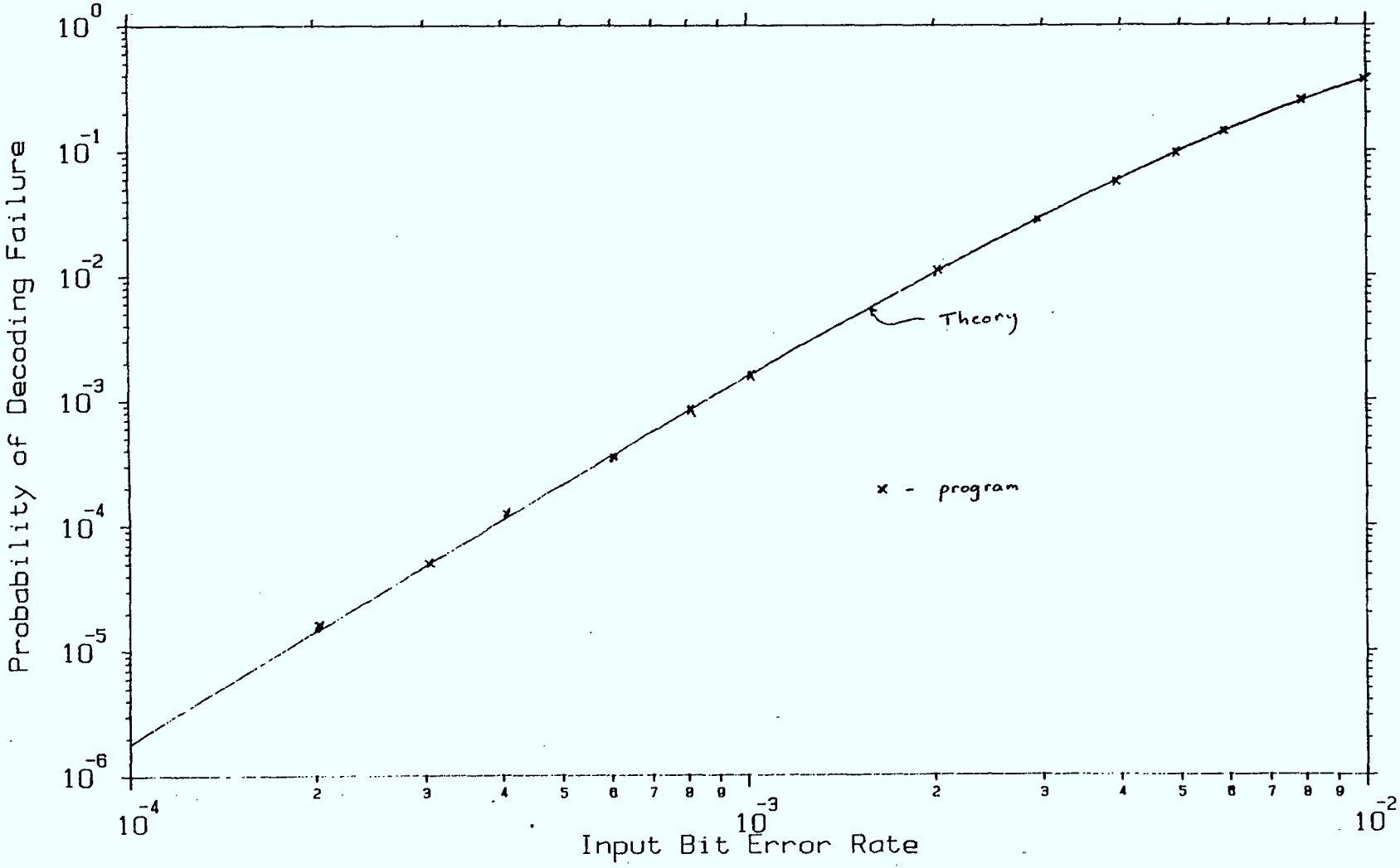
(c) Bundle Code

The bounds for the bundle code are much more difficult to determine. To simplify the calculation a slight modification of the decoding algorithm was made. The modification to the algorithm is as follows:

Previous Bundle Strategy - If more than two packet were rejected by the Code C, then undecoded errors are insertd into the bundle. If only one packet is rejected by the Code C, then a line of erasures is inserted into the bundle.

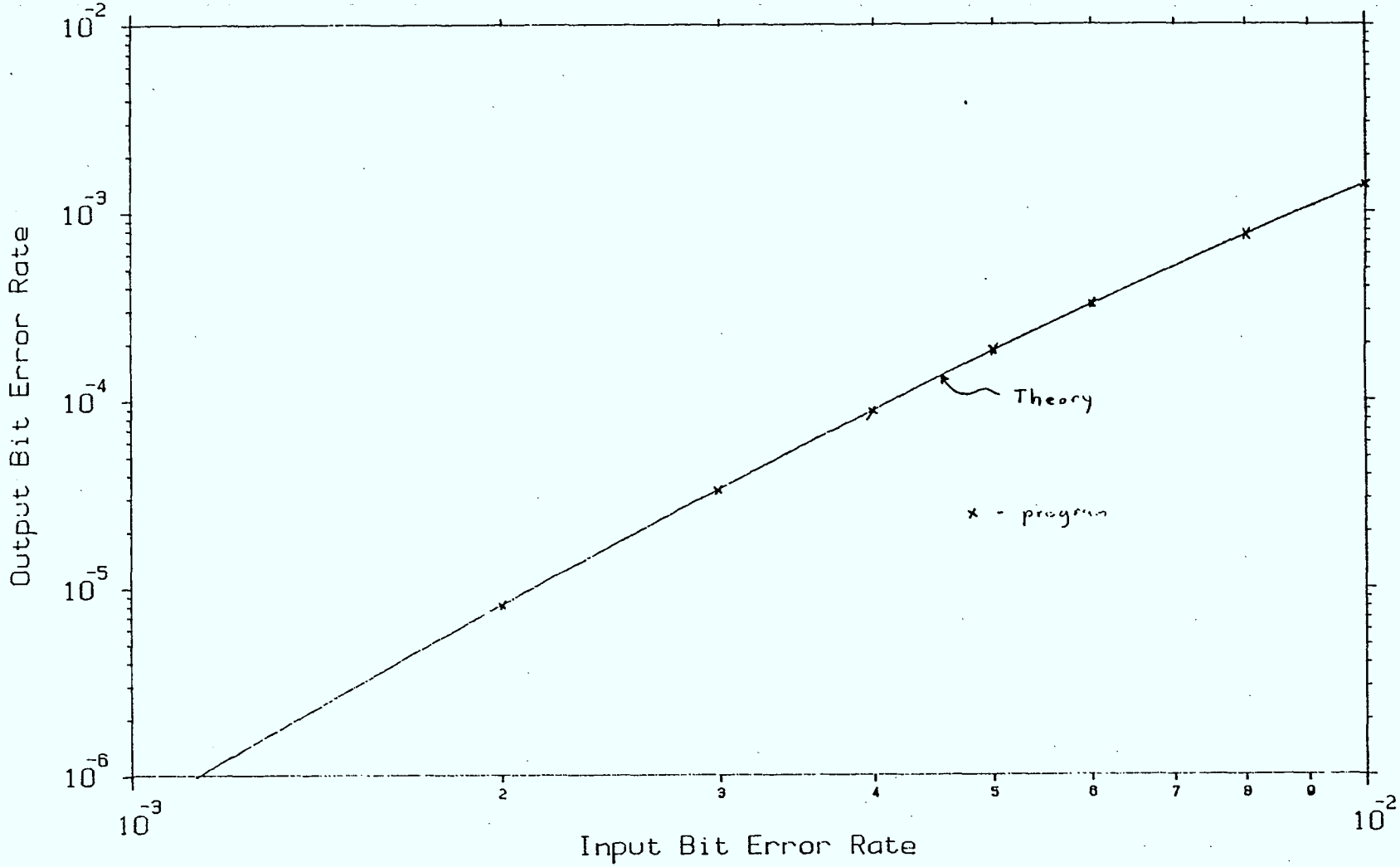
Modified Strategy - A line of erasures is inserted into the bundle any time there is a Code C rejection.

The modified algorithm is expected to perform worse than the unmodified algorithm, however, validation of the modified algorithm effectively validates the other algorithm.



Code C - Performance with Independent Errors

Figure 3.5



Code C - Performance with Independent Errors

Figure 3.6

In Appendix A, the probability of a decoding failure for the bundle code has been theoretically estimated. The difficulty in performing this task makes the bound not as tight as that determined for Code C. The results of the comparison of the evaluation program to the theoretical calculation is given in Figure 3.7. The close agreement indicates that the algorithm is operating correctly.

3.3 Performance on Simulation Error Sequences

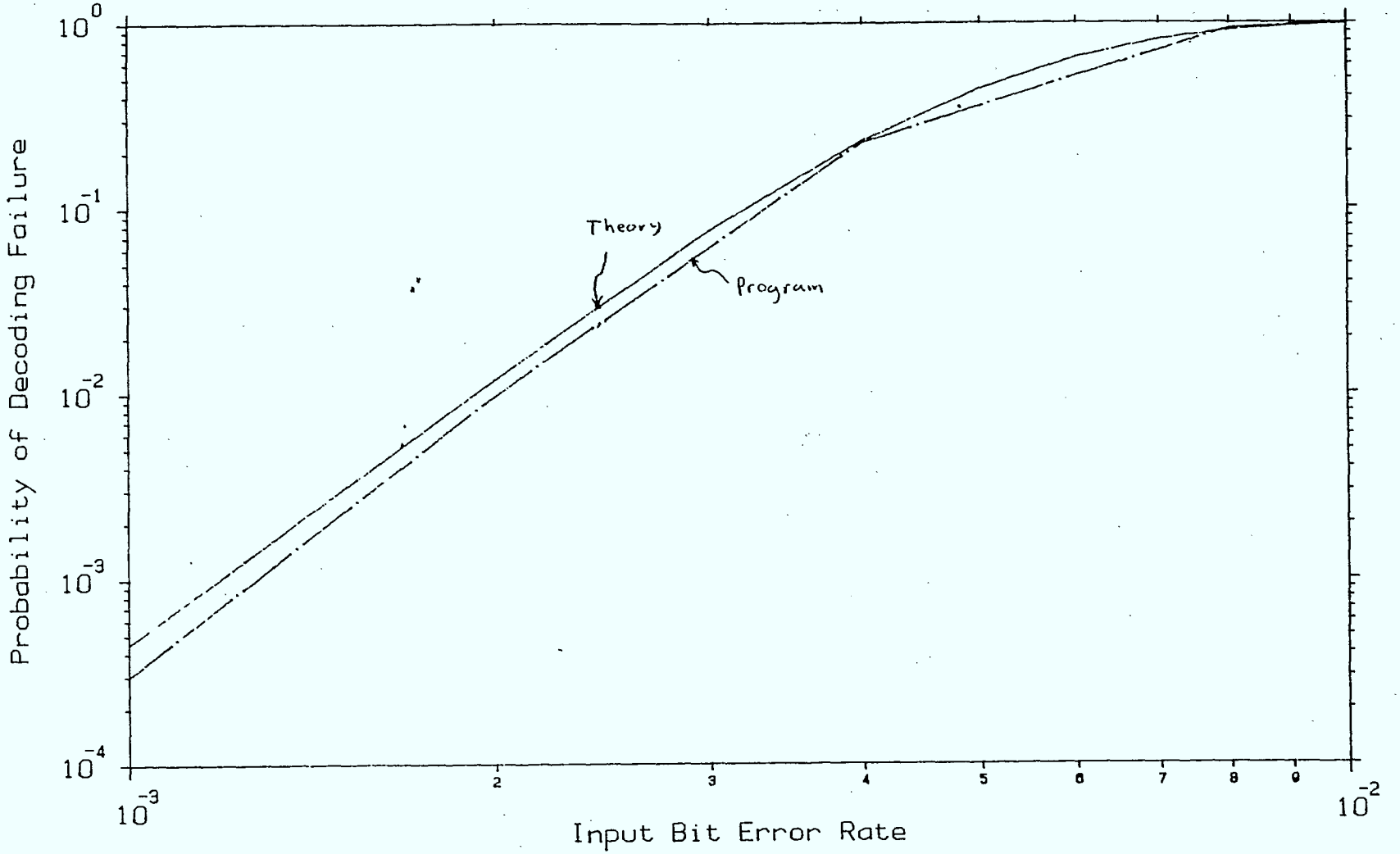
This section examines the performance of the various codes on the error sequences generated by the Telidon simulation program. The comparison will be based on the Rejection Rate of the codes as this is the limiting parameter with regard to the performance thresholds evaluated in [1]. It is important to note that the Rejection Rate given in this section takes into account smeared packets as well the Rejections of the Prefix code. The Input Bit Error Rate given in this section is the error rate excluding the contribution of the smeared packets (this enables the results here to be used in conjunction with the performance curves in other sections of the report, where the error rate is similarly defined).

The codes are evaluated for three receiver configurations, viz:

Case 1 - Peak Detector slicer / Zero-crossing BTR

Case 2 - Averaging slicer / Modified Zero-crossing BTR

Case 3 - Averaging slicer / Correlator BTR.



Bundle Code - Performance with Independent Errors

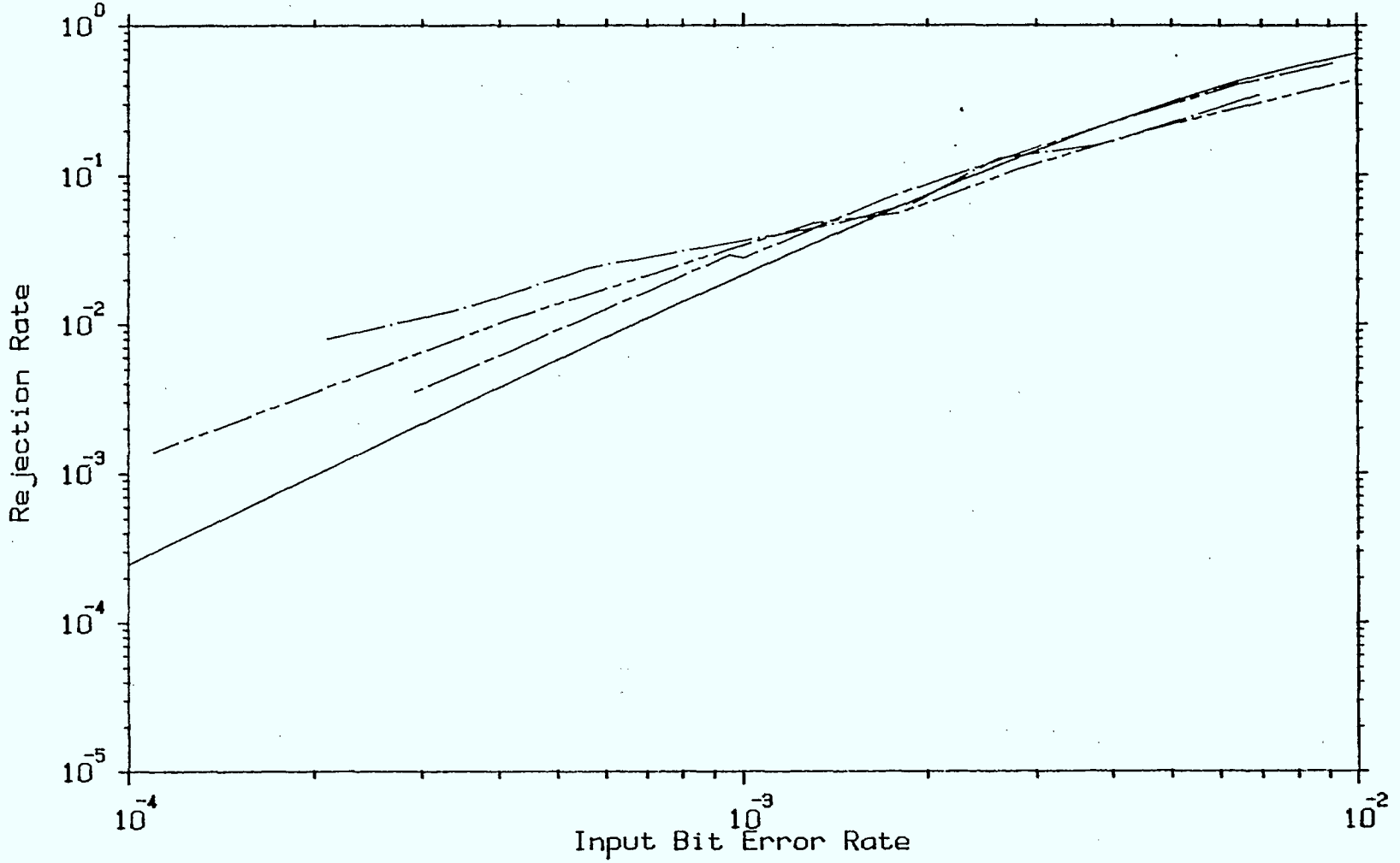
Figure 3.7

3.3.1 Relative Performance of Receiver Configurations

Figure 3.8 shows the Rejection Rate of the Product code for the three receiver configurations. Also included on this figure is the Rejection Rate for the case of Independent errors (note - the Prefix rejections are not factored into the Independent error case). The first thing to notice from this figure is that as the receiver configuration improves (Case 2 is considered an improvement over Case 1, etc.) the Rejection Rate approaches the case of Independent errors. Figure 3.9 and 3.10 show the same trend for Code C and the Bundle code respectively. Another observation that can be made is that as the coding technique increases in terms of strength (error correction ability) the degradation with respect to the Independent error curve increases. This effect is particularly apparent for the Case 1 receiver configuration. An explanation for these trends will be given in section 3.5.

3.3.2 Relative Performance of the Codes

Figures 3.11, 3.12 and 3.13 show the performance with each of the Case 1, Case 2, and Case 3 receiver configurations of the Product code, Code C, and Bundle code. Also included on these figures is the 'Telidon threshold' which indicates the acceptable rejection rate - see [1]. From Figure 3.11 it can be seen that there is very little difference in the performance of the various codes in the vicinity of the threshold for the Case 1 receiver configuration. With each step of improvement in receiver, however, the performance of the codes separates and the more powerful codes become more attractive. For the Case 3 receiver configuration an input bit error rate of 0.003 will still yield acceptable performance (assuming the threshold is accurate). The implications of this



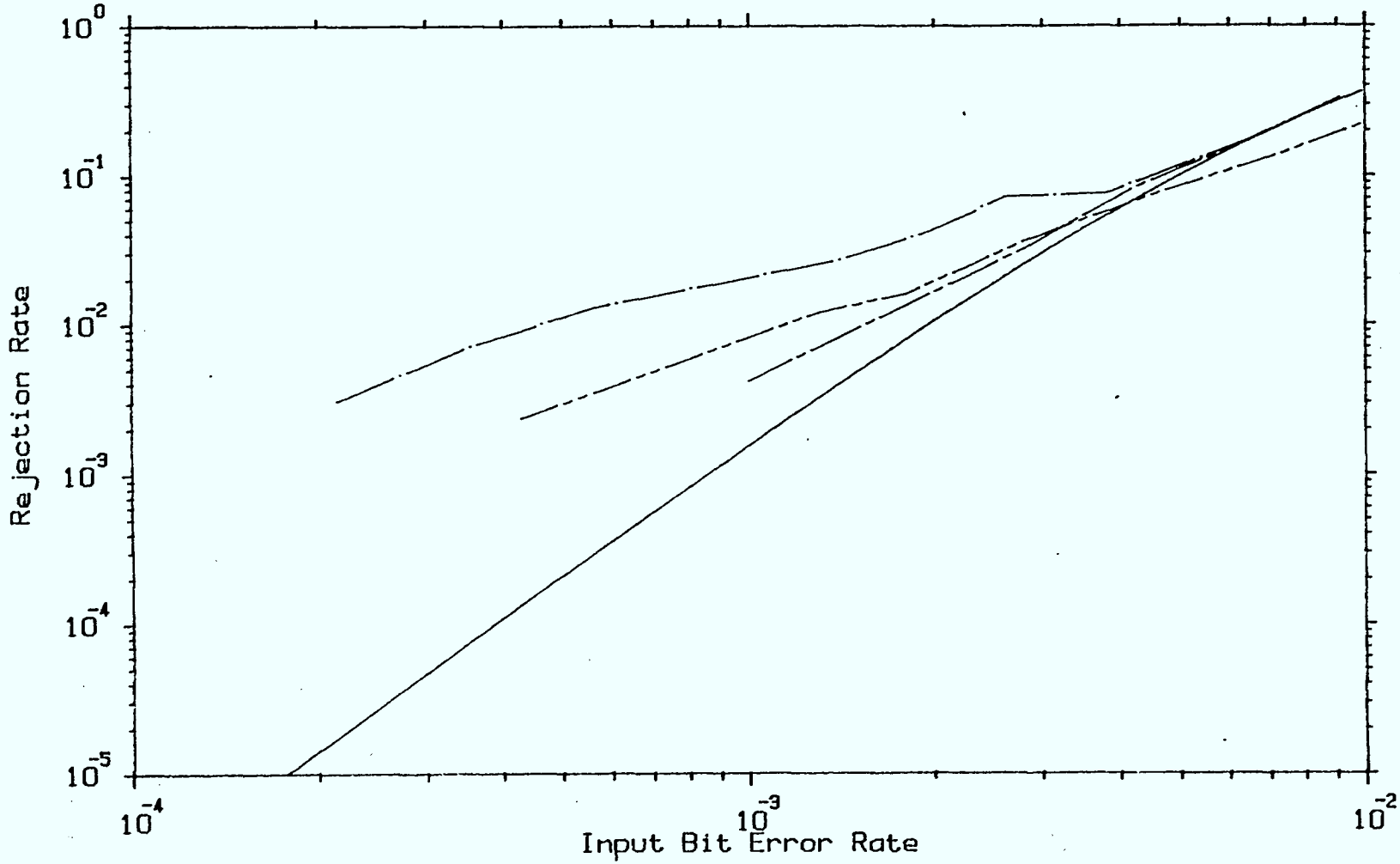
Product Code - Performance with Simulation Error Sequences

Independent Errors : _____

Case 1 : _____ . _____ . Case 2 : ___ _ _ ___ _ _

Case 3 : _____ _ _____ _

Figure 3.8



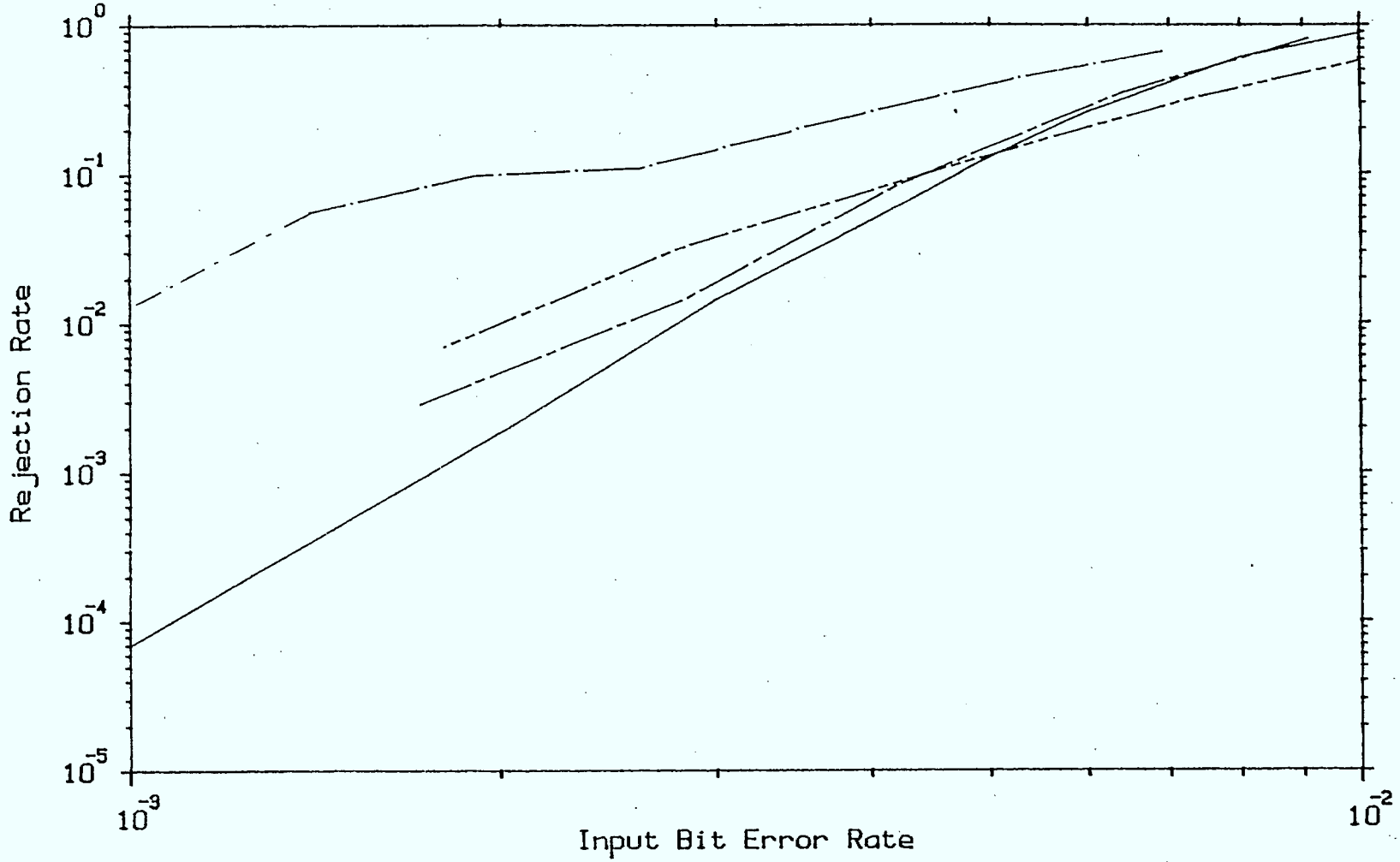
Code C - Performance with Simulation Error Sequences

Independent Errors : _____

Case 1 : _____ . _____ . Case 2 : ___ _ _ ___ _ _

Case 3 : _____ _ _____ _

Figure 3.9



Bundle Code - Performance with Simulation Error Sequences

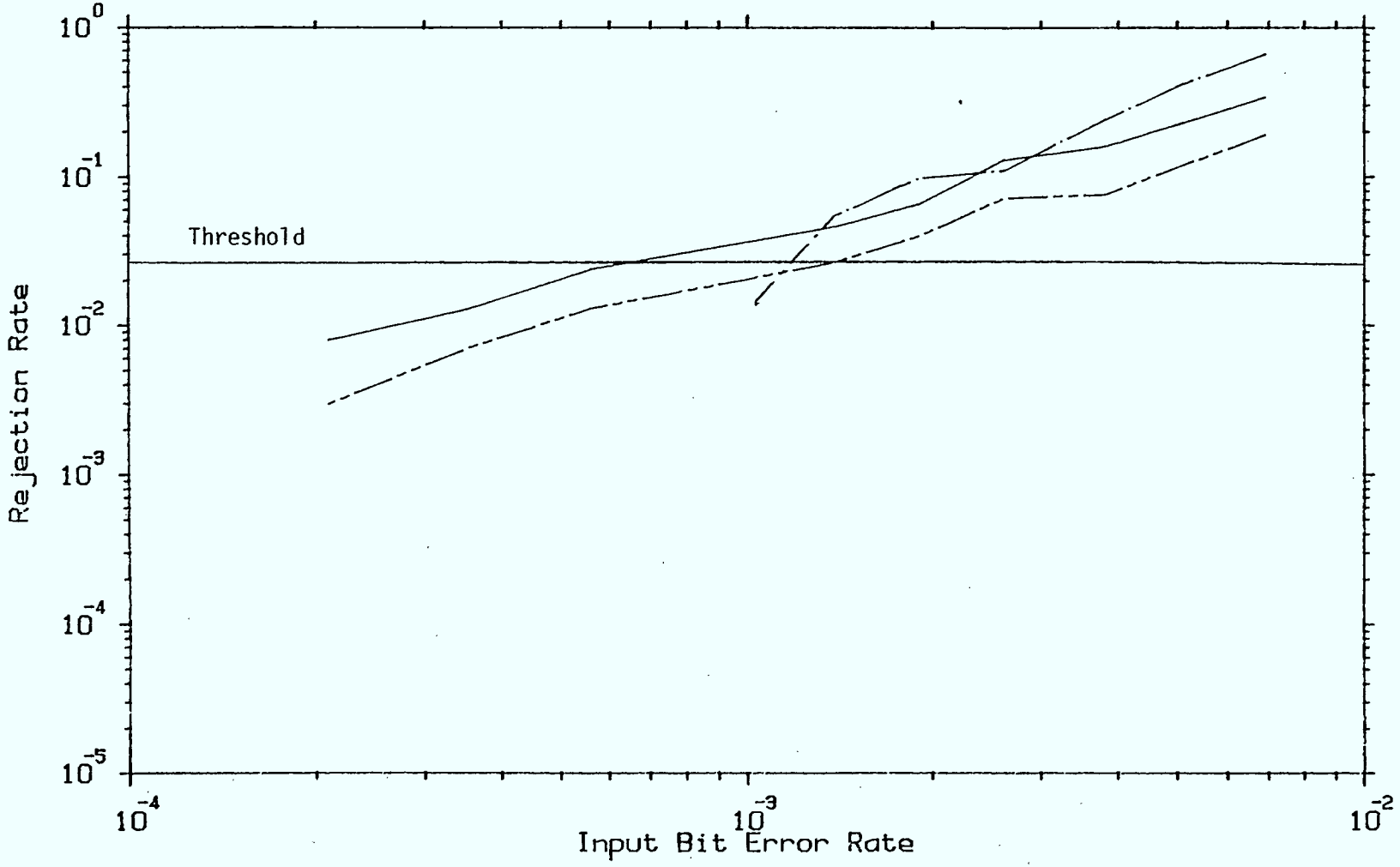
Independent Errors : _____

Case 1 : _____ . _____ .

Case 2 : ___ _ _ ___ _ _

Case 3 : _____ _ _____ _

Figure 3.10

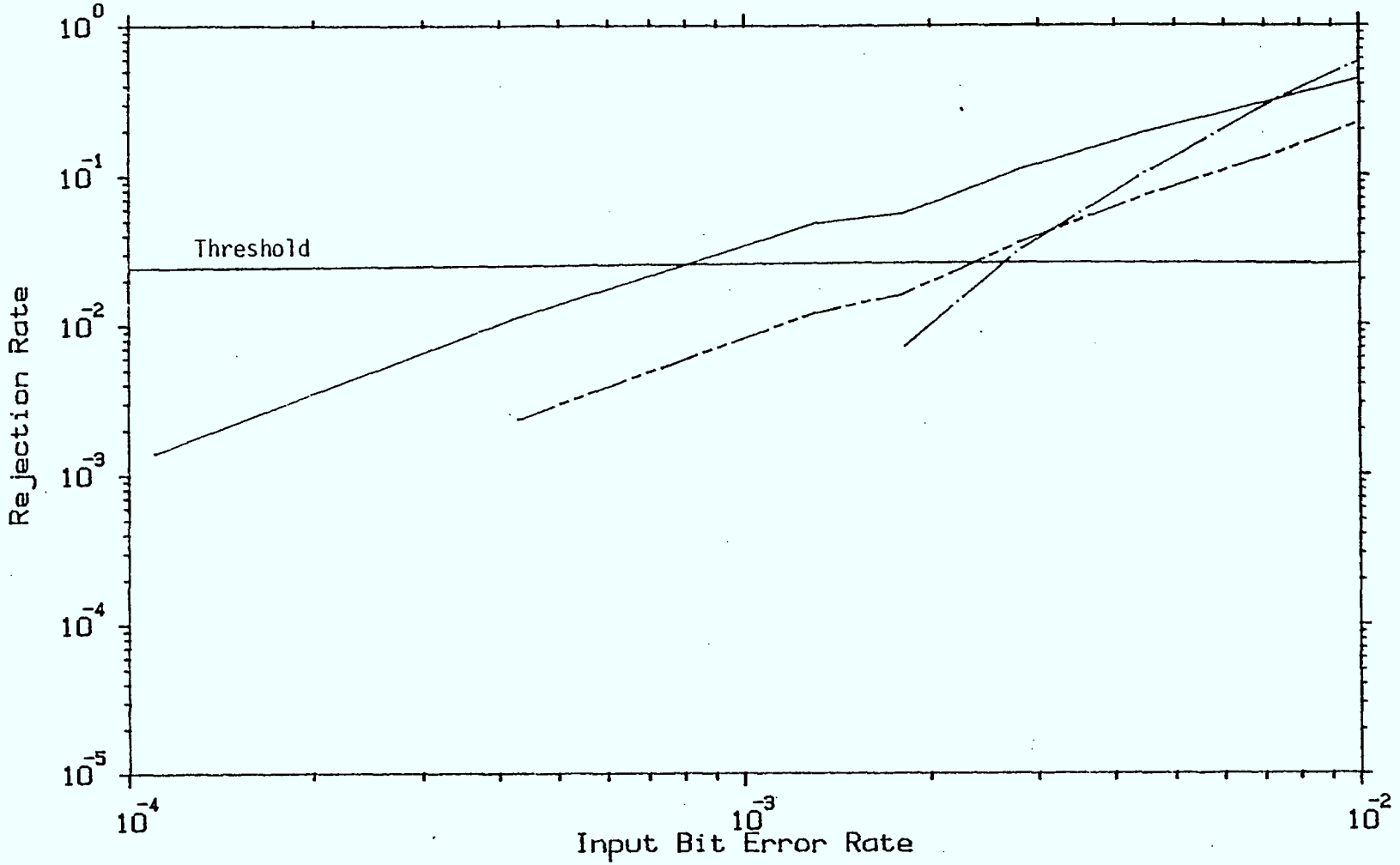


Case 1 - Performance with Simulation Error Sequences

Product : _____ Code C : ___ _ _ ___ _ _

Bundle : _____ . _____ .

Figure 3.11

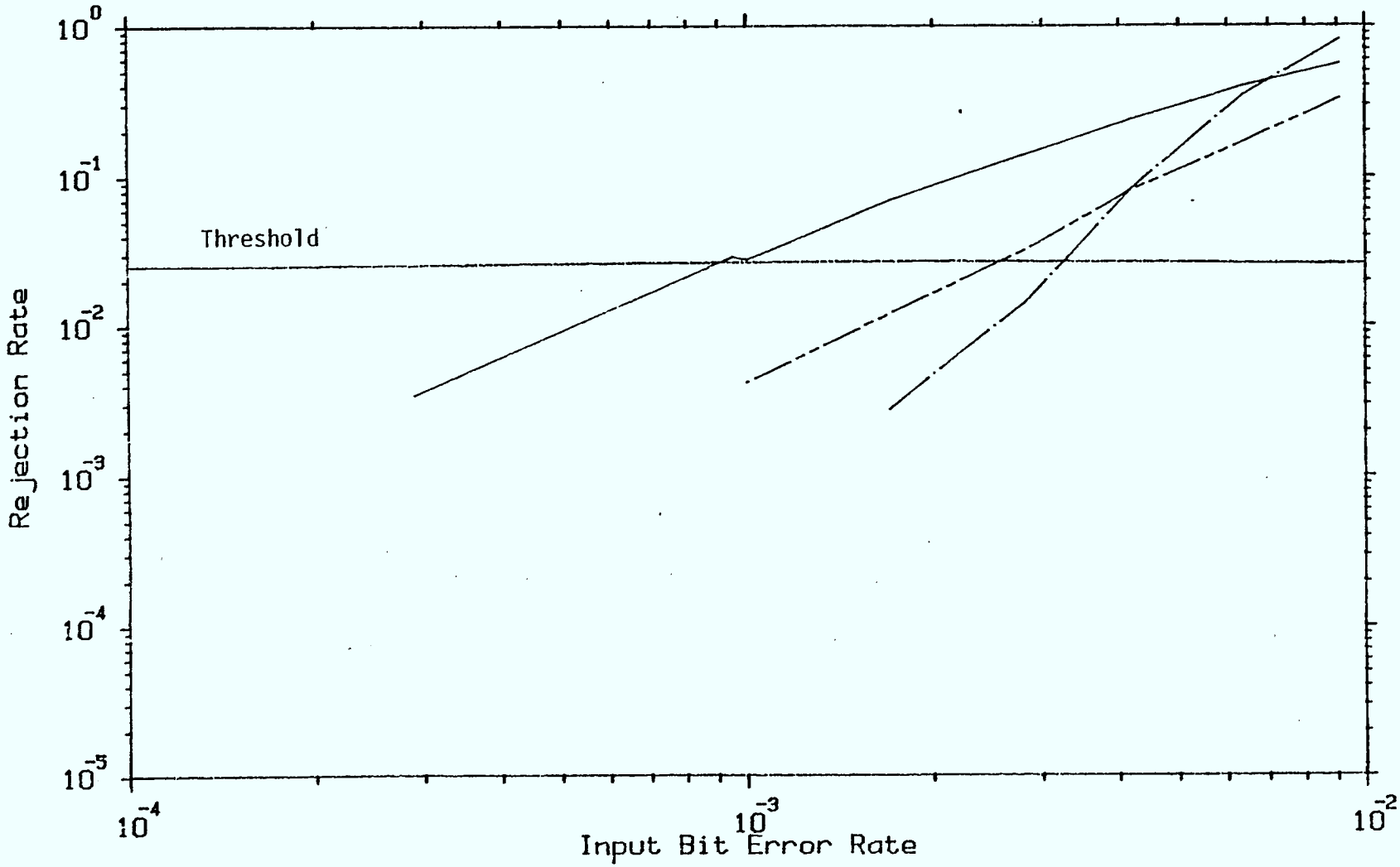


Case 2 - Performance with Simulation Error Sequences

Product : _____ Code C : ___ _ _ ___ _ _

Bundle : _____ . _____ .

Figure 3.12



Case 3 - Performance with simulation Error Sequences

Product : _____ Code C : ___ _ _ ___ _ _

Bundle : _____ . _____ .

Figure 3.13

phenomenon are two-fold. First, the improved receiver configuration will result in a reduced input bit error rate to the decoder for a fixed TV signal to noise ratio. Secondly, a larger coding gain may be expected with the improved receiver configuration. From this it is clear that improving the receiver performance results in a substantial improvement in the overall performance of the system.

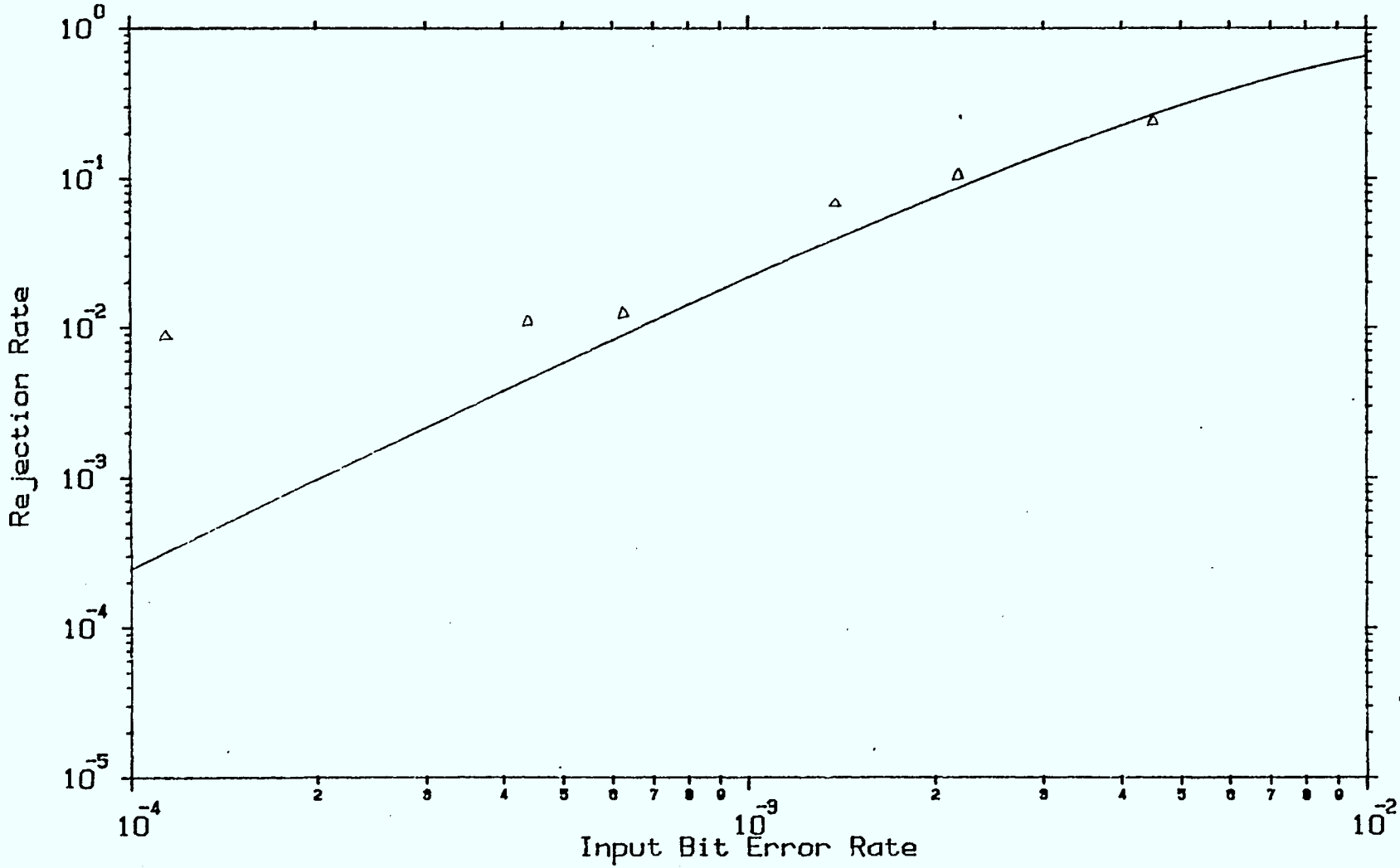
3.4 Performance on CRC Measurement Data

This section examines the performance of the various codes on the error sequences measured in CRC field trials. In many cases the evaluation could not be performed as the error sequence files contained no errors or an insufficient number of errors to generate a reasonable statistic. Again the performance measure will be the rejection rate of the various codes as a function of the input bit error rate.

Figures 3.14, 3.15 and 3.16 show the performance of the Product code, Code C, and Bundle code for the CRC measurement data. The degradation relative to the Independent error case is similar to that experienced by the Case 1 receiver configuration. This is reasonable as the Case 1 receiver is a model of the field equipment used to collect the measurement data. The explanation for the degradation relative to the ideal case is handled in the following section.

3.5 Explanation and Conclusions

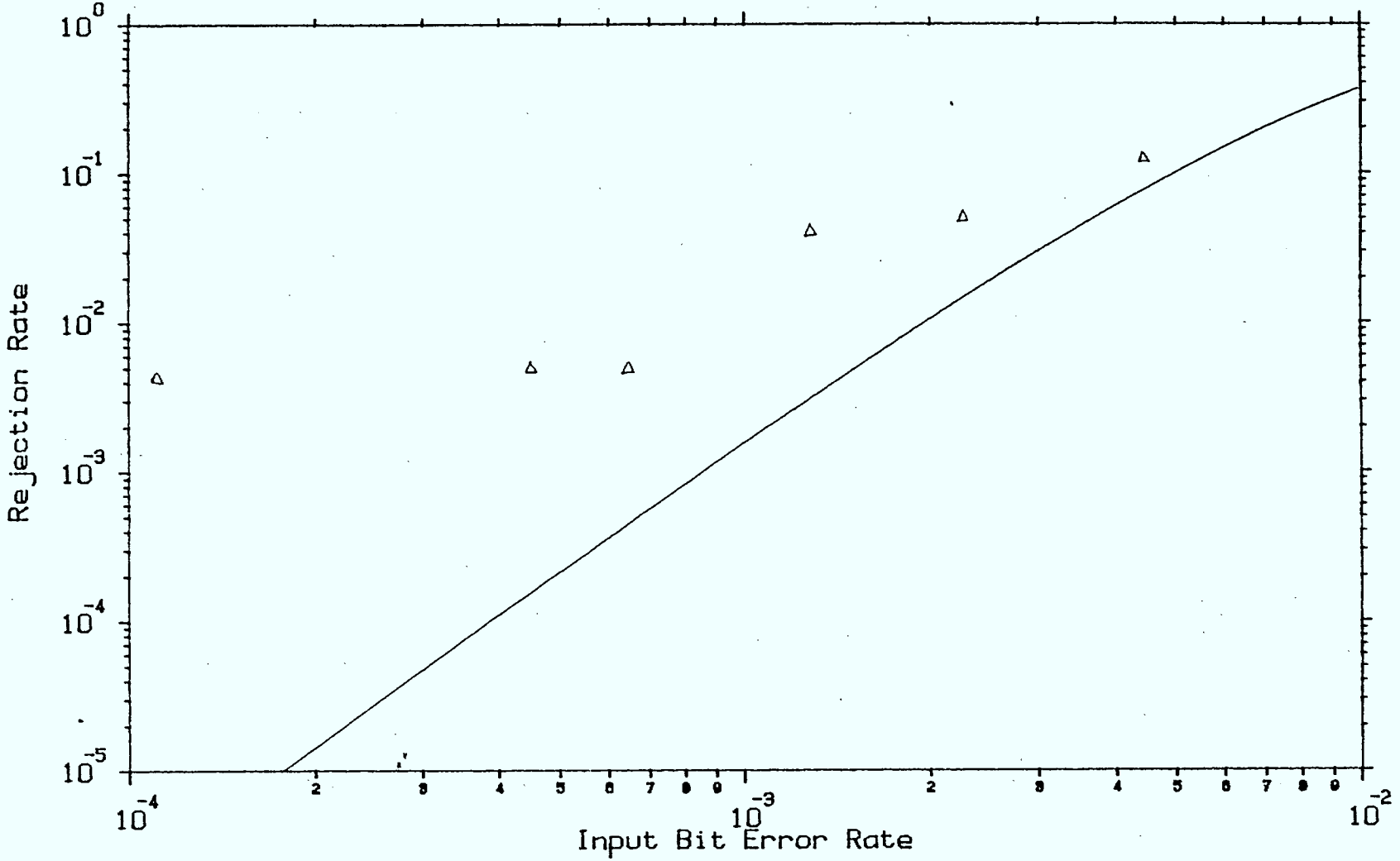
The results of the coding analysis may be summarized as follows. The theoretical analysis of the proposed coding techniques provided in Appendix A is matched closely by the software implementation in an ideal channel at the BERs of interest.



Product Code - Performance with CRC measurement data

Independent errors _____

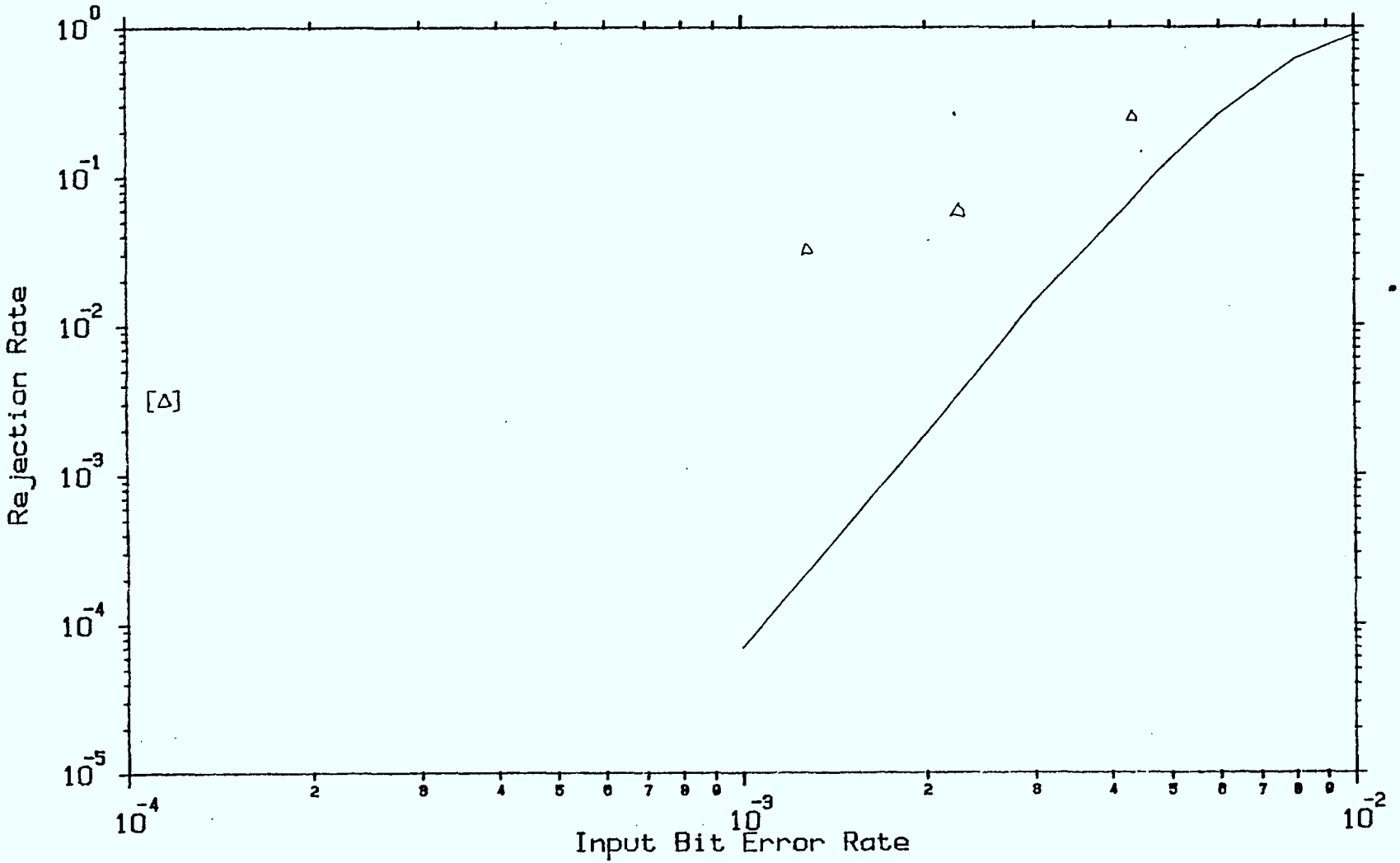
Figure 3.14



Code C - Performance with CRC measurement data

Independent errors _____

Figure 3.15



Bundle Code - Performance with CRC measurement data

Independent errors _____

Figure 3.16

A comparison of the coding performance of the simulated Telidon error sequence data and field measurement data indicated that packet-wise correlation of errors results in strong coding degradation. These correlations are primarily due to bit timing and slicing level errors. One of the early receiver (Case 1) exhibits the worst bit timing and slicing level performance of those simulated, and this poor performance is verified by field measurement data. A receiver with near optimum bit timing recovery and slicing level techniques was shown to result in little degradation of performance.

The poor performance of the Case 1 receiver with respect to coding noted in sections 3.3.1 and 3.4 is due to the fact that this implementation causes a significant error correlation within packets because of the poor slicing level and bit timing receovery technique. This would obviously degrade the performance of a code designed for independent errors.

In summary, performance can be enhanced in two distinguishable ways when bit timing and slicing level performance are improved. The first is a reduced error rate into the decoder for a specific SNR. Secondly, the reduced correlation among errors results in a larger coding gain with the decoder.

4.0 SLICING LEVEL ALGORITHMS

The objective of this task of the contract was the development of suitable slicing level algorithms for the Telidon signal format. In particular an adaptive slicing level algorithm was to be developed and compared to existing and other proposed techniques. In review, the objective of the slicer is to estimate the average d.c. level of the data in the Telidon signal format.

The performance of the Telidon decoder is as much dependent on the bit timing recovery algorithm as it is on the slicing algorithm, thus to compare slicing algorithm performance one must fix the BTR algorithm. However as discussed in the following chapter, there are a number of BTR algorithms. As a result this chapter will contain a number of forward references to the BTR algorithms of the next chapter.

The structure of this chapter is the following. The first section reviews the existing techniques of determining the slicing level and their drawbacks. The second section discusses the design of a modified version of an existing technique, the third section introduces an adaptive slicer. The fourth section compares the simulated performance for all these slicing techniques with various bit timing recovery techniques, and under non-multipath and multipath conditions.

4.1 Existing Techniques

In the previous Telidon study [1], two techniques were investigated for estimating the proper slicing level for a Telidon signal. They were the peak detector approach and the averaging approach.

4.1.1 Peak Detector Slicing Algorithm

This peak detector algorithm has been used in conjunction with the zero crossing bit timing recovery scheme in an actual hardware decoder which uses the two byte clock sync in the Telidon signal format to estimate both the slicing level and the clock phase serially as opposed to in parallel. It begins by estimating the slicing level, using two peak detection circuits (one for the "overshoots" and one for the "undershoots") both of which have a short rise time and a long decay time. The slicing level is taken to be the midpoint between the two peak values. Once the slicing level has been determined, the decoder uses a single zero crossing to choose one of five possible clock phases. There are several severe weaknesses in the method that result in the decoder's performance falling far short of optimal. Several of the problems with this slicing level determination technique are the following.

- The peak detectors are very sensitive to impulse noise. In fact simulations have shown that a loss of approximately 3 dB can be attributed to this slicing level and clock phase circuitry even under fairly ideal conditions (i.e. no multipath propagation and ideal carrier recovery) [2].

- Even relatively modest multipath propagation can cause a substantial error in the estimated slicing level. This is because the signal level immediately preceding the clock sync signal is at 0 IRE where as the average level during the clock sync signal (i.e. the slicing level to be determined) is 35 IRE. Thus any multipath propagation will result in an apparent shift of level, for a duration that is proportional to the multipath differential delay. Note that if the delay spread is large enough that a

delayed version of the colour burst or sync pulse gets superimposed on the clock sync signal, then the situation gets even worse.

- The optimum slicing level may be time varying due to a time-varying phase offset in the recovered carrier. Thus any technique that attempts to determine the slicing level during the first two bytes and then freeze it for the remainder of the time, will yield inferior performance unless the carrier recovery is very good.

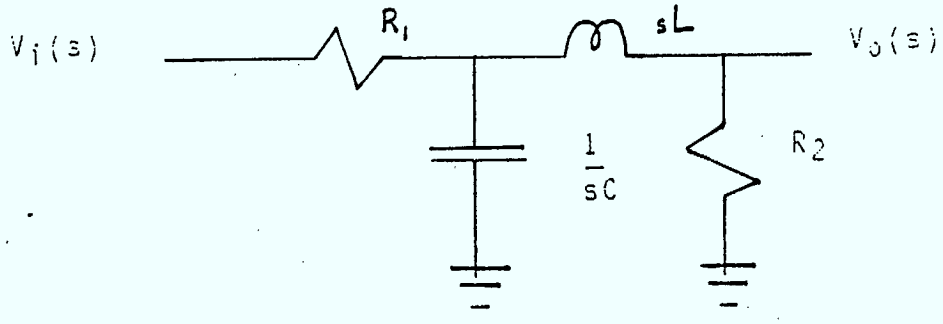
4.1.2 Averaging Slicer Algorithm

In an effort to improve upon the performance of the peak detector slicing algorithm a simple numerical averaging technique was introduced into the simulation. This approach eliminated the problem of sensitivity to impulse noise and as illustrated in [1] resulted in a significant improvement in performance.

4.2 Modified Averaging Slicer

This modified version of the averaging slicer was included to simulate a hardware implementation. The circuit implementation which was simulated is shown in Figure 4.1. This circuit should be followed by a amplifier with a gain of 2 to provide a unity d.c. gain. In software this lowpass filter was emulated by using a discrete time infinite impulse response (IIR) approximation which was impulse response invariant at the sample points [3]. The details of this implementation are given in Appendix C.

The major benefit of using a lowpass filter to estimate the slicing level is the reduction in sensitivity of the



$$\frac{V_o(s)}{V_i(s)} = \frac{1}{s^2LC\left(\frac{R_1}{R_2}\right) + s\left(\frac{L}{R_2} + CR_1\right) + \left(1 + \frac{R_1}{R_2}\right)}$$

$R_1 = R_2 = 500$ ohms
$C = 1.2$ nF
$L = 390$ uH

Figure 4.1. Averaging Filter for Modified Slicer

slicing level estimate to impulse noise. In designing this filter it is desirable to have a filter with a narrow bandwidth in order to minimize the effect of noise, to suppress the 2.863636 MHz clock sync signal, and possibly to suppress the 3.58 MHz colour burst signal. On the other hand the filter step response should settle fairly quickly so that the level change (i.e. from 0 IRE to 35 IRE) does not affect the filter output for too long. Also, the order of the filter should be low so that it can be economically realized.

The output of this filter is "frozen" at the end of the clock sync signal and used as the slicing level for the rest of the Telidon line (in the non-adaptive version). This prevents the slicing level from tracking the d.c. level variations due to the data modulation. The performance of this slicer on a noiseless signal is shown in Figure 4.2. From this figure and further investigation we noted that if the signal fed to the averaging slicer is a windowed version of the Telidon line then the window should open signficantly before the start of the clock sync signal to reduce the overshoots on acquiring the slicing level.

To analyze the performance of the averaging slicer several simplifying approximations must be made. The first assumption is to assume the channel is a wideband white and Gaussian and that there is no ISI causing one bit to affect its neighbours. Obviously this is not true in practice but it allows us to obtain an upper bound on performance. The second assumption is that the noise on the slicing level is independent of the noise on the data bits. This assumption is quite reasonable as the slicing level is determined from the clock run-in sequence which has a significant separation in time from the data bits. The third

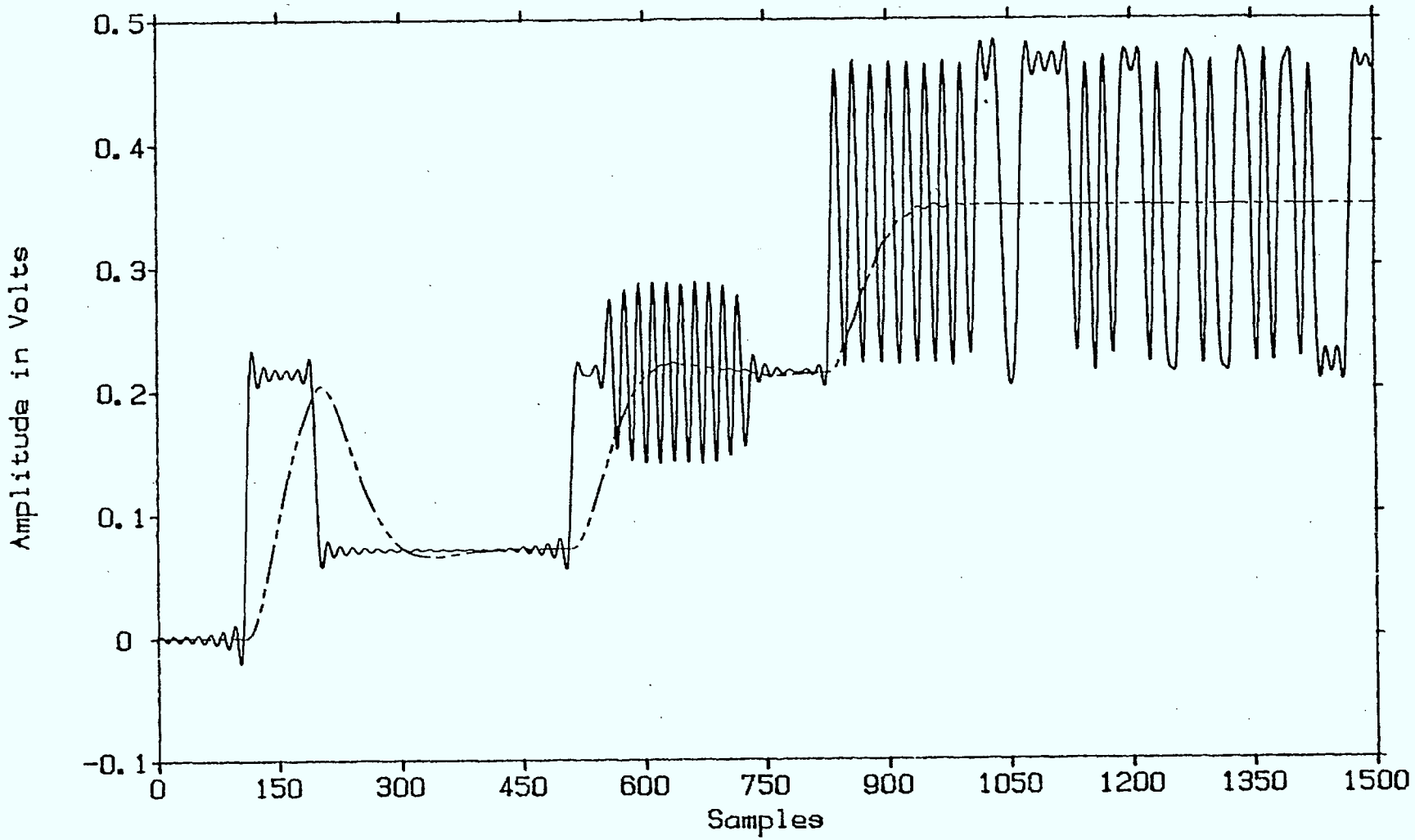


Figure 4.2 Telidon line response of modified averaging slicer.

assumption is that of perfect clock synchronization. The effects of imperfect clock synchronization will be discussed later.

With these assumptions the noise process can be modelled as in Figure 4.3. With this model the probability of error given the slicing level is

$$P(\text{error}|s) = \frac{1}{2} Q(\frac{A-S}{\sigma}) + \frac{1}{2} Q(\frac{A+S}{\sigma})$$

if the "0" and "1" bits are assumed to be equally likely. However the slicing level, S, also has a Gaussian distribution. To see this observe that the input to slicing filter is essentially an out of band tone plus Gaussian noise, thus the output process will be approximately Gaussian. Thus the overall probability of error is

$$P(\text{error}) = \frac{1}{\sqrt{2\pi}\nu} \int_{-\infty}^{\infty} P(\text{error}|S)\, e^{\frac{S^2}{2\nu^2}}\, ds$$

where ν^2 is the variance of the slicing level. This expression can be evaluated in closed form but the result can more easily be obtained from physical considerations of Figure 4.3. Since the decision is based on the difference between the data sample and the slicing level, the independent noise processes on both can be combined into a single noise process with variance equal to the sum of the individual variances. Thus

$$P(\text{error}) = Q(\frac{A}{\sqrt{\sigma^2+\nu^2}})$$

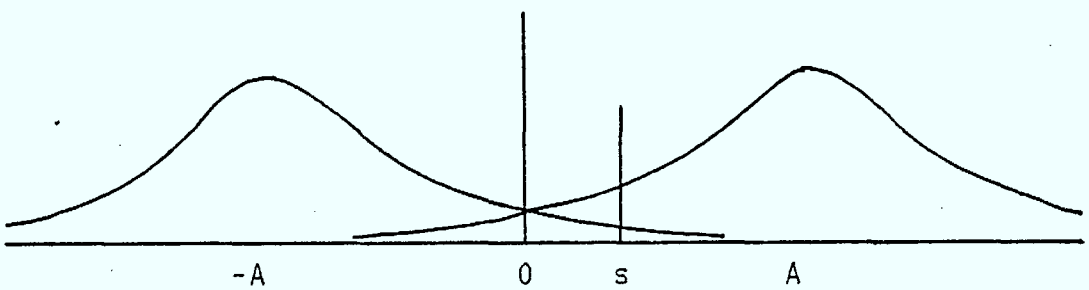


Figure 4.3. Noise Distribution with Offset Slicing Level.
Assuming i) s is independent of data
ii) wideband Gaussian channel
iii) perfect clock recovery.

For the filter of Figure 4.1, the output noise power

$$\nu^2 \simeq 0.1\ \sigma^2$$

one tenth of the input power, which is the noise in the video bandwidth. This increase in noise will cause a 0.4 dB degradation of performance from ideal. Decreasing the filter bandwidth would decrease the noise power but at the expense of increasing the filter response time. That is, the filter bandwidth can only be decreased to the point that it can still reach steady state during the clock sync period. Since the clock sync is only 16 bit periods long and a filters bandwidth is inversely proportional to its response time, this implies that the minimum filter bandwidth is in the neighbourhood of 300 kHz, as is the specified filter.

What was not considered in this analysis was the fact that the slicing level is frozen for the length of the packet. This implies that the slicing level is correlated between bits within the same packet, but uncorrelated between packets. This means that over a small number of packets one may expect considerable variance from the above prediction, but as the number of packets becomes large and one gets a true Gaussian distribution of slicing levels, then the gross bit error rate will approach that predicted (under the stated assumptions).

In practice both the presence of a narrowband channel, imperfect synchronization and other distortions will cause a greater loss in performance than the 0.4 dB suggested above. However, it does indicate that in the absence of these distortions, the modified averaging slicer is definitely a good approach.

4.3 Adaptive Slicer Design

If time-varying slicing levels, due to phase errors on the recovered carrier or due to wide multipath spreads are to be tracked then adaptive slicing level circuitry is required. The objective of an adaptive slicer is to track a slowly varying "d.c." level. The impairments which must be reduced are the effects of the data modulation and the noise.

Obviously, since we are estimating a pseudo "d.c." level the adaptive slicer will contain a lowpass filter. However, a lowpass filter by itself is insufficient because a long string of bits of the same polarity would cause the filter output (slicing level estimate) to track toward that extreme.

To remove the effects of data modulation, one must form a control variable which is independent of the data. Intuitively, one would choose the slicing level to be the middle of the data "envelope". This would imply estimating (averaging) the most recent "1" levels and the most recent "0" levels and choosing the slicing level as the average of the two.

The simplest way of implementing this strategy, is to average the single most recent "1" and "0" levels (rather than a series of each). A circuit which implements this strategy is shown in Figure 4.4. This circuit attempts to estimate the logic "0" level and the logic "1" level in a decision directed manner.

At the beginning of the Telidon line, the switch in Figure 4.4 is in the upward position allowing the video to pass directly to the low pass filter, which performs the same

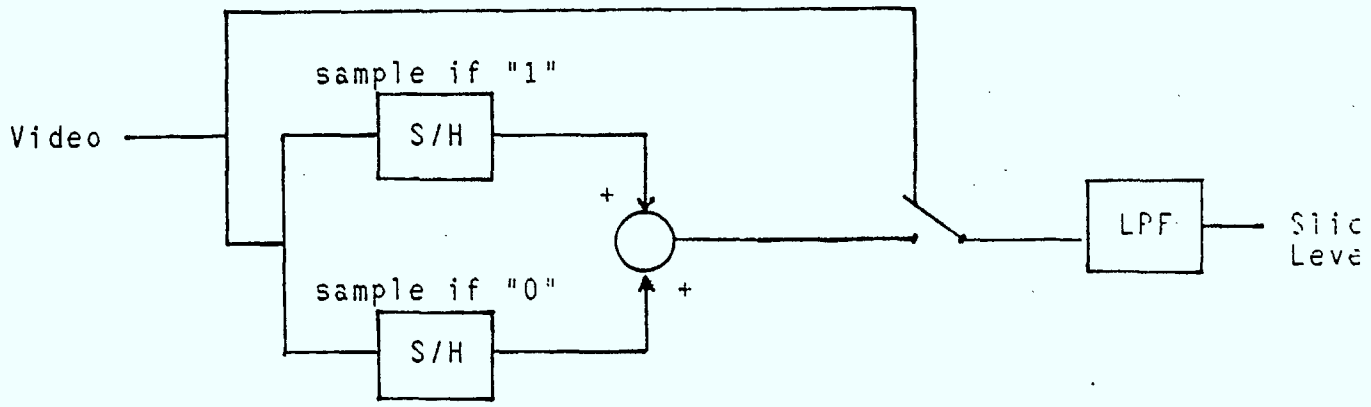


Figure 4.4. Adaptive Slicing Level Circuitry.

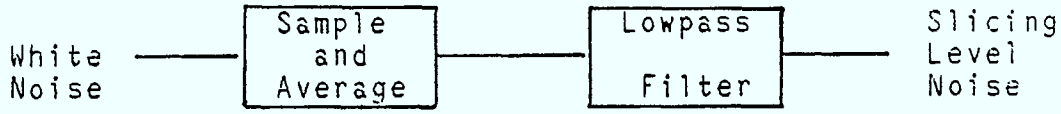


Figure 4.5. Block Diagram of Adaptive Slicer.

function as in the modified averaging slicer to obtain a first estimate of the slicing level. At an instant shortly after the bit clock phase is chosen (see the next chapter for details) and the sample and holds (S/H) have been initialized, the switch is thrown to engage the adaptive circuitry. The transient effects of switching are minimized by the lowpass filter. The adaptive circuitry samples and holds the most recent logic "1" and "0" values. The sum of these two levels is used to control the slicing level since the slicing level should be half of the sum.

Before analyzing the adaptive slicer, several points should be noted about adaptive slicers in general. Although they may be expected to perform better in the presence of multipath or recovered carrier phase errors, there are some tradeoffs. The major tradeoff is between adaptibility and noise performance. For a slicer to adapt quickly it must have a wide bandwidth, but this results in a deteriorated noise performance, and vice versa.

On a higher level, there is still another tradeoff between non-adaptive and more expensive adaptive slicers. That is, on what portion of the channels does one actually get a performance improvement using an adaptive slicer.

4.3.1 Analysis in a Gaussian Channel

The following development gives an approximate analysis of the adaptive slicer in a Gaussian channel with perfect bit synchronization. The performance of the adaptive slicer in multipath is discussed in the next section.

Several simplifying assumptions are made in the analysis of the adaptive slicer. The first of which is that when making a bit decision, the noise on the sample of the

received waveform is independent of the noise on the slicing level. This is a reasonable assumption because the slicing level is only affected by noise samples which are one bit period or more, previous to the present decision. (That is, noise samples one bit period or more apart are assumed to be independent.) Secondly, the analysis is done in the digital domain to represent the sample and hold operations.

Using the results of Section 4.2, we know we can estimate the performance of the adaptive slicer if we know its output noise variance, thus the analysis of the adaptive slicer is really a determination of the properties of this noise process. This analysis is broken down into two steps corresponding to the two blocks of Figure 4.5. (The analysis will not include the behaviour during the data preamble which is the same as discussed in Section 4.2).

The input noise to the adaptive slicer is assumed to be white and Gaussian with variance σ^2. The output of the sample and average is

$$s(n) = \frac{1}{2}\left(t(n_0) + t(n_1) \right)$$

where t(n) are the bit decision samples of the Telidon signal; n_0 and n_1 are the indices of the most recent "0" and "1" decisions, respectively. This can be approximated as

$$s(n) = s_0 + \frac{1}{2}\left[\nu(n_0) + \nu(n_1) \right]$$

where s_0 is the nominal slicing level and $\nu(n)$ are samples of white Gaussian noise. Let us consider the properties of the noise term

$$e(n) = \frac{1}{2}\left[\nu(n_0) + \nu(n_1)\right]$$

Assuming the input noise is of zero mean then e(n) is also Gaussian and of zero mean. Using the property that

$$E[\nu(i)\nu(j)] = \begin{cases} \sigma^2 & i = j \\ 0 & i \neq j \end{cases}$$

we can determine the second order statistics of e(n), defined by the correlation, $\phi_e(n)$.

$$\phi_e(n) = E[e(m)e(m-n)]$$

$$= \frac{1}{4} E\left(\nu(n_0^m) + \nu(n_1^m)\right)\left(\nu(n_0^{m-n}) + \nu(n_1^{m-n})\right)$$

$$= \frac{1}{4} E[\nu(n_0^m)\nu(n_0^{m-n}) + \nu(n_1^m)\nu(n_1^{m-n})]$$

since the properties of $\nu(i)$ imply that $E\left(\nu(n_0^i)\nu(n_1^j)\right)=0$ for all i and j. Also note that $E(\nu(n_0^i)\nu(n_0^j)]\neq 0$ only if $n_0^i=n_0^j$ and similarly for $\nu(n_1)$. There are three cases to consider

(i) $n = 0$ then $\phi_e(0) = \frac{\sigma^2}{2}$

(ii) $n = 1$ then one of the samples $\nu(n_0)$ or $\nu(n_1)$ will have been updated and therefore $\phi_e(1)=\frac{\sigma^2}{4}$

(iii) $n > 2$ On each increment of n, one of the noise samples is updated. There will be non-zero contribution to the correlation at lag n, only if one of the samples is the same as at lag

n=0. This occurs only if all (n-1) bits following the first are the same. If the bits are independent and the probability of a bit transition is $(1-\rho)$. Then the probability of a non-zero contribution at lag n is ρ^{n-1} and the magnitude of that contribution is $\frac{\sigma^2}{4}$.

Summarizing these case we can describe the general correlative properties of e(n) (combining cases (ii) and (iii)),

$$\phi_e(n) = \begin{cases} \frac{\sigma^2}{2} & n = 0 \\ \rho^{n-1} \frac{\sigma^2}{4} & |n| > 0 \end{cases}$$

Thus we have defined the properties of the noise process at the input to the lowpass filter of Figure 4.5. The power spectral density of this error process can be obtained by evaluating the z-transform of $\phi_e(n)$

$$\Phi_e(z) = \sum_{n=-\infty}^{\infty} \phi_e(n)\, z^{-n}$$

at $z = e^{j\omega}$, which gives

$$\Phi_e(\omega) = \frac{\sigma^2}{2} \left[1 + \frac{\cos \omega - \rho}{1 + \rho^2 - 2\rho \cos \omega}\right].$$

The noise power after sampling and averaging is

$$N_e = \frac{1}{2\pi} \int_{-\pi}^{\pi} \Phi_e(\omega)\, d\omega$$

which reduces to

$$N_e = \frac{\sigma^2}{2}.$$

Thus the total noise power after sampling and averaging is one half the input noise power, but it no longer has a constant spectral distribution.

To model the effects of the lowpass filter, we will assume that it is a one-pole filter and model it by a one step difference equation.

$$y(n+1) = a\ y(n) + b\ e(n)$$

where y(n) is the filter output. The z-transform of this filter is

$$H(z) = \frac{bz^{-1}}{1-az^{-1}}$$

If the filter is to pass d.c. with unity gain, then obviously b = 1-a. The z-transform of the autocorrelation of the noise process at the filter output is given by

$$\Phi_y(z) = H(z)\ H(z^{-1})\ \Phi_e(z)$$

The output noise spectrum is

$$\Phi_y(\omega) = \frac{b^2}{1 + a^2 - 2a\ \cos\ \omega}\ \left[\frac{\sigma^2}{2}\left(1+ \frac{\cos\ \omega - \rho}{1+\rho^2-2\rho\ \cos\ \omega}\right)\right]$$

and the output noise power is

$$N_y = \frac{1}{2\pi} \int_{-\pi}^{\pi} \Phi_y(\omega)\ d\omega$$

$$= \frac{\sigma^2 b^2}{2(1-a^2)}\ \left[1 + \frac{a}{1-a\rho}\right]$$

Note that this expression for the noise power on the adaptive slicing level contains two terms, the first is directly proportional to filtered white noise, and a second term which may be considered the adaptive portion.

To determine the degradation in performance caused by the noise on the slicing level, we shall use the results of the previous section where it was shown that in a ideal channel the expected bit error rate performance is

$$P_e = Q\left(\frac{A}{\sqrt{N_v+N_y}}\right)$$

where N_v is the noise power on the video signal. In the two sided digital domain $N_v = \sigma^2 B_v T$ where B_v is the two sided video bandwidth (after demodulation). T is the sampling period. Therefore the performance ratio is proportional to N_y/N_v.

If in the analog domain the lowpass filter has a bandwidth ω_o, then in the digital domain $a = e^{-\omega_o T}$ and for unity d.c. gain $b = 1-a$. It was shown in the discussion of averaging slicers that a minimum filter bandwidth of approximately 300 kHz was required to accurately estimate the slicing level during the clock sync signal. If we use the same bandwidth in the adaptive slicer and assume that the probability of a bit transition, $(1-\rho)$, is one half then

$$\frac{N_y}{N_v} = .124$$

This corresponds to a 0.5 dB degradation of performance from ideal, and very little difference with that of the averaging slicer, in an ideal channel. However Figure 4.6

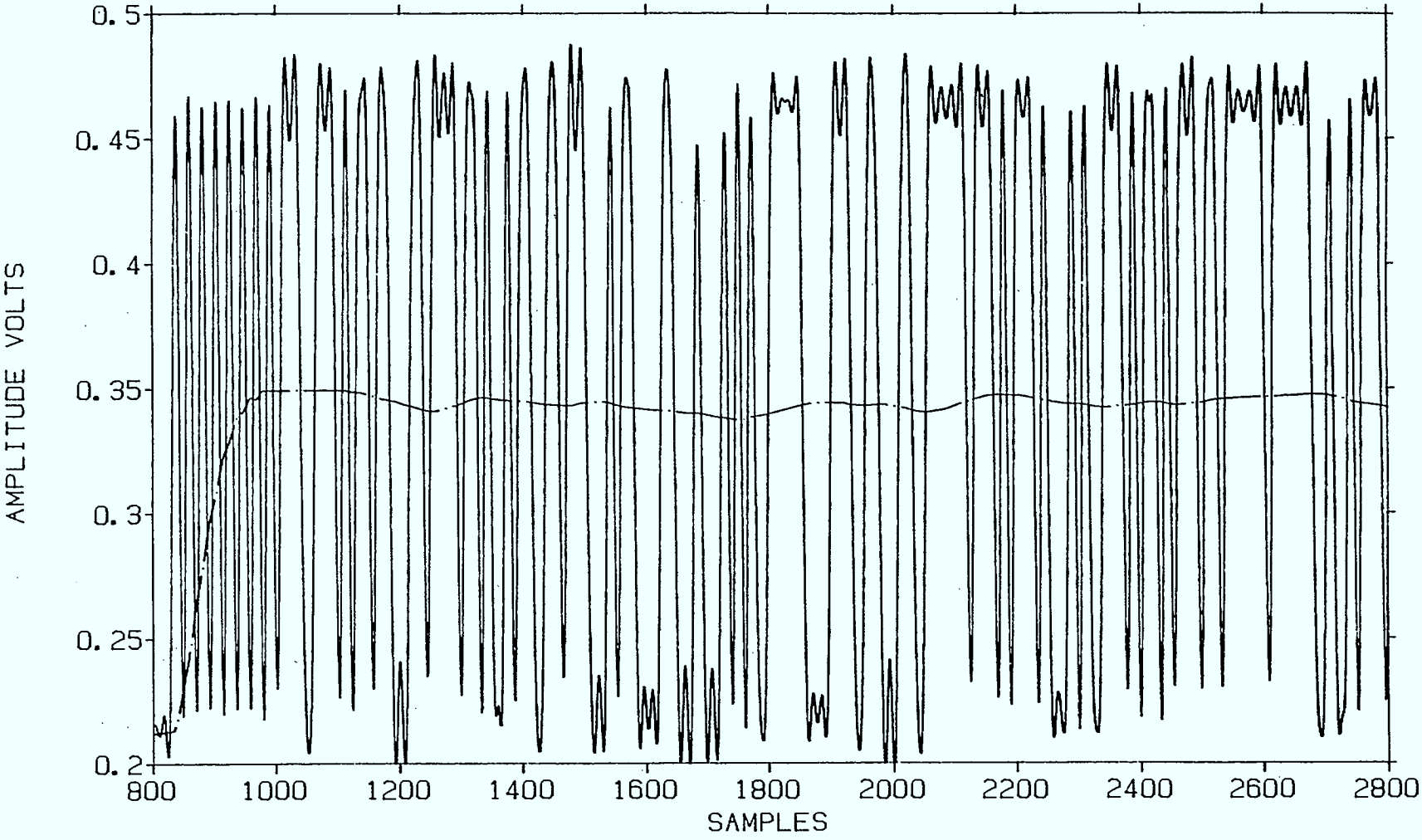


Figure 4.6 Telidon line response of adaptive slicer with wide bandwidth.

illustrates the behaviour of the adaptive slicer in a Telidon channel with no noise or multipath. As one can see the level of the adaptive slicer wanders up and down noticeably. This wandering is due to the intersymbol interference noise on the sample points, which effectively adds more noise to the slicing level, which degrades the performance more than expected. As a result, an alternative topology for the adaptive slicer was decided upon and this is shown in Figure 4.7. This more complex implementation requires two filters, the first to obtain an initial slicing level for word sync and initial data decisions, the second is the adaptive filter which tracks the average d.c. level throughout the line.

The bandwidth chosen for the narrower adaptive slicing filter was 90 kHz and it was implemented as a two pole Butterworth lowpass filter. The performance of this adaptive slicer in a Telidon channel is shown in Figure 4.8. The behaviour is much closer to the desired but one should note that with this narrower filter, the delay and therefore the response time to d.c. offsets produced by multipath is longer. (Rise time to a step response is on the order of 30 bit periods).

4.3.2 Analysis in a Multipath Channel

The objective of using an adaptive slicer is to track slowly varying "d.c." levels. These level variations are due to wide multipath spreads, that is, delayed and attenuated (using a complex coefficient) versions of the Telidon line are added to the original line. Since the sync pulse, the colour burst and the data stream are all at different d.c. levels, delayed versions will cause variations in the average d.c. level.

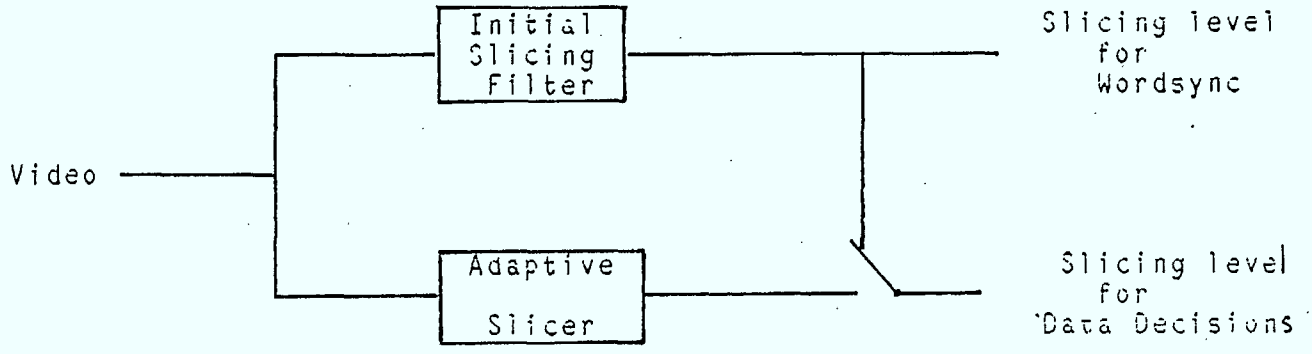


Figure 4.7. Alternative Adaptive Slicer Topology.

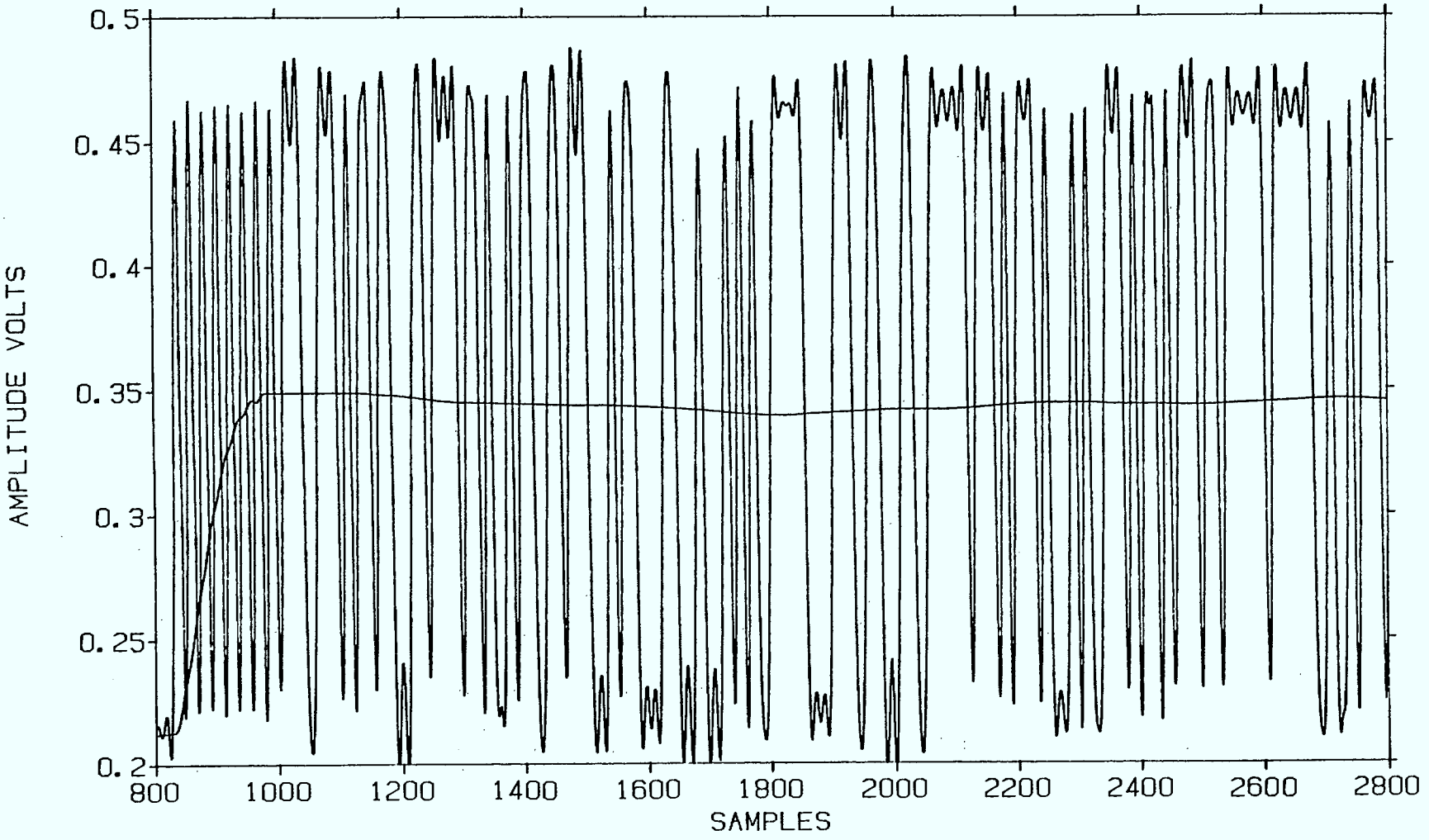


Figure 4.8 Telidon line response of adaptive slicer with narrow bandwidth.

As an example consider the multipath channel of PDUR=14 dB discussed in [1]. This is a particularly bad channel and the three dominant components of the multipath are shown in Table 4.1. By discussing this case in particular, one can obtain a general idea of when an adaptive slicer will provide some improvement.

The first component of the multipath in Table 4.1 is delayed only a fraction of a bit period. This type of close-in echo will not cause the d.c. level of the Telidon line to vary. It will cause an offset from the nominal level but this offset will be constant throughout the line and the initial averaging filter should be able to estimate it very well. In this case, the adaptive slicer would offer no advantage in offsetting the degradation in performance caused by the multipath.

The second component of the multipath in Table 4.1 is delayed two and half bit periods. This type of close-in echo will cause a slight jump in the d.c. level just after the beginning of the clock sync signal but has no affect on the level for the rest of the line. An appropriate averaging slicing filter would correct for this offset during the clock run-in and thus an adaptive slicer would offer no advantage in this case either. A strong echo of this delay will cause the most degradation of the Telidon system performance.

The third component of the multipath in Table 4.1 is delayed twenty-five bit periods. This means the colour burst of the delayed signal will be superimposed on the clock sync of the original, and the clock sync of the delayed will be superimposed on the data sequence of the original, etc. The resulting Telidon signal should show a d.c. shift shortly after the end of the data preamble. That is, shortly after an averaging slicer would have its

Delay		Attenuation
(nsec)	(bit periods)	
42	.25	.08
411	2.5	.33
4125	25	.10

Table 4.1: Dominant Multipath Components in MTP1050E.14 Channel (PDUR=14 dB)

level frozen. It is in this case that an adaptive slicer offers an advantage, because it can track the change in d.c. level after the averaging filter has provided a first estimate.

From this discussion, we immediately see that only in the case of isolated long multipath delays is the adaptive slicer expected to be of advantage. Close-in echoes can not be handled by the adaptive slicer and will probably negate any advantage obtained at long echoes if both are present.

4.4 Simulation Results

The initial effort of the slicer simulation was to determine the improvement possible using the modified averaging slicer. Note that the receiver performance is also a function of the symbol synchronization technique used, thus when making comparisons one should take this into account. The first set of results are given in Figure 4.9 where the bit error rate performance is shown as a function of weighted video signal to noise ratio.

The lower curve in Figure 4.9 is the optimum BER curve. The curve labeled (MA-MZ) refers to the decoder with a modified averaging slicer and the modified zero-crossing BTR (as discussed in the next chapter). The curve labeled (PK-ZC) refers to the original peak detector slicer and zero-crossing BTR which was simulated in the previous report [1]. Both of these decoders are approximations to actual hardware implementations. The fourth curve in the figure, (PK-MZ), refers to the peak detector, modified zero crossing combination.

In comparing the two approximations of hardware decoders, (MA-MZ) and (PK-ZC) one notes the almost 3 dB improvement

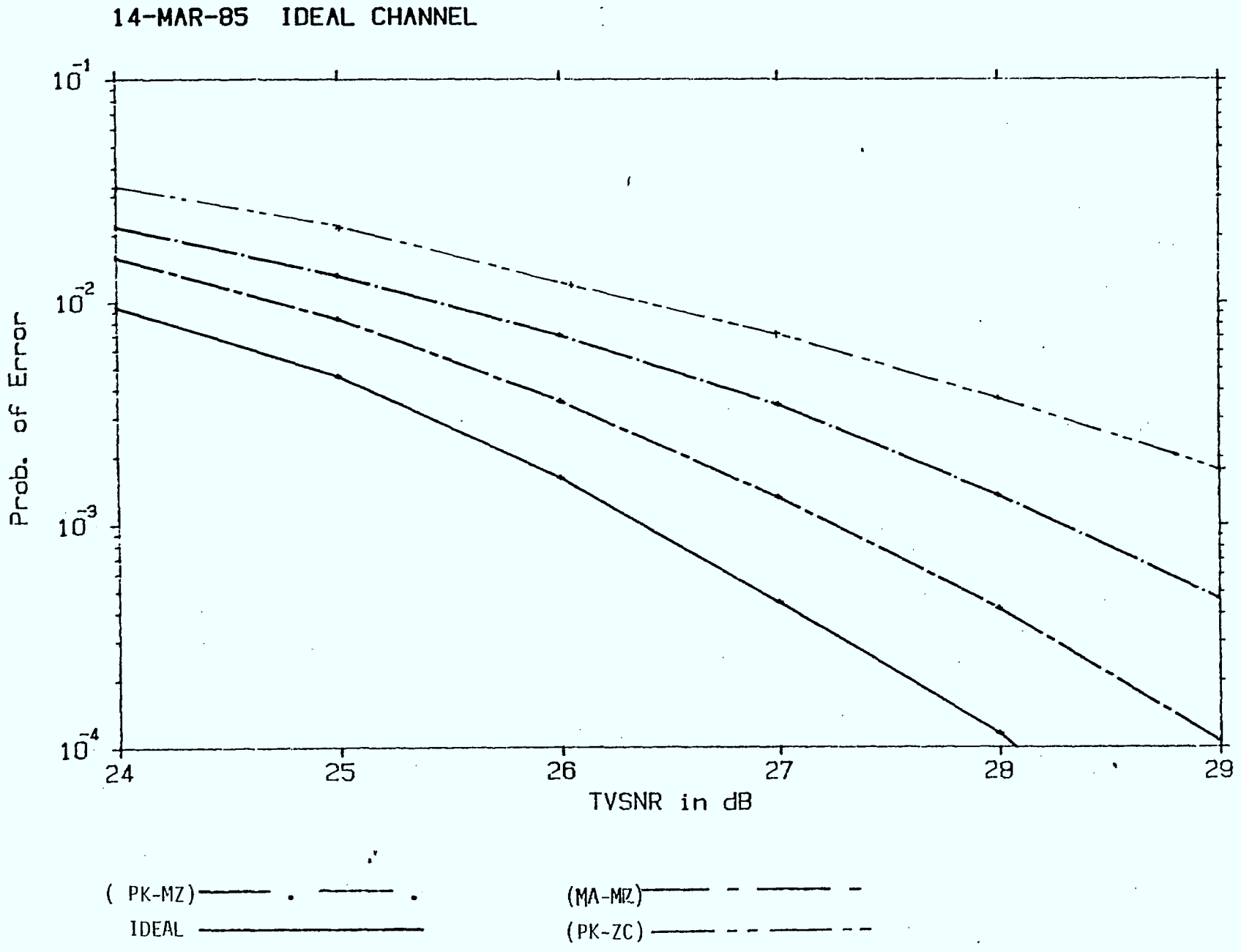


Figure 4.9 A comparison of performance of the different slicing techniques.

of the former over the latter. However, to determine how much of this is actually attributable to the slicing technique, one must compare the two curves with the same bit timing recovery technique, (MA-MZ) and (PK-MZ). In this comparison, we note that only about one half of the improvement can be attributed to the improvement in slicing technique. Still the improvement is considerable, and more in line with expected receiver performance.

In Figure 4.10, we have used the same notation except that we have introduced two new elements; the near optimum symbol synchronization technique (COR) developed in the next chapter and the wide bandwidth adaptive slicer (WAD) discussed in Section 4.3. Comparing the (WAD-COR) to the (MAV-COR) curve we see that the degradation in performance due to the wide bandwidth adaptive slicer is approximately 0.5 dB and decreasing at higher signal to noise ratios. As indicated before, this relative degradation of the wide bandwidth adaptive slicer is due mainly to the effects of ISI on the input samples.

Comparing the curve (MAV-COR) to the ideal curve, we note that is just slightly more than the 0.4 dB predicted in Section 4.2. This difference increases slightly at higher SNR, but as Chapter 5 will show this can be attributed to the non-ideal symbol synchronization. This corraboration of simulation with theory indicates that we have a good model of the behaviour of the modified averaging slicer. It also indicates that if implemented correctly, the modified averaging slicer should be near optimum for a white noise channel.

The narrower adaptive slicer (NAD) was first simulated in an ideal channel to insure that it did not provide a significant degradation from the modified averaging slicer, under these conditions. Figure 4.11 shows the bit error

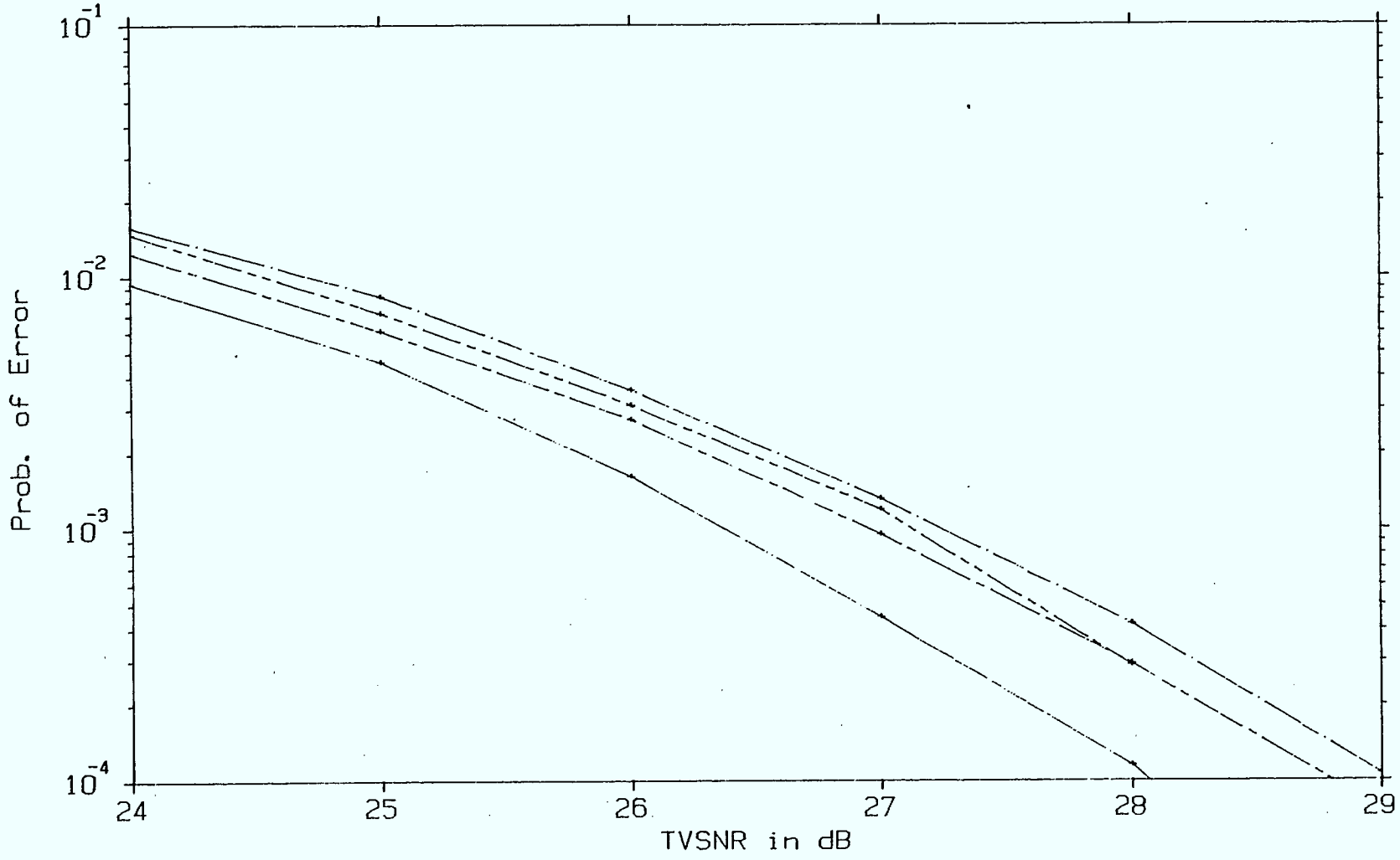


Figure 4.10 A comparison of performance of the modified averaging and wide adaptive slicers.

rate performance of the two slicing techniques versus the weighted video signal to noise ratio. To reduce the dependence of the receiver performance on the bit timing recovery techniques, all tests were done using the correlator BTR, which is described in the next chapter as being a near-optimal symbol synchronization strategy.

As can be seen in Figure 4.11, the adaptive slicer performs slightly better than the modified averaging slicer in a Gaussian white noise channel. This can be explained as follows. Both types of slicers start off with the same initial estimate of the slicing level, but for the averaging slicer the level is frozen and the expected variance of the level is proportional to the slicing filter bandwidth (300 kHz). In the adaptive slicer the slicing filter is changed to a 90 kHz filter, resulting in an improved estimate of the slicing level and an improvement in performance.

The performance of the two slicing techniques in a multipath channel are shown in Figure 4.12. The performance shown is the result of a number of simulation runs. There is very little difference between the two slicers. The close-in echoes in the multipath channel dominate performance so much that the choice of slicer, adaptive or otherwise makes little difference. This supports the discussion of section 4.3.2, and it indicates that those multipath cases in which the adaptive slicer provides an advantage are extremely limited.

4.5 Summary of Performance of Various Slicers

There were two slicers of interest discussed. The first was the modified averaging slicer which is an approximation to a hardware slicer. This slicing technique was shown to

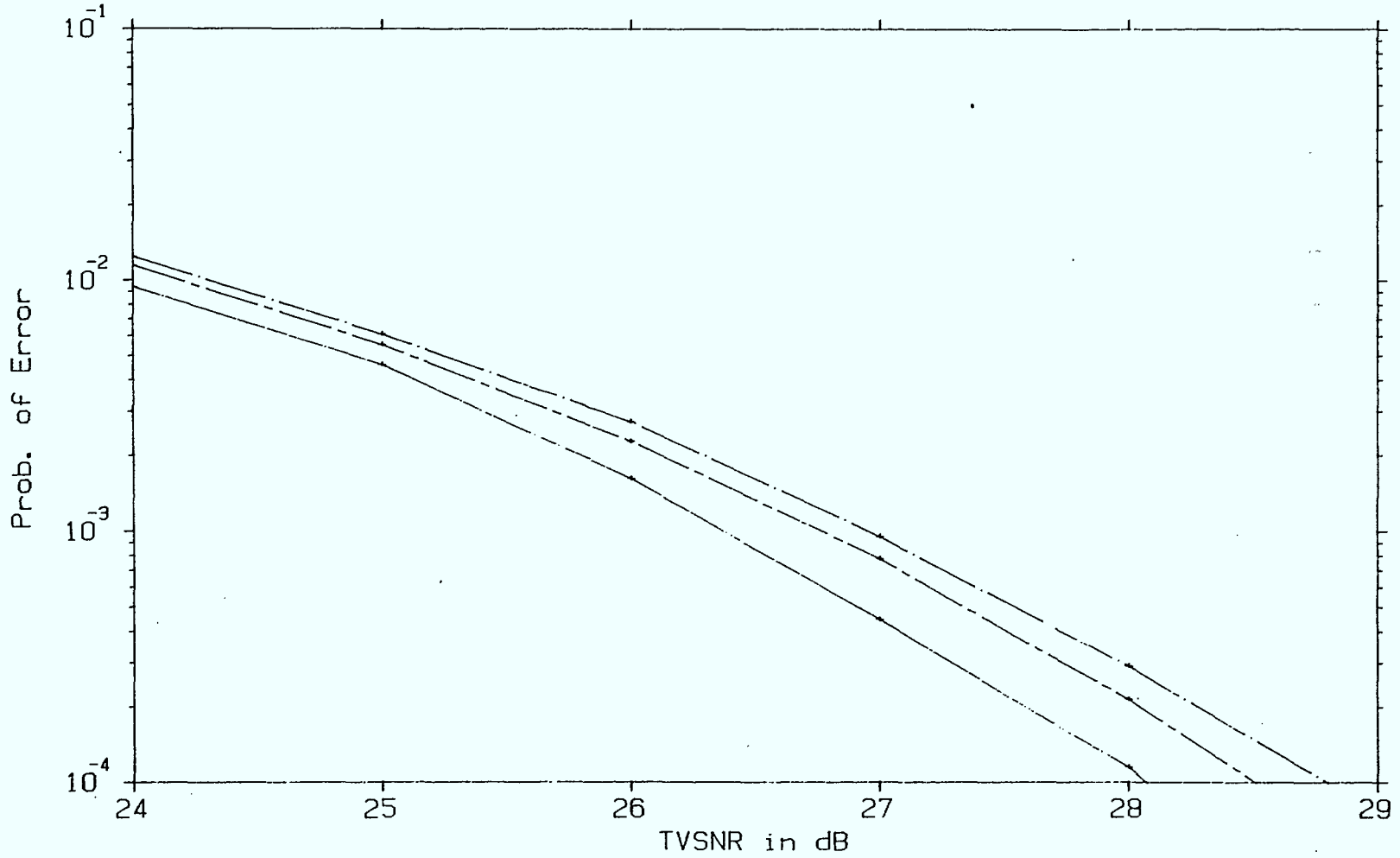


Figure 4.11 A comparison of performance of the modified averaging and narrow adaptive slicers

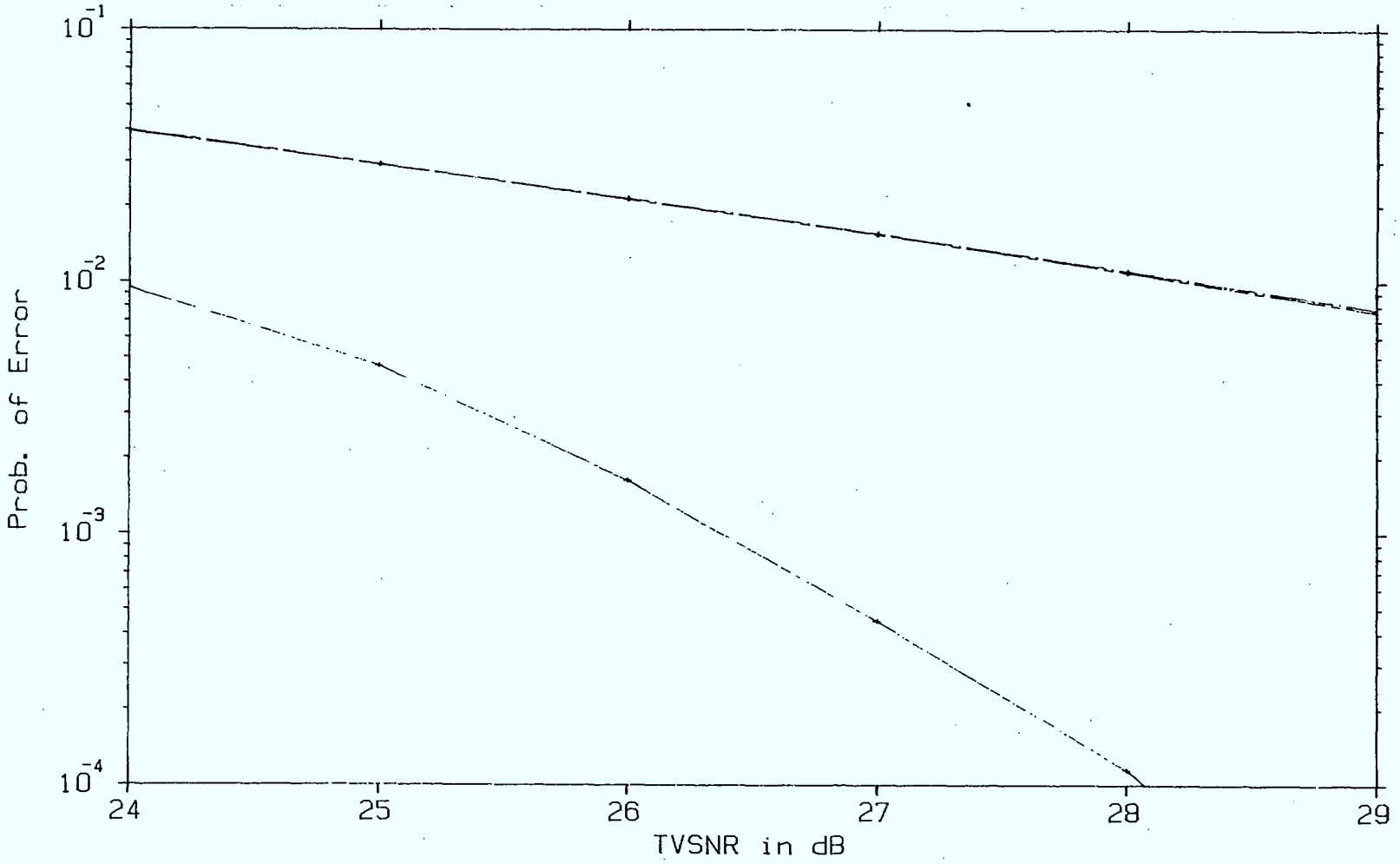


Figure 4.12 A comparison of performance of the modified averaging and narrow adaptive slicers in a multipath channel.

be near optimal on a white noise channel. On a large portion of multipath channels, its performance was no worse than other proposed techniques.

The second slicer discussed was adaptive. It was pointed out that the adaptive slicer will be of limited use on multipath channels. Those channels where it does provide an advantage are the ones with long multipath delays. However, as multipath delays become longer, the interference caused by the multipath is expected to become weaker. This means there is less expected degradation of the channel and the adaptive slicer provides less advantage.

Channels with long multipath delays are the most objectional from a viewer's standpoint because of the ghosting effect. Thus the adaptive slicer may only provide an advantage on those channels which are unacceptable for a video signal, which may make their use very questionable.

5.0 SYMBOL SYNCHRONIZATION ALGORITHMS

The objective of this task of the contract was the implementation of a suitable improved bit timing recovery (BTR) technique. The performance improvement that can be achieved is quantified by simulation and comparing the results with other techniques.

As mentioned in the previous chapter, the bit error rate (BER) performance is dependent on both the slicing level algorithm and symbol synchronization. Thus one must be careful to keep the slicing level constant when comparing between the different techniques.

This chapter is organized into five sections. The first section reviews existing BTR algorithms, sections two and three discuss an existing hardware technique and a new technique, a modified version of the zero-crossing BTR and a BTR scheme based on a correlator. The fourth and fifth sections compare the analytical and simulation performance of the different techniques.

5.1 Existing Techniques

In the existing Telidon simulation program, there were two techniques of clock synchronization. One was ideal which provided a upper bound on performance, but obviously is not typical of a realistic circuit. The second BTR technique was based on a Telidon receiver in which the slicing level determination and bit timing recovery were coupled together.

In this receiver, the slicing level was first estimated and then a zero crossing detector was used to choose the one out of five clock phases which was closest to the zero

crossing. The performance of this technique is discussed in [1] but there are two serious problems with this technique.

- An error in the slicing level will result in an error in the clock phase.

- The clock phase is estimated from a single zero crossing. Thus only a small fraction of the signal energy in the two byte clock sync signal is being used to estimate the clock phase. The estimated clock phase tends to be very sensitive to noise and other forms of interference.

The techniques explained in the following two sections are an attempt to overcome these problems.

5.2 Modified Zero Crossing BTR

The first point to be noted about the existing BTR scheme, is that there is no need for the slicing level estimation and the clock phase estimation to be so dependent upon each other. During the clock sync signal, the slicing level circuitry is attempting to estimate the d.c. level of the signal, while the clock phase circuitry is attempting to estimate the phase of the 2.863636 MHz clock signal.

The obvious improvement that can easily be made is to bandpass filter the received signal before estimating the clock phase. The specific technique of this decoder is to bandpass the received signal and then choose the clock phase which is closest to the sixth negative going zero crossing of the filtered signal. The phase is chosen as one of ten possible discrete clock phases.

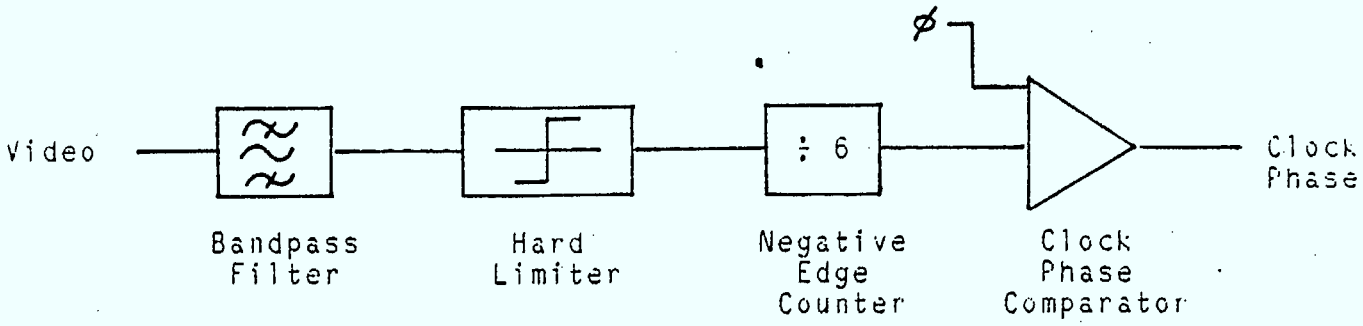
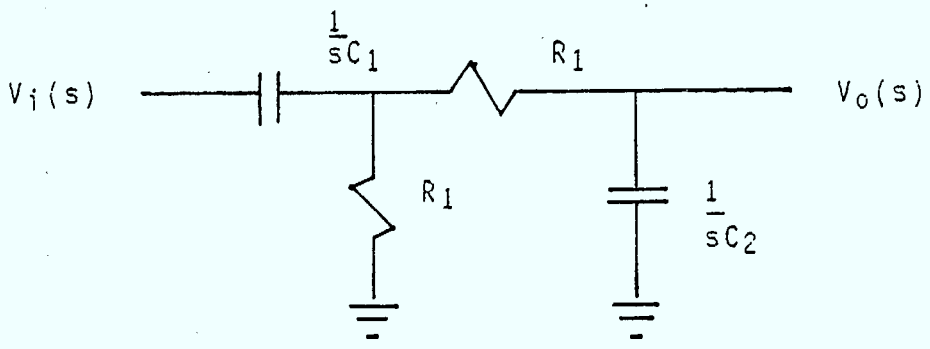


Figure 5.1. Block Diagram of Modified Zero Crossing BTR.



$$T(s) = \frac{V_o(s)}{V_i(s)} = \frac{sC_1R_1}{s^2R_1R_2C_1C_2 + s(R_1C_1+C_2(R_1+R_2)) + 1}$$

$$T(\omega) = \frac{j\omega C_1R_1\omega_o^2}{(\omega_o^2 - \omega^2) + j2\zeta\omega_o\omega}$$

where

$$\omega_o = 1/(R_1R_2C_1C_2)$$
$$= 2\ (2.863636\ \text{MHz})$$
$$2\zeta = \omega_o[R_1C_1+C_2(R_1+R_2)]$$
$$= 2.0$$

Figure 5.2. Bandpass Filter of Modified Zero Crossing BTR.

A block diagram of the BTR scheme is given in Figure 5.1. The bandpass filter simulated is shown in Figure 5.2. This filter is simpler, than the actual hardware implementation but the transfer characteristic and performance should be similar. It has the added advantage that with proper choice of components it introduces no delay of the 2.863636 MHz clock sync signal.

With the parameters chosen as shown in Figure 5.2, this filter has a low Q (on the order of 1) and thus does not remove a large portion of the noise, its most beneficial effect would seem to be the removal of the low frequency noise as well as the d.c. Although this filter has a very different circuit implementation than the circuit which does the same duty in the corresponding hardware decoder, the amplitude response would seem to be very similar over the 4 MHz video bandwidth.

Although this approach does provide a solution to problem of minimizing the dependence of the BTR performance on the slicing level, it leaves the fact that the clock phase is determined from a single zero crossing. However one would expect some reduction in the effects of noise and interference on the estimate because of the bandpass filtering.

5.3 Correlator BTR

In an attempt to overcome the limitations of using a single zero crossing to estimate the clock phase, a bit timing recovery scheme was developed based on correlating the received preamble clock with the different phases of the receiver clock. The correlation process is shown in Figure 5.3. It includes a bandpass filter and limiter the same as in the modified zero crossing method of Section 5.2. The

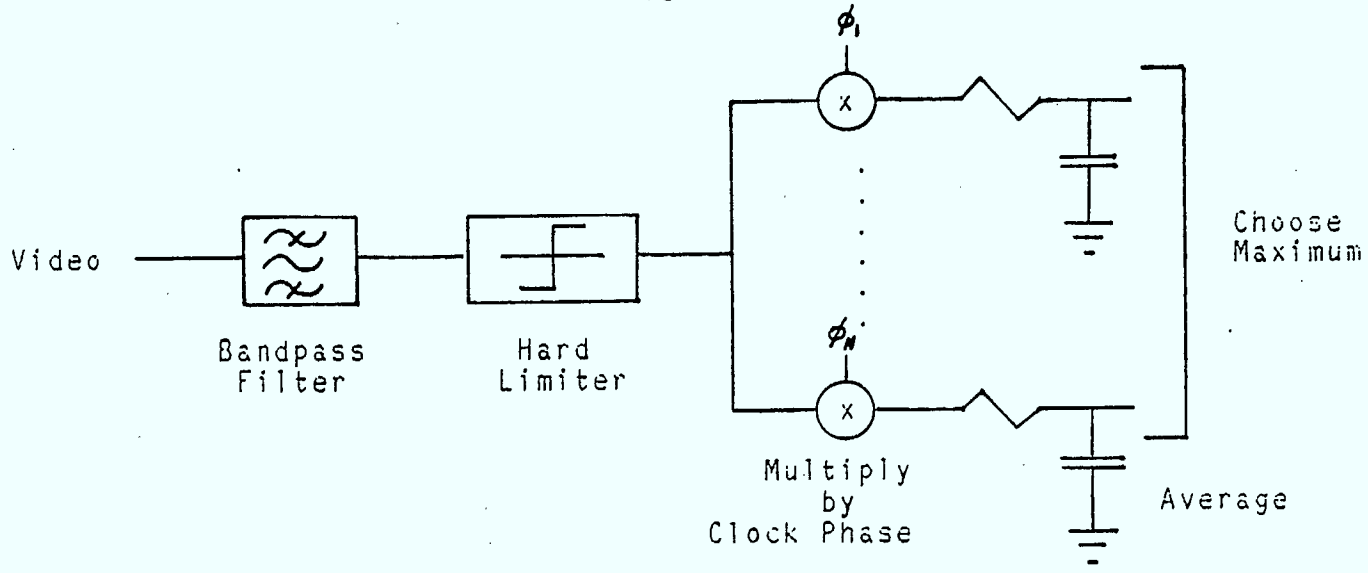


Figure 5.3. Block Diagram of Correlator BTR.

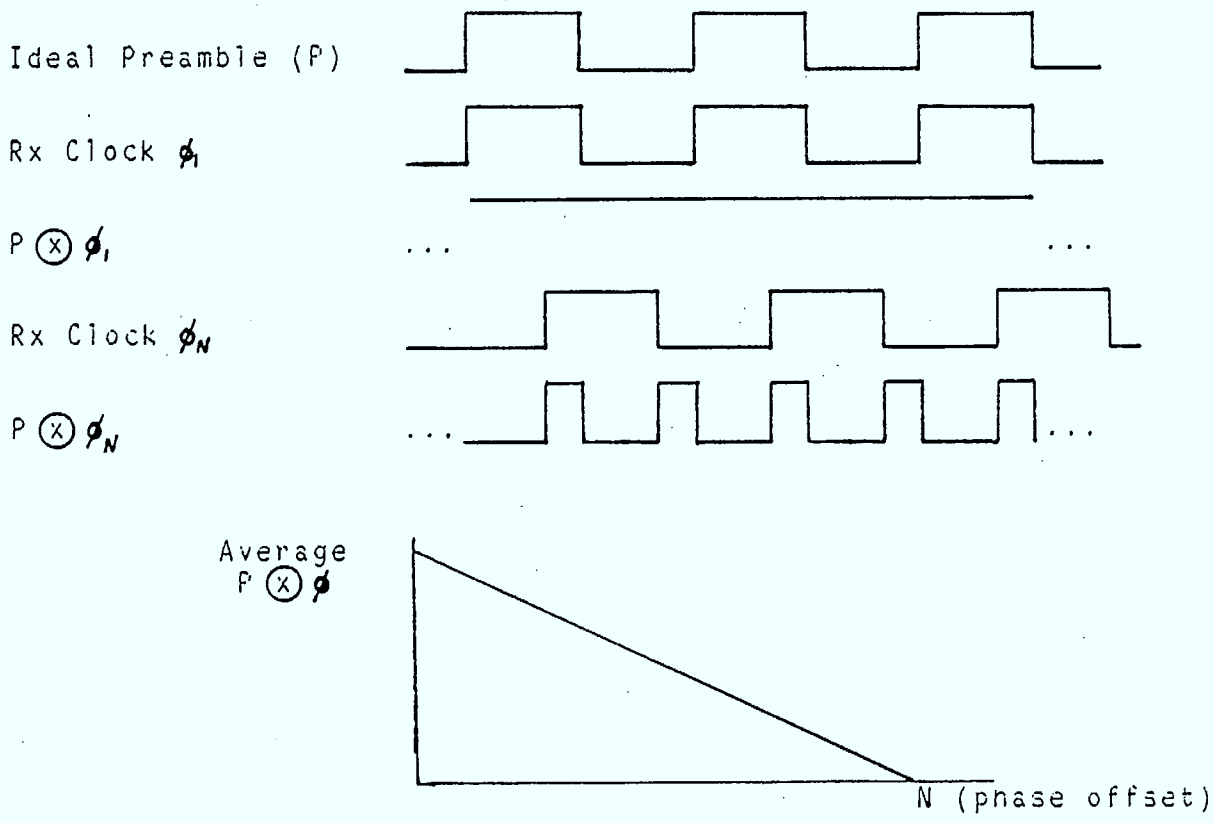


Figure 5.4. Correlation Behavior of Correlator BTR.

limiter is then fed to a bank of correlators, consisting of multipliers followed by averaging lowpass filters. The multipliers would be implemented digitally as exclusive NOR gates.

For maximum correlation, the receive correlation clock should be at the same frequency as the clock preamble (2.863636 MHz), half of the bit rate. It had been originally suggested that the output of the averaging be compared to a threshold, and to choose the clock phase which is the first to cause the threshold to be exceeded. However, it may be more prudent because of the potential effects of multipath and noise to choose the actual maximum of the correlator outputs after a fixed length of time.

An example of the action of multipliers, and a plot of the ideal correlation output as a function of clock phase is shown in Figure 5.4. The linear relationship indicates that the clock phases should be uniformly spaced to minimize the potential error. The number of clock phases was chosen to be five, to be consistent with a previous circuit as well as to provide a circuit which can be implemented with a relatively low component count. A hardware block diagram of how the decision making portion of this correlator based bit timing recovery scheme might be implemented is shown in Figure 5.5.

5.4 Performance Analysis

This section does an approximate analysis of the effects of three different bit timing recovery schemes, zero-crossing, modified zero-crossing and correlator, on the bit error rate performance of the Telidon system.

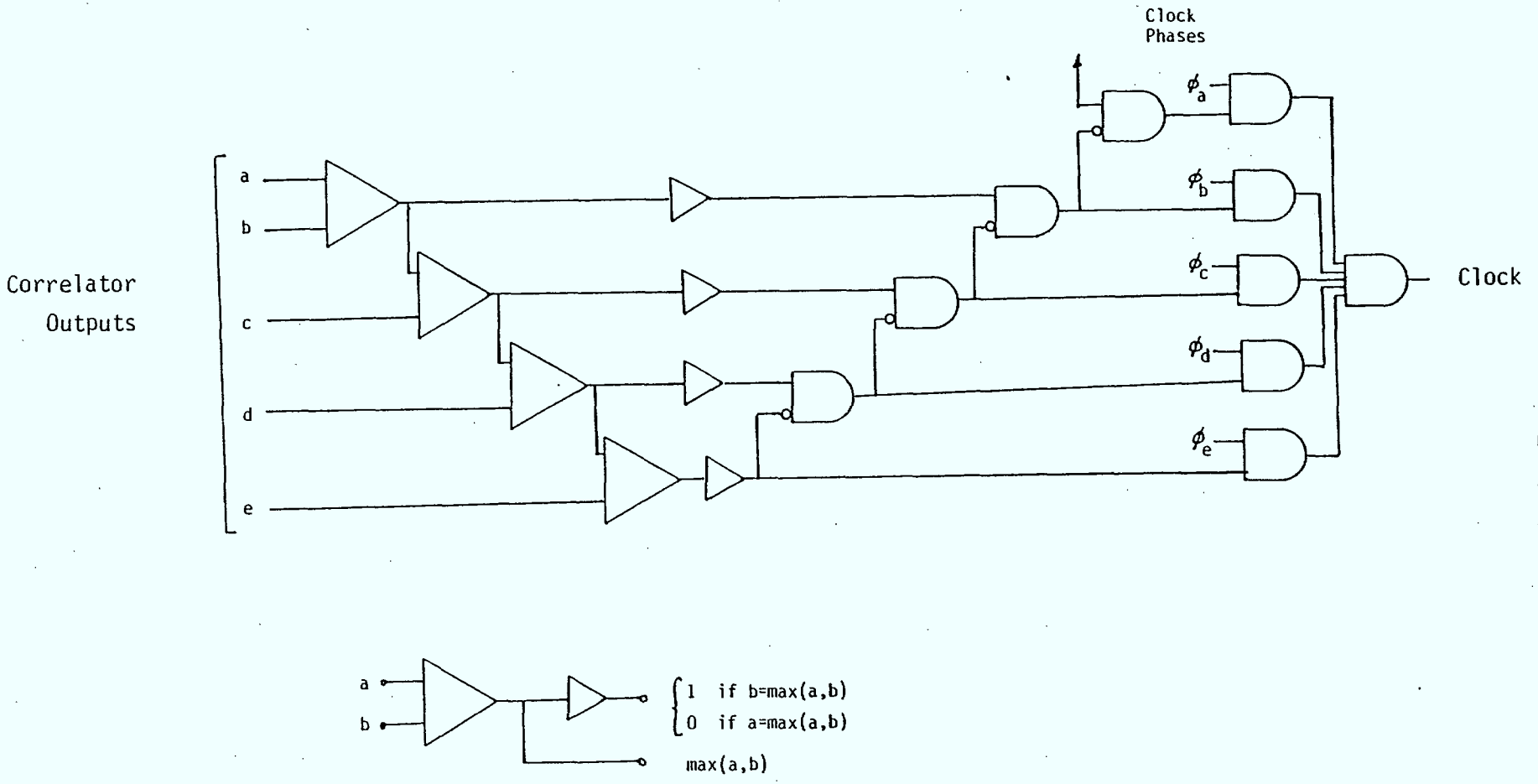


Figure 5.5 Clock Decision Hardware Block Diagram.

The objective of the bit timing recovery circuitry is to estimate the phase of the clock in the clock sync portion of the preamble sequence, this allows the appropriate sampling of the received data sequence. Since the object of interest is the clock phase, the main degradation in its estimation is due to phase noise on the clock sync signal.

If we consider the clock sync portion of the data line in isolation (and a.c. coupled) then it can be considered as a bandpass signal centred around the clock sync frequency of 2.863636 MHz, with additive white noise. This is illustrated in Figure 5.6a. In a bandpass system, the phase noise caused by additive white noise is mainly due to the quadrature component of the noise [4]. In fact the phase noise is approximately

$$\phi_e(t) = \frac{n_q(t)}{A}$$

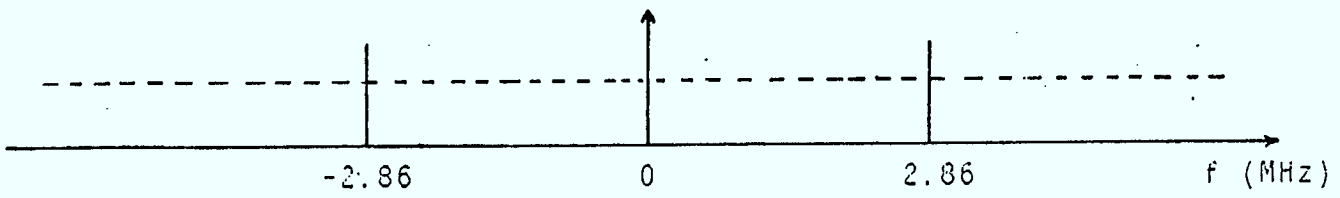
where $n_q(t)$ is the quadrature component of the noise, and A is the received signal amplitude. Note that if $E\, n_q^2(t)=\sigma^2$ then the variance of the phase noise is

$$\sigma_e^2 = \frac{\sigma^2}{A^2}$$

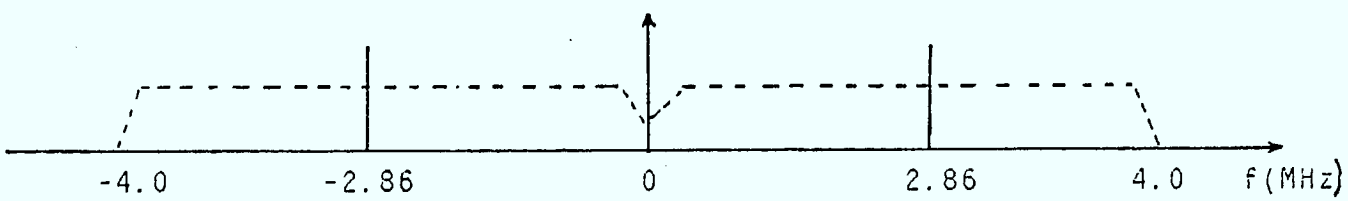
which is approximately

$$\sigma_e^2 = (SNR_0)^{-1} \approx \left(\frac{2}{L_m}\frac{E_b}{N_o}\right)^{-1}$$

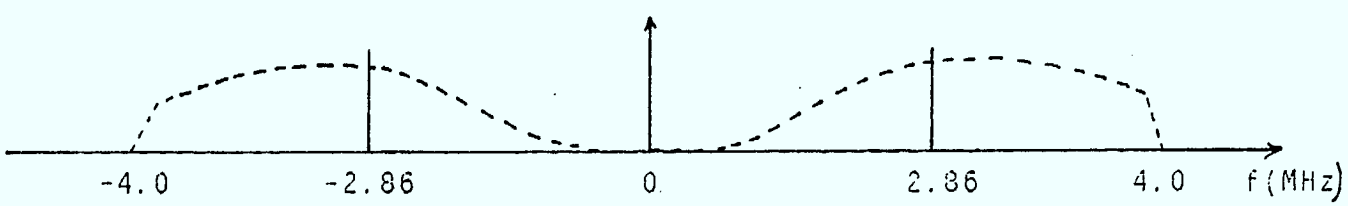
where SNR_0 is the digital signal to noise ratio as defined in [2] and L_m is a weighting factor dependent on the pulse shaping. (L_m = 1.04 for 100% raised cosine pulse shape).



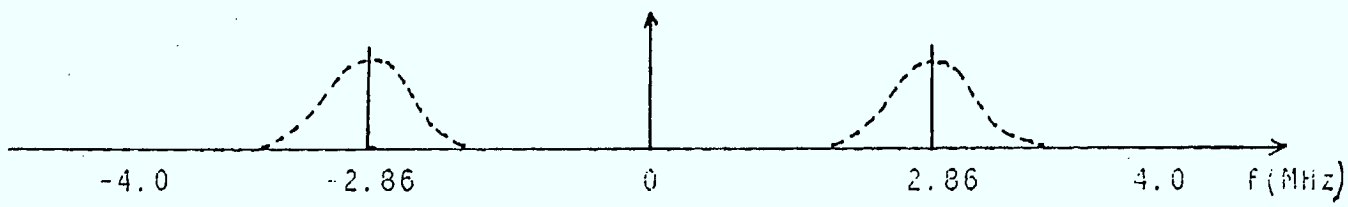
a) Bandpass signal representation of clock sync plus white noise.



b) Filtering of Zero-Crossing BTR scheme.



c) Filtering of Modified Zero-Crossing BTR scheme.



d) Filtering of Correlator BTR scheme.

Figure 5.6. A comparison of the noise performance of the different BTR schemes.

The objective of the different BTR schemes is to reduce σ_e^2 as much as possible. The zero crossing scheme simply uses the television receiver filtering to reduce noise as shown in Figure 5.6b. Since this filtering is also applied to the noise on the data, this provides the baseline case with

$$\sigma_e^2 \approx (SNR_0)^{-1}$$

The modified zero-crossing technique inserts a narrower bandpass filter around the clock sync signal to provide a noise advantage as illustrated in Figure 5.6c. The advantage is proportional to the bandwidth advantage of the bandpass filter. The noise advantage where the video bandwidth B_v (assuming white noise) for this filter is

$$N_a = \frac{B_v}{\int_0^{N_v} |T(\omega)|^2 \, d\omega} = 1.66 \text{ dB}$$

The correlator based BTR technique can be considered as convolving the received signal with the impulse response

$$h(t) = \begin{cases} \sin \omega_c t & 0 \leq t \leq T_c \\ 0 & t > T_c \end{cases}$$

which has the amplitude response

$$H(f) = -je^{j(\omega_o-\omega)T_c/2} \left(\frac{\sin(\omega-\omega_o)T_c/2}{(\omega_o-\omega)}\right)$$

$$+je^{-j(\omega_o+\omega)T_c/2} \left(\frac{\sin(\omega+\omega_o)T_c/2}{(\omega+\omega_o)}\right)$$

which is a narrow bandpass filter centred on the clock sync frequency as shown in Figure 5.6d. The bandwidth of this filter is approximately $2/T_c$, where T_c is the duration of the correlation process (approximately 14 bit periods) so that the bandwidth advantage of this case over the base line is

$$N_a = \frac{B_v}{(2/T_c)} = \frac{4.0 \text{ MHz}}{(2 \cdot 5.72 \text{ MHz}/14)}$$

$$= 6.9 \text{ dB}$$

The expected BER performance of an ideal channel is a function of σ_e/T illustrated in Figure 5.7. The corresponding σ_e/T values for the three techniques discussed are shown in Table 5.1. Note that σ_e is in radians and that π radians corresponds to a bit period (T) since the clock sync signal runs at half the bit rate.

The performance curves for these three different BTR techniques are penciled in on Figure 5.7. As upper curve indicates the original zero-crossing BTR could cause a 2 dB degradation of performance at low E_b/N_o. The modified zero crossing approach should improve performance by between 0.5 and 0.75 dB, while the correlation BTR will add an additional improvement of approximately the same amount. Note that higher E_b/N_o all three BTR curves approach the optimum.

The curves shown in Figure 5.7 are for continuous clock phase distributions. In the Telidon system, the clock phase is chosen from among a number of discrete clock phases. This implies that there is an additional factor to consider in the variance of clock phases. The receiver clock is not synchronized with respect to the data,

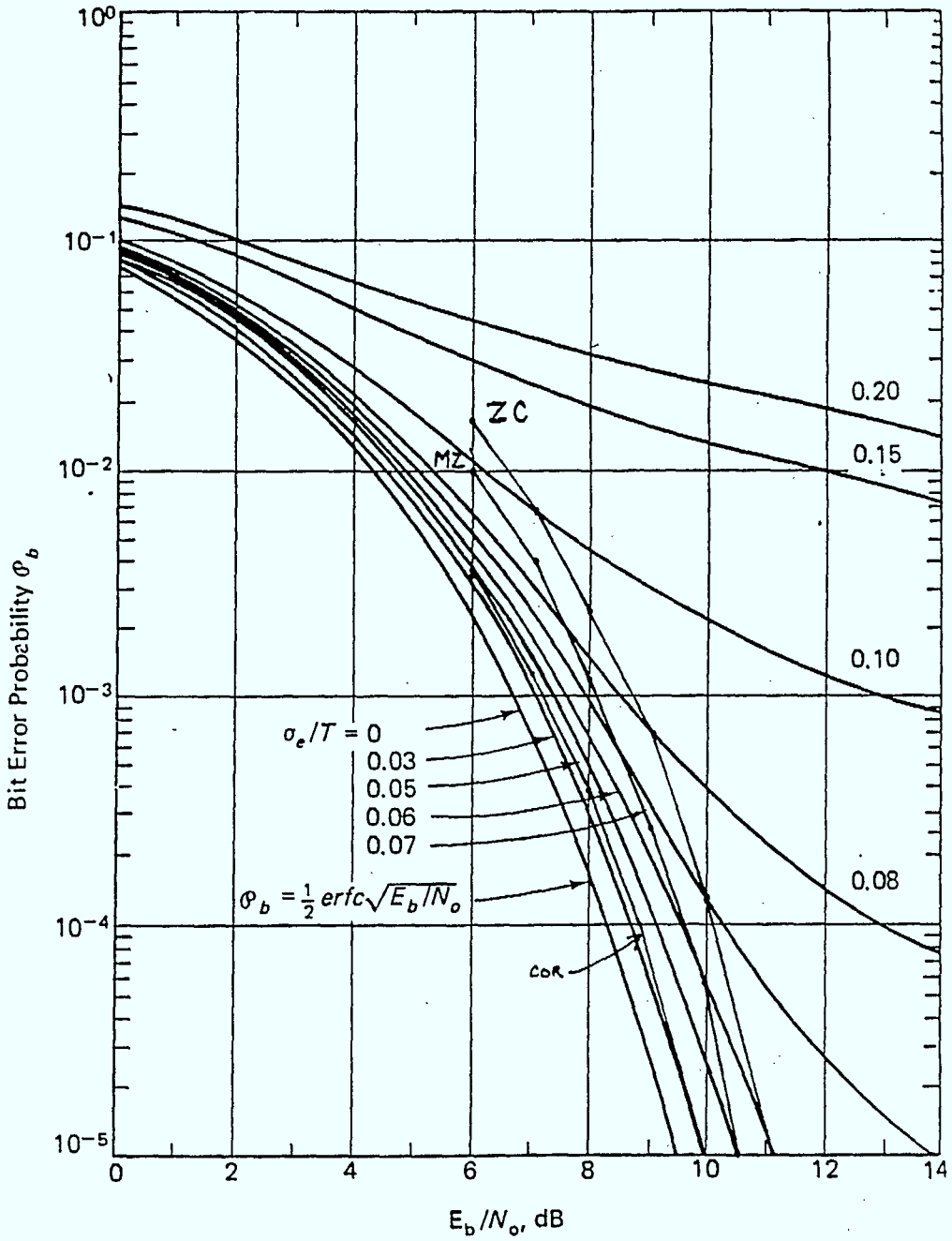


Fig. 5.7 Average probability of bit error versus E_b/N_0 with standard deviation of the symbol sync error σ_ϵ as a parameter (NRZ) [From Lindsey and Simon, 1973, 469, Fig. 9-37]

E_b/N_o (dB)	Zero-Crossing		Modified Zero-Crossing		Correlator	
	σ_e	σ_e/T	σ'_e	σ'_e/T	σ''_e	σ''_e/T
6	.36	.115	.297	.095	.163	.052
7	.322	.102	.265	.085	.145	.046
8	.287	.09	.237	.075	.130	.041
9	.256	.08	.211	.067	.166	.037
10	.228	.07	.188	.060	.103	.032
11	.203	.06	.168	.053	.092	.029

(uses relationship $\sigma_e^2 \approx \left(\frac{2}{L_m}\frac{E_b}{N_o}\right)^{-1}$, $L_m \approx 1.04$)

Table 5.1: Normalized Phase Deviations for Different BTR Schemes.

initially, so that given a zero crossing, ϕ_z, the closest clock phase from the set of discrete clock phases should be uniformly distributed about this point, that is, if ϕ is the phase of the clock chosen then

$$\rho(\phi|\phi_z) = \begin{cases} \frac{N}{\pi} & \phi_z - \frac{\pi}{2N} < \phi < \phi_z + \frac{\pi}{2N} \\ 0 & \text{otherwise} \end{cases}$$

where π radians corresponds to T seconds and N is the number of clock phases. One could assume that the zero crossings were Gaussian distributed, so that the zero crossing distribution is

$$\rho(\phi_z) = \frac{1}{\sqrt{2\pi}\sigma_e} \; e^{-\phi_z^2/2\sigma_e^2}$$

and then using Bayes theorem, the clock phase distribution is

$$\rho(\phi) = \int_{-\infty}^{\infty} \rho(\phi_1, \phi_2) \, d\phi_2$$

$$= \int_{-\infty}^{\infty} \rho(\phi|\phi_z) \, \rho(\phi_z) \, d\phi_z$$

and we are interested in the variance of the resulting clock phase distribution

$$E\phi^2 = \int_{-\infty}^{\infty} \phi^2 \rho(\phi) \, d\phi.$$

However by noting that the error on the resulting clock phase is

$$\phi = \phi_z + \phi_c$$

where ϕ_z is the error introduced by the noise on the zero crossing and ϕ_c is the error introduced by the discrete clock phase, and that these are independent, so

$$E\phi^2 = E\phi_z^2 + E\,\phi_c^2$$

and that if ϕ_c has a uniform distribution over $[-\pi/2N, \pi/2N]$ then

$$\sigma_e^2 = E\phi^2 = \sigma_z^2 + \frac{1}{12}\left(\frac{\pi}{N}\right)^2$$

where π is the bit period and N is the number of clock phases. Evaluating the phase deviation introduced by the discrete clock phases for N=5 and N=10 we obtain

N	$E\phi_c^2$(rad)	σ_e/π
5	.032	.058
10	.008	.029

Note that the phase variance introduced by the discrete clock phases is fixed and does not decrease with increasing E_b/N_o. This implies that this is a constant degradation of performance, and for the case N=5 (σ_e/π corresonds to σ_e/T in Figure 5.7) the resulting phase variance will cause a degradation of 1.5 dB in performance at higher E_b/N_os.

The results of this section should be interpreted more in a qualitative manner rather than quantitatively. The analysis includes a number of assumptions about the channel and data detector, and the performance curves of Figure 5.7 contain some hidden assumptions about the distribution of phase errors. However, it should reflect the relative performance gains expected from the various techniques and their limitations.

5.5 Performance Simulation

One difference between the practical situation described in Section 4.4 and that which was simulated was that in the simulation the discrete clock phases were only allowed to vary randomly by an integer number of samples with respect to the signal. As a result only discrete phase errors can occur. This still allows a reasonable representation of the effects of phase noise on the sampling times (the continuous representation is replaced by a discrete distribution, i.e., a histogram with intervals centred on the sample points.) but the effects of discrete clock phase are somewhat distorted. Fortunately the former effects dominate the later over most of the range of E_b/N_o considered, however one may still expect a shift in the performance from that predicted in the previous section.

5.5.1 BER Performance

The bit error rate (BER) performance of the different BTR schemes is a strong function of the slicing level algorithm so any performance comparisons must be done with a fixed slicing level algorithm.

The following Figure 5.8 shows the performance of the three BTR algorithms with the modified averaging slicer in a Gaussian noise channel. From this figure one notes that as one progresses from the zero crossing to the modified zero crossing to the correlator there is about a one half dB improvement at each stage. The 90% confidence limits on these curves are approximately ±10% of the BER value. This half dB improvement with each improvement on the BTR scheme is consistent with that predicted by analysis. The best of the techniques is the one based on the correlator and it is about one half dB from optimum over this range of E_b/N_o, very close to what was predicted from analysis. The confirmation of the analytical results by the simulation, would indicate that for an ideal channel the modified averaging slicer is very nearly optimum. This is because the differences of the curves in Figure 5.8 from the ideal curve can be mainly attributed to the bit timing recovery errors. Note that at higher video signal to noise ratios, the differences with ideal become larger; as discussed in Section 5.4, this can be attributed to the effects of discrete clock phases beginning to dominate the effects of noise.

In Figure 5.9, the performance of two BTR techniques with the peak detector based slicer are shown. The first thing one notes is the deterioration in performance due to the much poorer slicing technique. Secondly, the combination of the peak detector slicer and zero crossing BTR, degrades performance even more than might be expected, this is because of the coupling of the two processes in one version of the receiver hardware. However the other technique does seem to have the relative difference predicted by analysis.

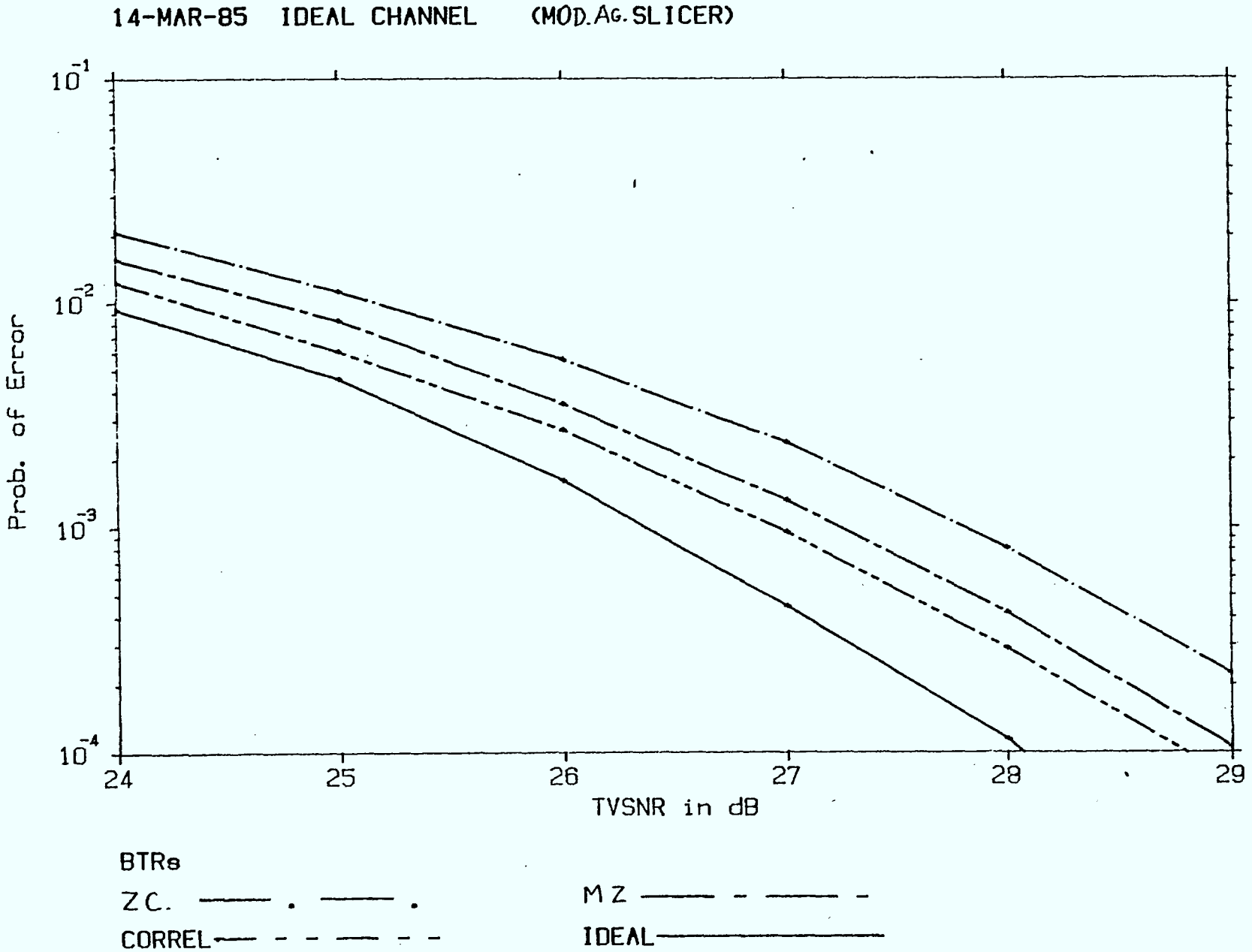


Figure 5.8 A comparison of the different BTR techniques using the modified averaging slicer

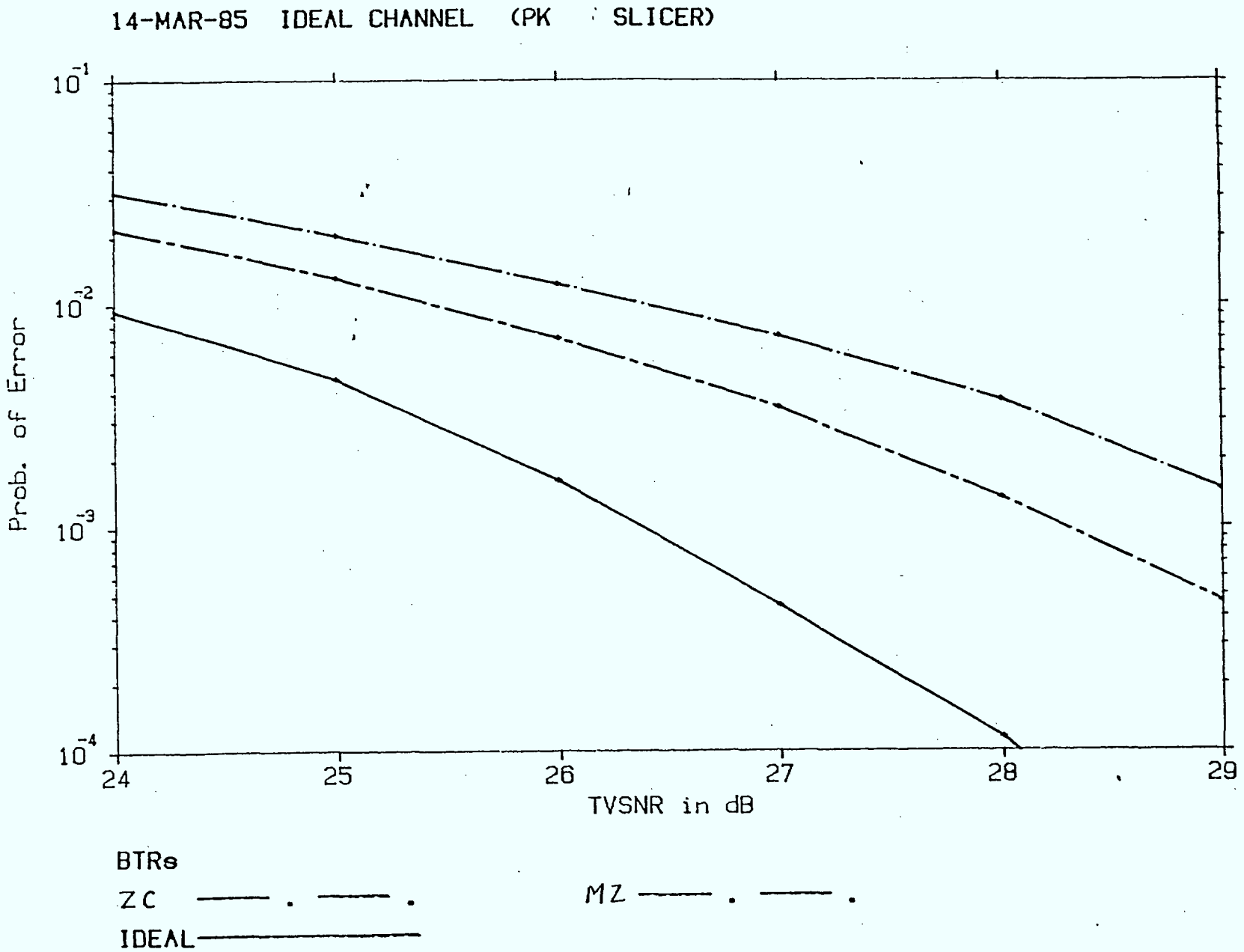


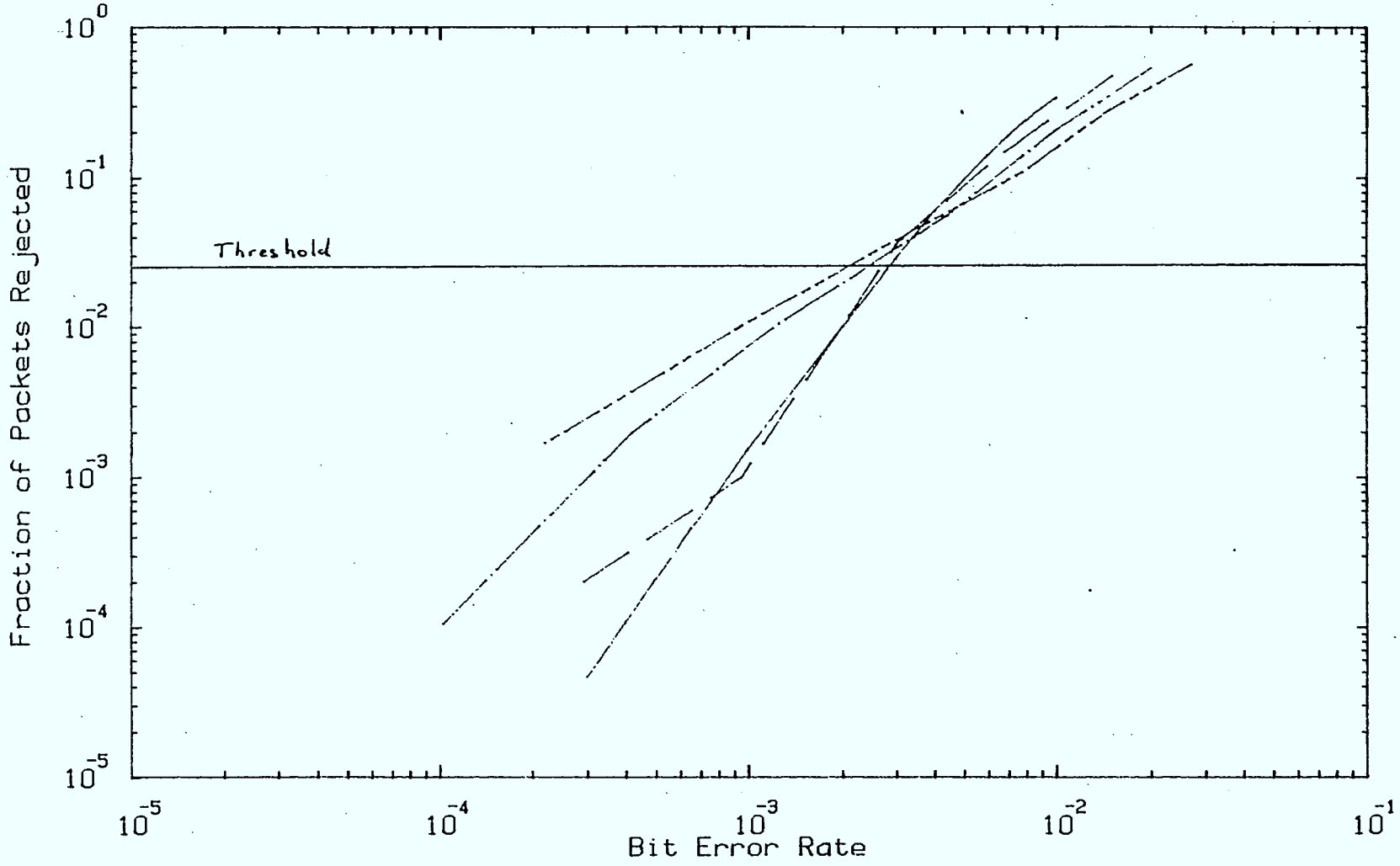
Figure 5.9 A comparison of the performance of the different BTR techniques using the peak detector based slicer.

5.5.2 Coding Performance

In addition to gross BER performance, of particular interest in the Telidon channel is the coding performance. In particular, a correlation in the bit errors may degrade the performance gain expected from the coding algorithm. Inherent in the Telidon channel, one may expect some correlation among errors due to the fact that the slicing level and clock phase are frozen for the length of a packet. Over the long term, these effects will average out and the gross BER performance should approach that of independent errors. However, there is still the possibility of short term correlation of errors in packets. This can cause a degradation in the coding performance because the coding algorithm works on a packet basis.

To exhibit the dependence of the "C" coding algorithm on the BTR scheme, Figure 5.10 shows the fraction of packets lost as a function of the input BER to the decoder. Also shown on the graph is the theoretical performance of the decoding algorithm in a white Gaussian channel with independent errors.

There are several interesting things to note about the curves in Figure 5.10. The fraction of packets lost as a function of BER (not E_b/N_o) is very similar for both the zero crossing and modified zero crossing BTR schemes, at high BER rates. In fact they both cross the ideal curve at BERs greater than 4×10^{-3} and provide better performance in terms of fewer packets rejected at high BER. This type of behaviour would tend to indicate that bit errors are bursty or correlated and are more likely to appear in the same packet. This can be explained by the fact that both of these methods use a single zero-crossing to estimate a



Fraction of Packets Rejected as a Function of BER
for Various BTRs (Slicer is modified averaging technique)

IDEAL _____ ZERO-CROSSING ___ _ _ ___ _ _

MOD. ZERO-CROSS. _____ . _____ . CORRELATOR _____ _____

Figure 5.10

clock phase which is used for the whole packet. A packet with a bad estimate of the clock phase is liable to have a number of errors. This also explains the much poorer performance of these two techniques at lower BERs. Although errors and bad clock phases are less likely at lower bit error rates, when a bad clock phase estimate does occur, it is liable to introduce a number of errors into a packet, more errors than the decoder can correct and so the packet must be rejected.

The results using the correlator BTR scheme follow nearly exactly the theoretical curve until low BER. At low BERs, the effects of discrete clock phase is causing the degradation from theoretical. The fact that this scheme follows the theoretical curve so well confirms the explanation of the behaviour of the other two schemes described above.

The threshold shown on Figure 5.10 refers to the maximum acceptable fraction of packets which can be rejected. Note that BERs where the zero crossing schemes perform better than "theoretical" are above this limit and therefore do not deserve much consideration. Below this threshold there is a significant degradation of performance using the zero-crossing schemes. At BER of 5×10^{-4}, ten times more packets are rejected when these schemes are used than in the theoretical case. This is a very significant loss in performance and would seem to wipe out a large portion of the coding advantage.

To achieve the coding gain which was expected using the C code, these curves would seem to indicate that one has to use a more complex BTR scheme such as the correlator, in order to provide a reasonably white error process. The alternative to this is to vertically encode the sequence,

that is, to code across packets as in the bundle code rather than along the packet. If the codeword consists of one bit or possibly one byte from each packet for a number of packets, then the error process (since the clock phase errors are random) is likely to be more white.

However vertical encoding alone will upset the present format of the Telidon signal. If the bundle code is chosen, which performs C coding of packets both horizontally and vertically, then the above discussion would suggest doing the vertical decoding first where the errors should be reasonably white and one can take full advantage of the code. Followed by horizontal decoding where the remaining error sequence may be more white.

5.5.3 BER Performance in a Multipath Channel

The performance of the three BTR schemes with the modified averaging slicer is shown in Figure 5.11. This multipath channel has a PDUR of 14 dB and is described in detail in [1]. This multipath channel is particularly bad from the Telidon viewpoint in that it has two strong close-in echoes. These echoes should cause little degradation of the TV picture, but as Figure 5.11 illustrates there is little hope of using it as a Telidon channel at these signal to noise ratios.

One notes from this figure that there is little difference between the different BTR schemes in this channel, although the correlator has about one dB better performance than the zero crossing based BTR, this makes little difference in the BER. In this case, any degradations due to the receiver hardware are dominated by the channel degradations. A simulation of an RF propagtion model in [2] produced distributions of VHF multipath channels that

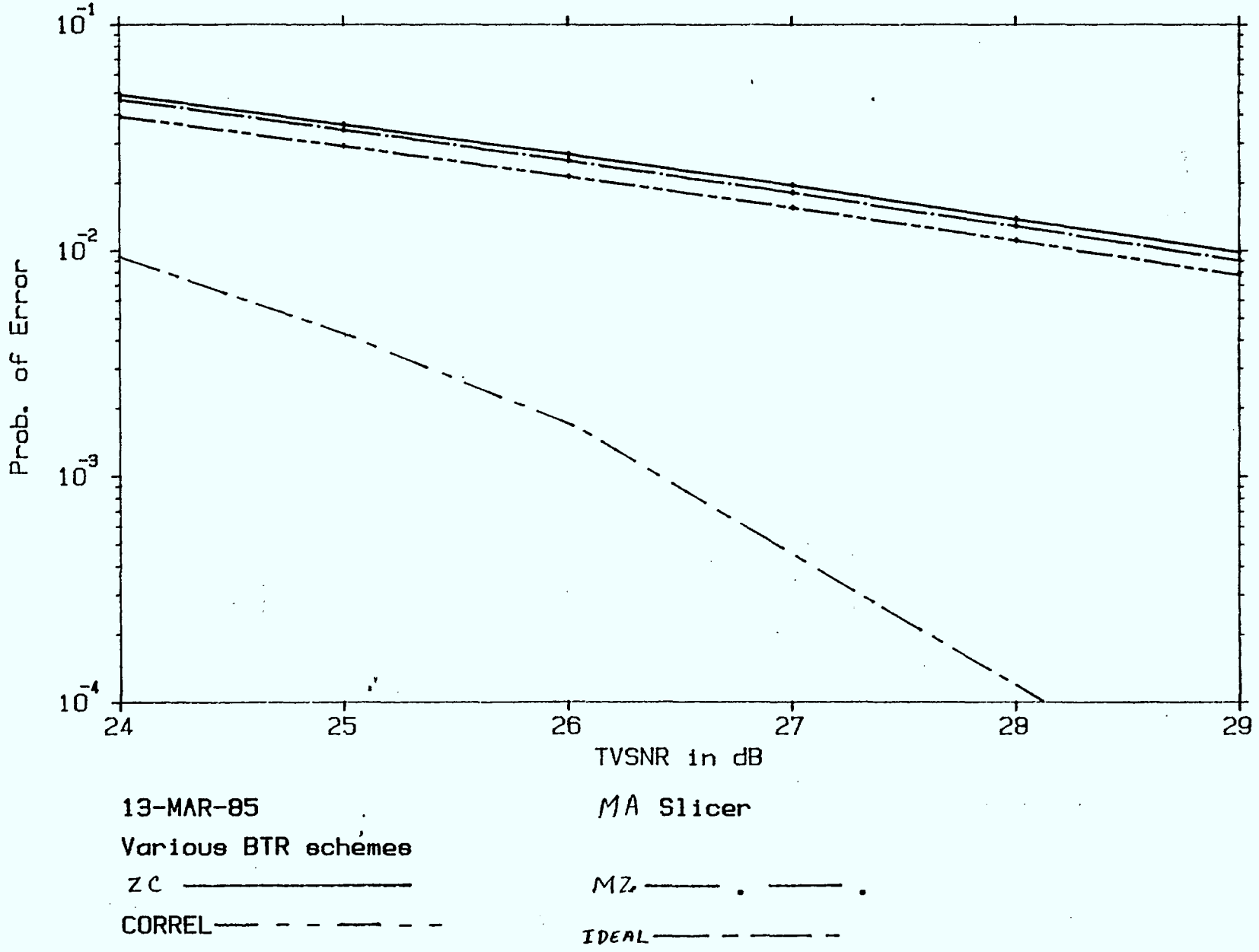


Figure 5.11 A comparison of the different BTR techniques in a multipath channel.

one might expect in urban and rural environments, and produced this channel as one example. While all multipath channels are not as bad as this one, it does point out that some channels will be of no use for Telidon unless some type of channel equalizer is in place.

5.6 Summary of Performance of Various BTR Schemes

Three bit timing recovery schemes have been described and compared in terms of performance. The first two schemes which base estimates of the proper clock phase on a single zero crossing, are similar to two hardware decoders that have been developed. The third method is closer to optimum and in practice would require more complex circuitry. However simulation results do indicate that it would provide a half dB improvement in gross BER performance over the best of the two other techniques. Simulation has also shown that the performance of the coding algorithm would be much enhanced with this technique. In terms of channel distortions, it appears that improved bit synchronization will not provide much relief for the problems of severe multipath channels.

6.0 VALIDATION OF THE DATA CHANNEL SIMULATION PROGRAM

In this section, validation of the data channel simulation program is discussed. To develop an appreciation for the difficulty of this task, we begin with an examination of an ideal validation scenario. Ideally, one would like to use real measurement data and perform a comparison of simulated and measured performance. To set up corresponding simulation runs one requires an accurate indication of the channel and noise characteristics for the measurement scenario, as well as an accurate simulation model of the teletext decoder used in the measurement program, and an exact match between the parameters of the simulation model (circuit time constants and filter bandwidths) and the hardware decoder used for the measurements. In addition to the above, one must also ensure that decoder hardware problems, such as sampling clock offsets and stability, do not colour the measurement results. If one is able to satisfy all the above prerequisites, one should obtain a reasonable match between simulated and measured performance. It is apparent from the above discussion that a precise validation would be difficult to achieve, given the amount and accuracy of the prerequisite data required.

Validation of the data channel simulation program has involved incorporating CRC field and lab measurement data into the simulation program, as well as DOC and EIA data on transmitter and receiver characteristics. The impulse responses extracted during the field measurement program were end-to-end inphase (real) impulse responses [11].

Ideally, one would like to extract complex baseband end-to-end impulse responses to characterize the vestigial sideband nature of the channel. Complex baseband impulse rsponses are essential for factoring out the multipath

channel characteristics from the measured end to end channel responses and examining the nature of the overall channel in the Nyquist slope region of the receiver. They are essential for isolating the responses of fundamental components of the teletext system so that the performance of advanced teletext receivers (characterizing the performance of carrier recovery systems and complex baseband equalizers) can be accurately predicted by simulation [12]. As documented in [11], it was clear that the necessary improvements to the CRC impulse response extraction system could not be incorporated in the field measurement system in time to provide complex baseband impulse responses for this contract. This is not a serious weakness for one-shot validation runs, as inphase impulse responses are sufficient for testing decoder performance under realistic channel conditions [11].

The validation approach is illustrated in Figure 6.1. Inphase impulse responses from the field measurement program are incorporated in the simulation program and simulated performance compared to measured performance. Simulation runs are also performed with transmitter and receiver responses that are more realistic than the nominal zero phase responses used in [1]. Since it is impossible to factor out the multipath characteristics from the measured end-to-end channel responses without complex baseband impulse responses, typical multipath channels generated using our VHF multipath propagation simulation [1], [2] will be incorporated.

In addition to this, simulation programs of other investigators, specifically P. Fockens and C. Eiler of Zenith Radio Corporation and M. Pittarelli of DOC, were examined. These simulation programs did not incorporate

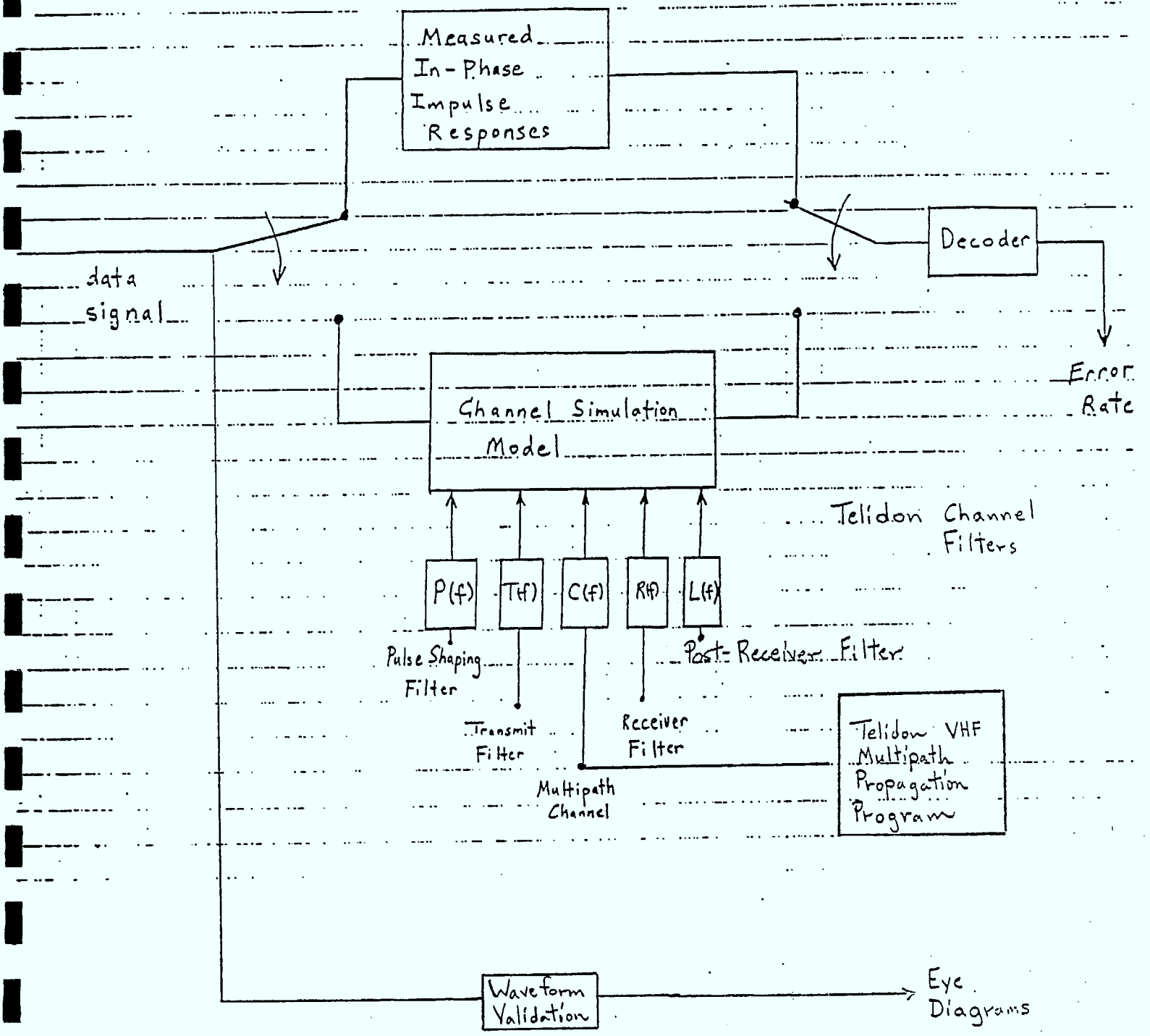


Figure 6.1 : Approach to Simulation Validation

specific teletext decoder implementations, but rather extracted properties of the eye diagram (i.e. eye height, eye width, and overshoots) as figures of merit. Although one cannot compare average bit error rate performance, one can set up corresponding simulation scenarios and essentially perform a "waveform validation" by comparing the resultant eye diagrams.

The two teletext decoders considered in the validation section are referred to by the acronyms PK-ZC and MA-MZ. These decoders employ suboptimal data detection procedures and are representative of commercially available teletext decoders. The operational aspects of these decoders will be briefly summarized here.

With both decoders, the two byte clock sync signal, an alternating one-zero bit pattern [6], is used to serially estimate the slicing level (dc - offset) and clock phase. The PK-ZC decoder first estimates the slicing level, using two peak detection circuits to follow the most and least positive signal excursions. These peak detection circuits have a short charging time and a long decay time. The peak detection outputs are smoothed and the slicing level taken midway between the two peak values. This slicing level technique is very sensitive to noise peaks, and has been substantially improved in the MA-MZ decoder. The MA-MZ decoder uses an averaging technique employing a 270 kHz second order low pass filter to determine the slicing level.

Probably the worst feature of the PK-ZC decoder was that after the slicing level was established, it used only a single zero crossing of the sliced data signal to choose between one of five possible clock phases. The interdependence of symbol synchronization and slicing level

estimation means that a slicing level error results in further degradation because it introduces an error in clock phase. This has been improved in the MA-MZ decoder, where slicing level and clock phase estimation are independent and a noise prefilter has been included in the clock synchronization circuit. Unfortunately, a single zero crossing is still used to establish the clock phase, although the jitter has been reduced somewhat by choosing one of ten possible clock phases instead of five.

In [1], it was stated that the performance of the PK-ZC detector was relatively poor, especially in multipath environments. Furthermore, performance is strongly dependent on model parameter settings such as the peak detector reference voltage settings, time constants and the activation point where the peak detectors start to operate. Because of this sensitivity, and the fact that it is difficult establishing an exact match between the parameters of the specific decoder used in the measurements and the software model, it was recommended that the more ideal averaging slicing level decoder be used to gather measurement data for channel validation. In addition to performing better, this decoder is easier to simulate and not nearly so sensitive to parameter settings of the model. It was planned that at least a limited number of field measurements would be conducted with this decoder to supply validation data for this contract. Characterization of the performance of the MA-MZ decoder in a back-to-back mode under controlled laboratory conditions (known levels of thermal noise injected into the system) was also planned.

Unfortunately, CRC had not received an operational version of the MA-MZ decoder by the completion date of this contract. The models of the averaging slicing level

decoder tested by CRC personnel under controlled laboratory conditions performed significantly worse than the peak decoder, which clearly indicates some type of fault or problem. As a consequence the field measurement validation runs will have to be performed with the PK-ZC simulation model. The performance level that would be expected for the prescribed channel and operating point (video signal to noise ratio) with the MA-MZ decoder is also provided. A partial validation involves verifying that one does, in fact, obtain better simulated performance with the averaging slicer than that measured with the peak detector decoder. The lack of measurement data for the averaging slicing teletext decoder is a major setback for the validation procedure.

6.1 Waveform Validation and Database Selection

Simulation programs developed by M. Pittarelli (DOC) and by Zenith Radio Corporation have been reviewed. As alluded to earlier, these simulation programs extract eye diagram parameters (i.e. eye height, eye width, and overshoots), which serve a figures of merit for system performance. The Telidon channel simulation program was developed to test the performance of teletext receiver implementations, incorporating slicing level estimation and bit timing recovery, with and without optimum linear equalization. The actual bit error rate performance of the system is evaluated by our simulation program. Furthermore, the resulting error sequences are stored in a compressed format for subsequent processing by a coding analysis program for assessing the performance of alternative coding strategies. Since our primary concern is actual system performance, routines for extracting eye diagram parameters have not

been developed. However, the Telidon channel simulation program does have the capability of generating eye diagram plots, and the eye height, eye width, and overshoot parameters can be measured from these plots.

The simulation program developed by M. Pittarelli [13] is essentially a baseband simulation program which does not consider the vestigial sideband nature of the teletext transmission channel. A block diagram of this simulation program is provided in Figure 6.2. The shaping filter represents the pulse shape, while the lowpass filter limits the data spectrum and is representative of the spectrum limiting imposed by the lowpass filter called for in the BS-14 specification [6]. Amplitude and group delay distortions can also be prescribed and their effect on the eye diagram quantified. In [13], a large number of results are presented for various combinations of raised cosine rolloff factors, lowpass filters, and amplitude and group delay distortion characteristics. We will not attempt to duplicate all these simulation results, but will concentrate on a few of the more interesting results. We will restrict our attention to the 100% raised cosine pulse shape (RC-100) and a lowpass filter (DOCLPF) with the amplitude response prescribed in Figure 6.3. Two group delay distortion characteristics, denoted by DOCGD1 and DOCGD2, illustrated in Figures 6.4 and 6.5, respectively, have been included in our database. In [13], the data pulse amplitudes are prescribed to be 0.25V (35 IRE). Eye diagrams for the simulation scenarios prescribed in Figure 6.6 have been obtained with our simulation program. The results from the Telidon channel simulation program are compared to the DOC simulation results in Figures 6.7, 6.8, and 6.9. Given that these eye diagrams are evaluated for different data sequences, these eye diagrams are in reasonable agreement, exhibiting the same trends. It is

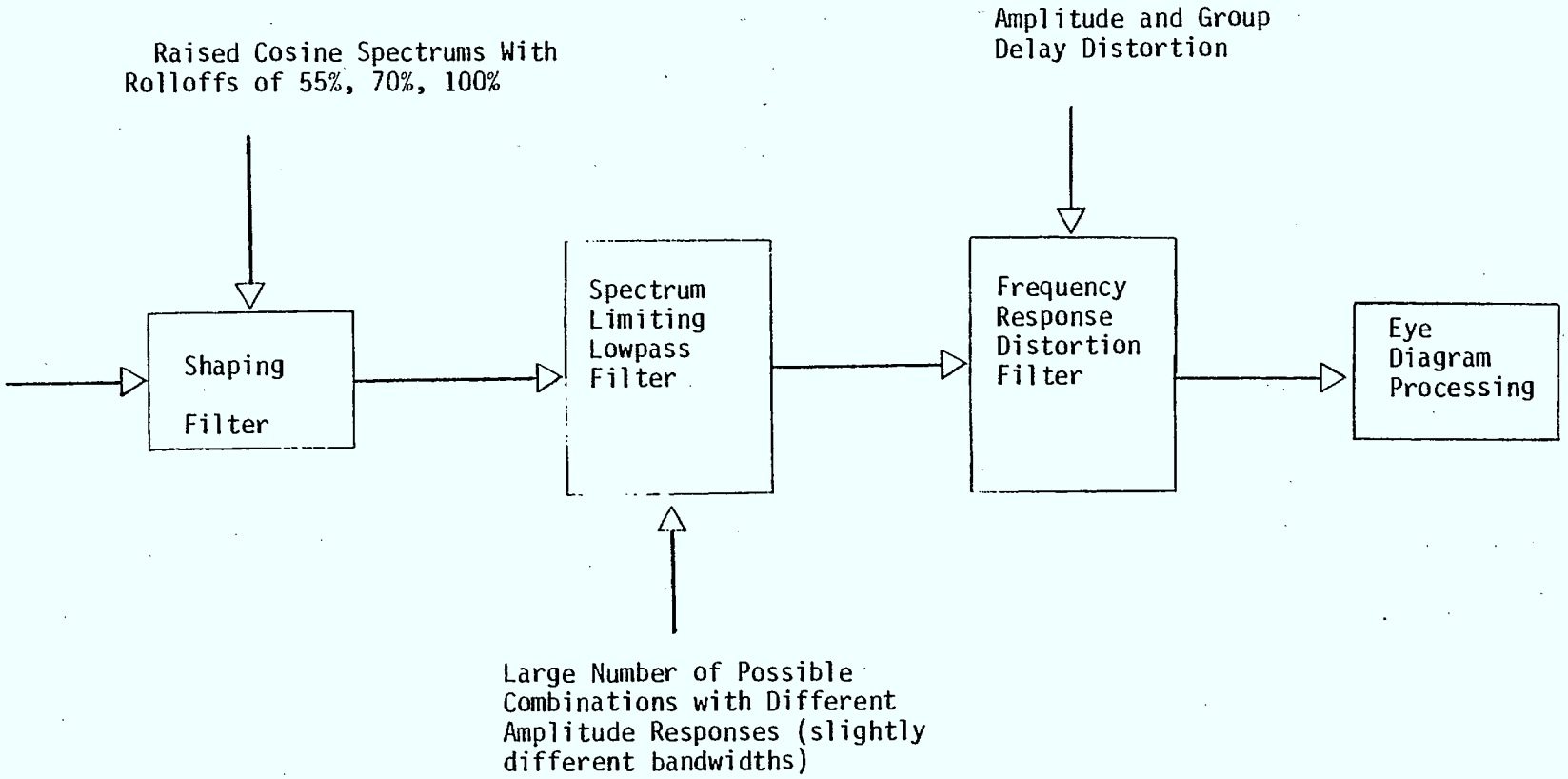


Figure 6.2 : Block Diagram of the DOC Simulation [13].

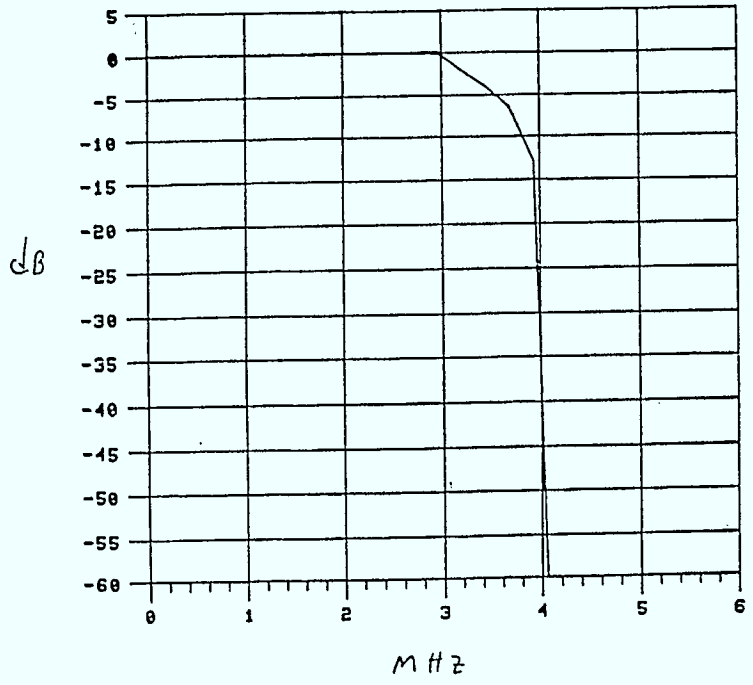


Figure 6.3 : Amplitude Response of Lowpass Filter DOCLPF.
(from [13])

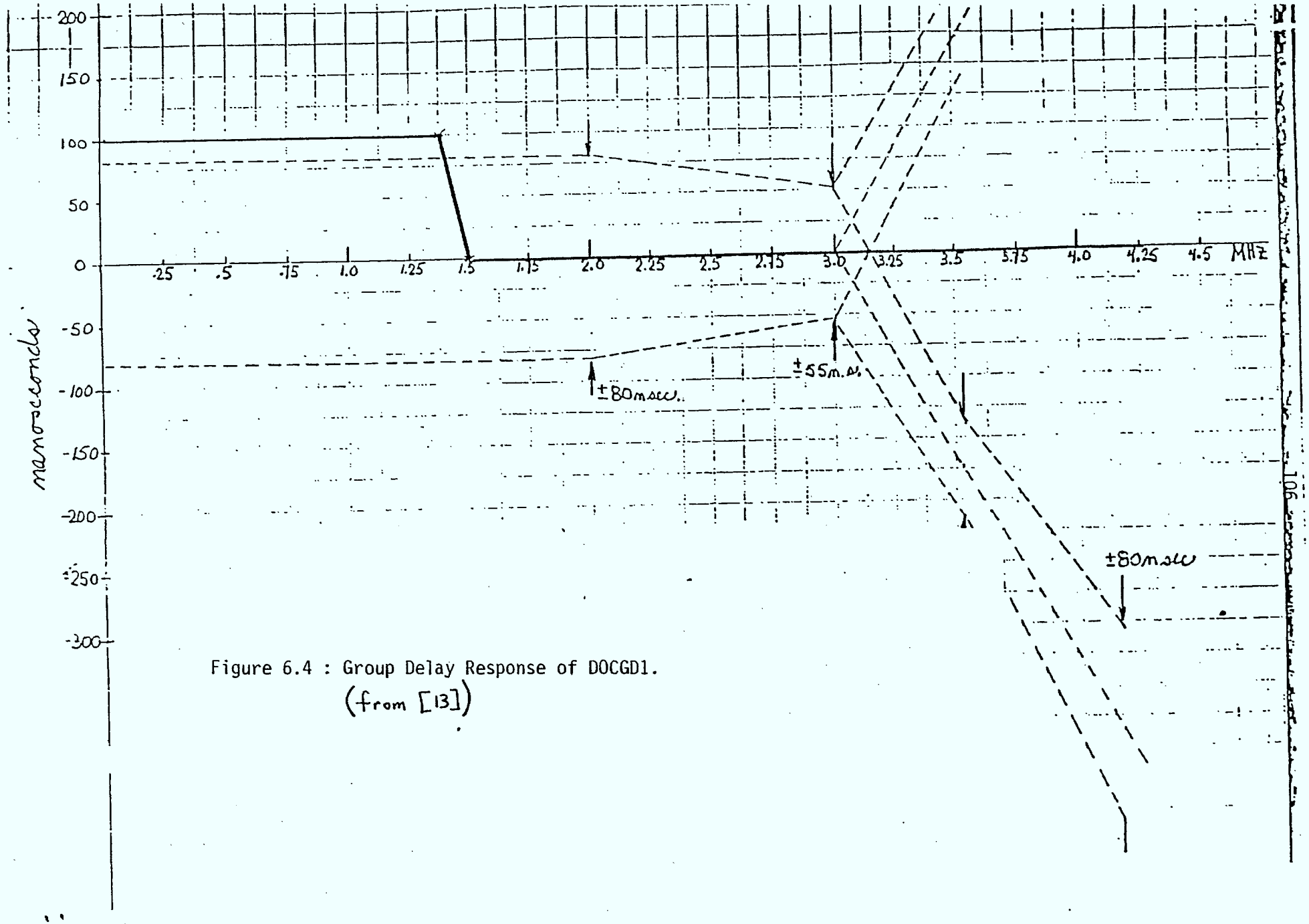


Figure 6.4 : Group Delay Response of DOCGD1.
(from [13])

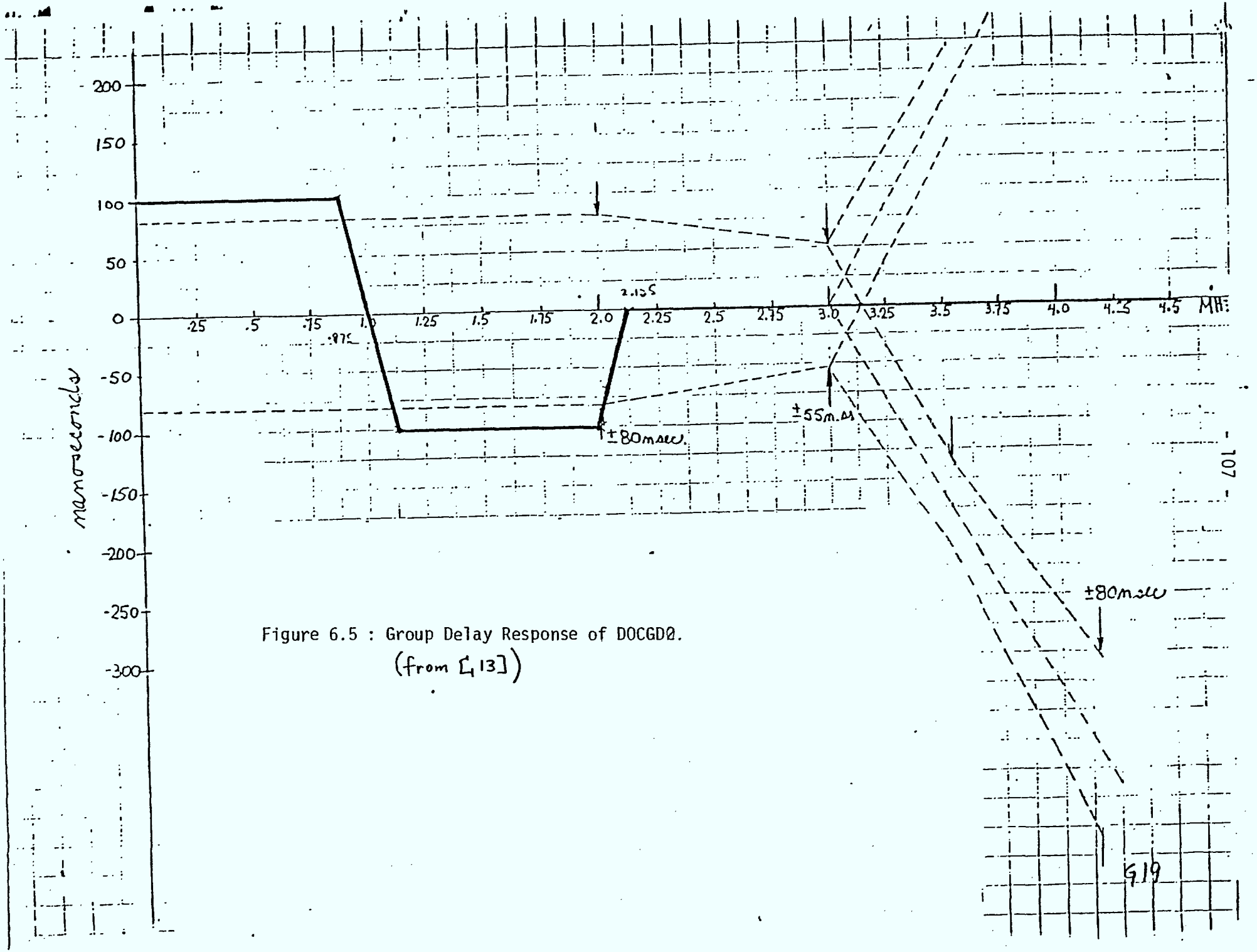
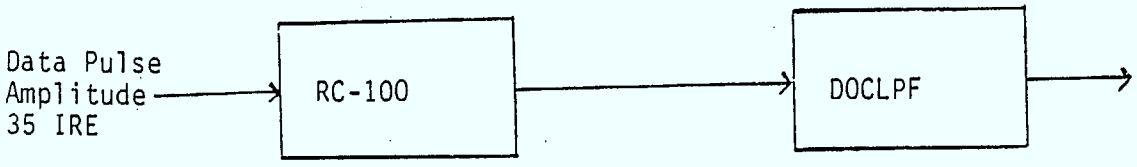
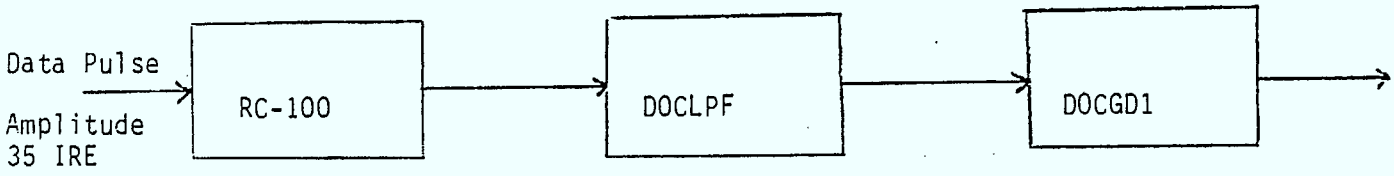


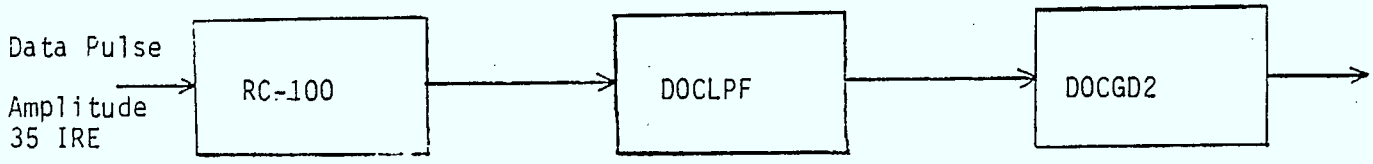
Figure 6.5 : Group Delay Response of DOCGD0.
(from [13])



(a) Simulation Scenario - A

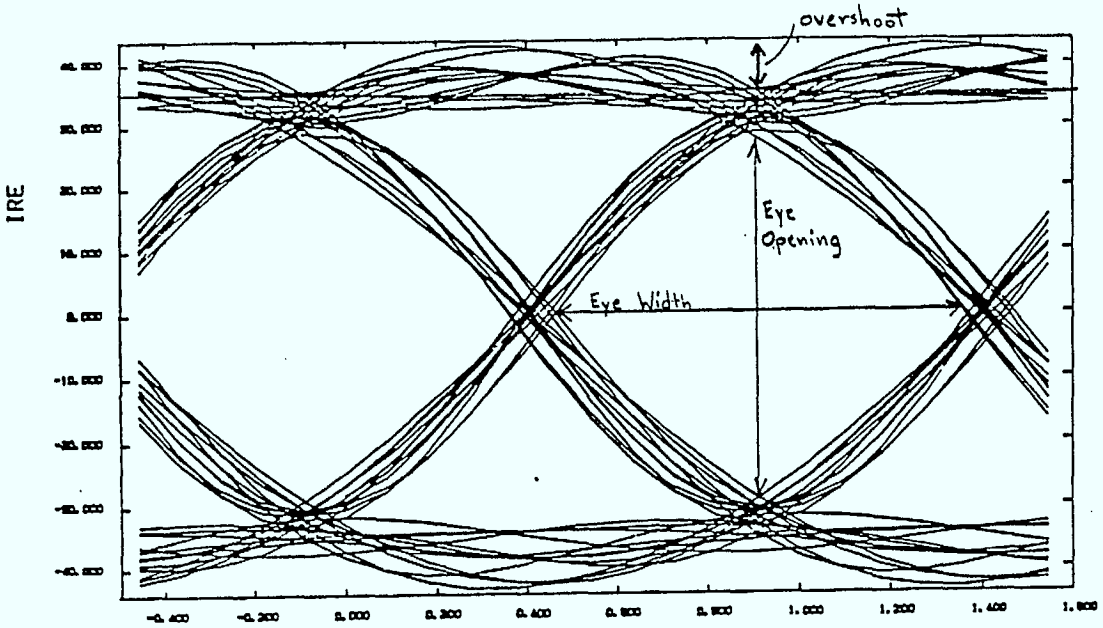


(b) Simulation Scenario - B



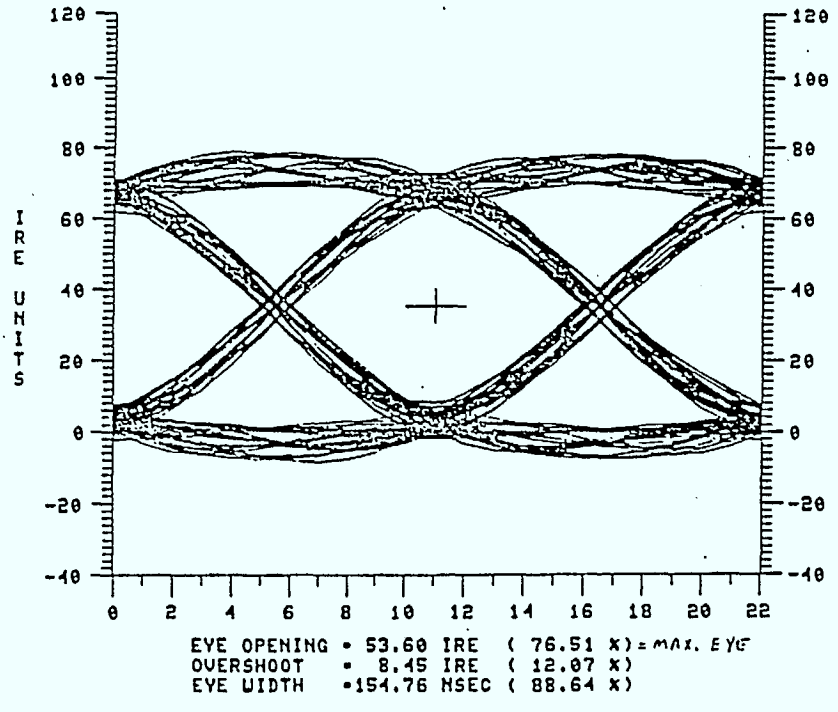
(c) Simulation Scenario - C

Figure 6.6 : Definition of Simulation Scenarios Considered



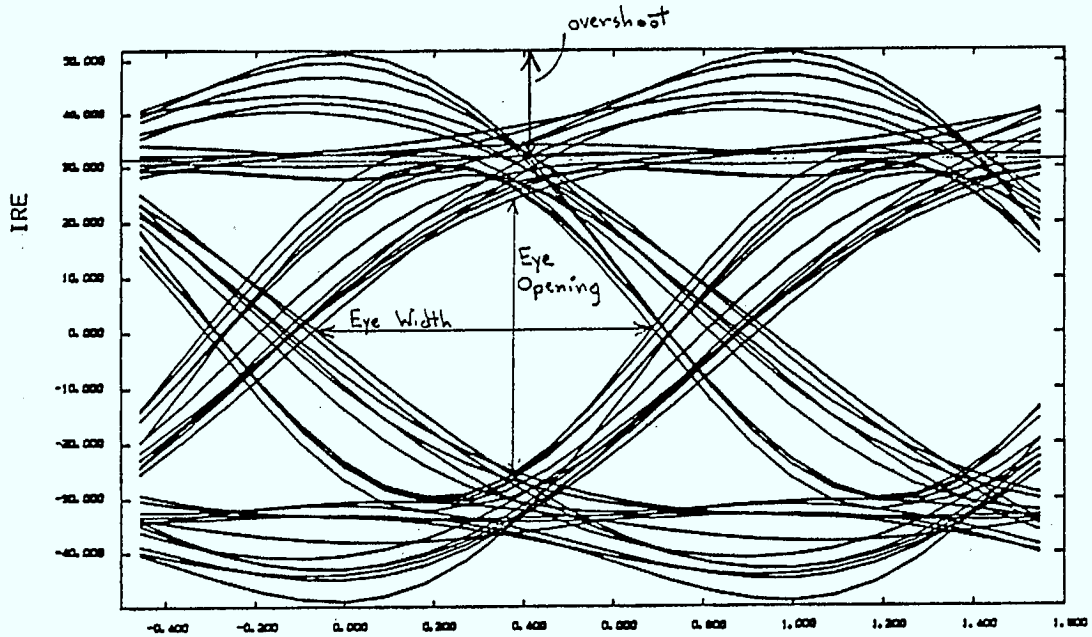
Max. Eye Opening = 55.5 IRE (79.3%)
Overshoot = 8.5 IRE (12.1%)
Eyewidth = 155 nsec (89%)

(a) MCS Simulation



(b) DOC Simulation

Figure 6.7 : Comparison of Eye Diagrams for Simulation Scenario A



Max. Eye Opening = 47 IRE (67.2%)
Overshoot = 16.3 IRE (23.3%)
Eye Width = 128 nsec (73.3%)

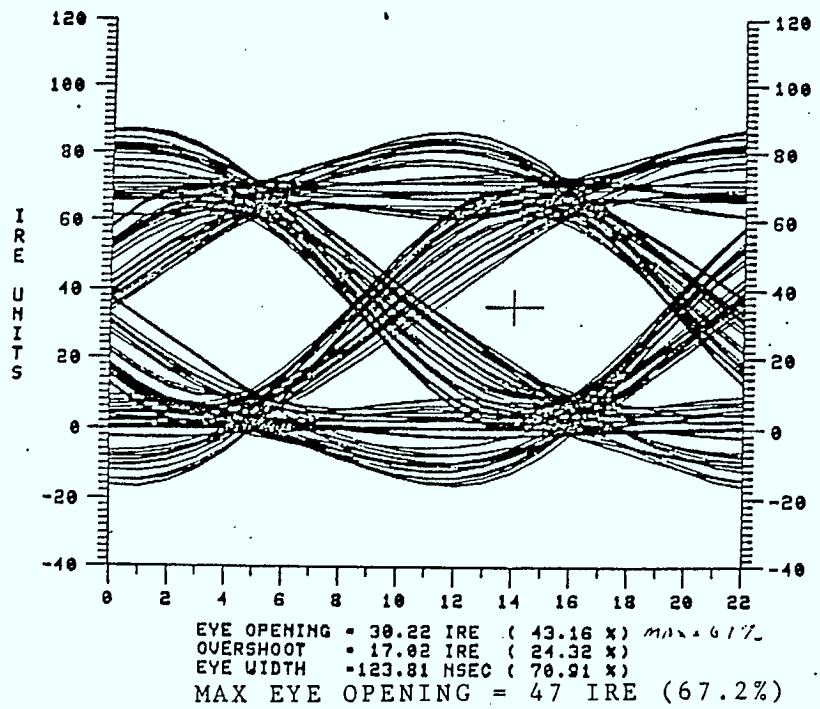
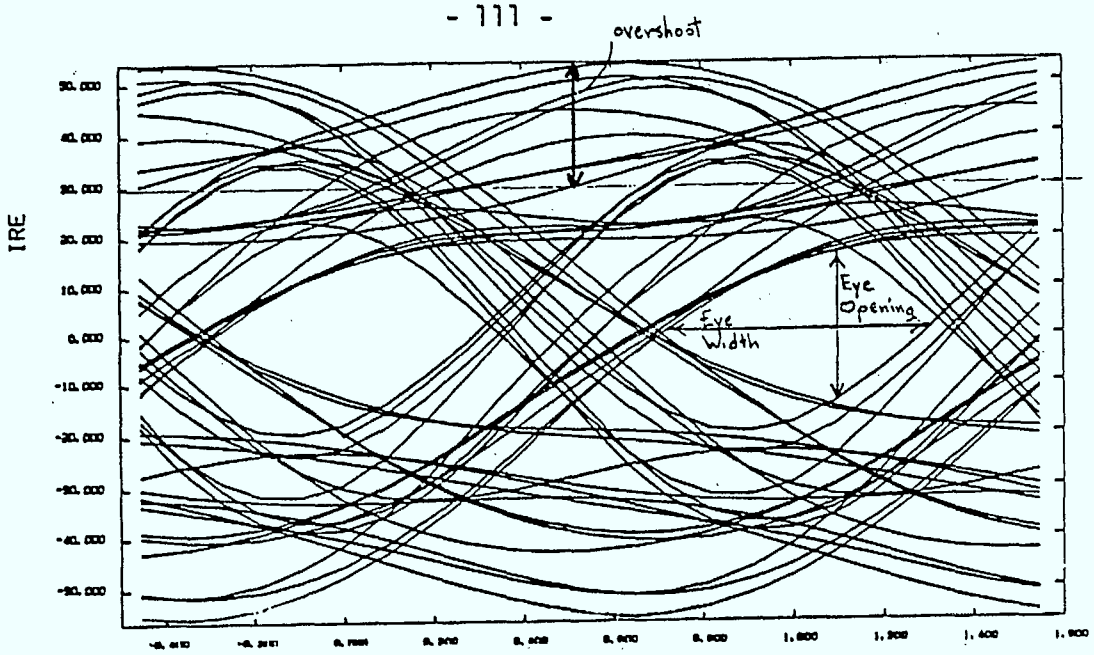
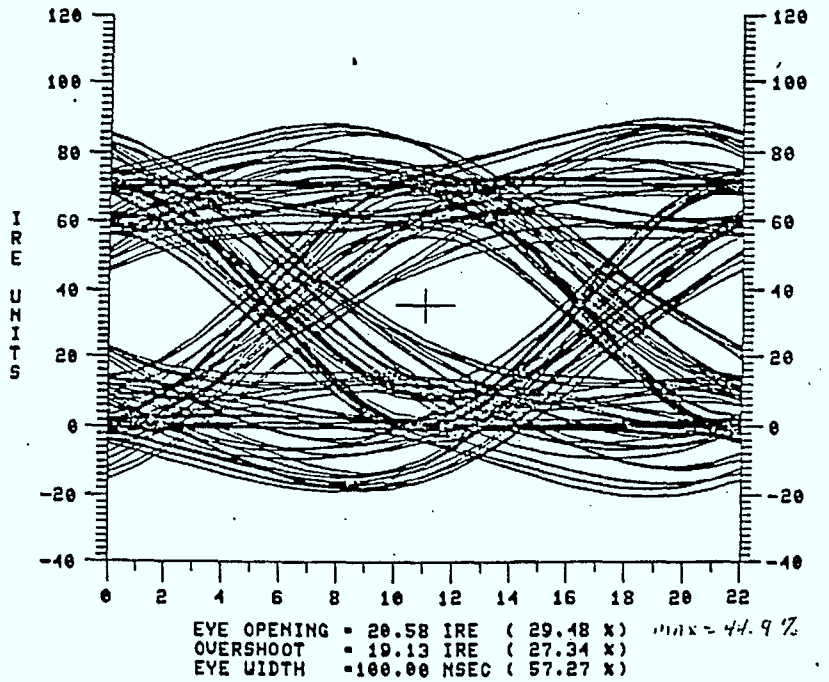


Figure 6.8 : Comparison of Eye Diagrams for Simulation Scenario B



Max. Eye Opening = 31 IRE (44.3%)
Overshoot = 19.4 IRE (27.7%)
Eye Width = 100 nsec (57.4%)

(a) MCS Simulation



MAX EYE OPENING = 31.4 IRE (44.9%)

(b) DOC Simulation

Figure 6.9 :Comparison of Eye Diagrams for Simulation Scenario C

clearly evident that more traces are incorporated in the DOC eye diagrams. Plotting execution time over slow data links dictated the choice of the number of traces considered in the Telidon channel simulation program eye diagram. The measured eye diagram parameters are compared to the values obtained with the DOC program in Table 6.1. Given the accuracy with which parameters can be measured from eye diagrams plots, these results are in reasonable agreement with the DOC results. It is apparent that group delay distortion can cause significant distortion in the eye diagram, causing the eye to become asymmetrical (skewed) with significant eye closure. This is especially the case for the group delay illustrated in Figure 6.5 (DOCGD2), which steps from roughly 100 nsec over the 0-1 MHz to -100 nsec over the range 1-2 MHz (see the corresponding eye diagram in Figure 6.9). These excursions are just beyond the ±80 nsec group delay masks for transmitters over this frequency range.

The simulation program developed by Zenith Radio Corporation [14] is more closely related to the Telidon channel simulation program [1]. That is, they have considered a vestigial sideband complex baseband system with individual component filter blocks similar to those defined in Figure 6.1 for the Telidon program. A number of transmitter and receiver responses are presented in [14]. The nominal transmitter and receiver responses presented in [14] are not appreciably different from the nominal zero phase transmitter and receiver responses used for system performance evaluation in [1]. Consequently, these responses will not be incorporated in our database. However, there is some very useful data presented in [14]. In particular, measured characteristics of a St. Louis, Missouri transmitter are provided. The tabulated amplitude and group delay responses for this transmitter are supplied in Table 6.2. The amplitude and group delay responses of

Simulation Scenario	Eye Diagram Parameter	Measured Values From Telidon Program Eye Diagrams	Computed Eye Diagram Parameters From DOC Simulation
A	Eye Opening Overshoot Eye Width	79.3% (55.5 IRE) 12.14% (8.5 IRE) 89% (155 nsec)	76.51% (53.6 IRE) 12.1% (8.45 IRE) 88.64% (154.76 nsec)
B	Eye Opening Overshoot Eye Width	67.2% (47 IRE) 23.3% (16.3 IRE) 73.3% (128 nsec)	67.2% (47 IRE) 24.32% (17.02 IRE) 70.91% (123.81)
C	Eye Opening Overshoot Eye Width	44.3% (31 IRE) 27.7% (19.4 IRE) 57.4% (100.2 nsec)	44.9% (31.4 IRE) 27.34% (19.13 IRE) 57.27% (100 nsec)

Table 6.1: Comparison of Measured Eye Diagram Parameters With Values Obtained With DOC Simulation.

FREQUENCY (MHZ)	AMPLITUDE (DB)	GROUP DELAY (NANOSEC)
-5.6,	0.,	-670.
-5.5,	0.,	-655.
-5.4,	0.,	-650.
-5.3,	0.,	-635.
-5.2,	0.,	-625.
-5.1,	0.,	-610.
-5.0,	0.,	-580.
-4.9,	0.,	-575.
-4.8,	0.,	-550.
-4.7,	0.,	-515.
-4.6,	0.,	-510.
-4.5,	0.,	-485.
-4.4,	0.,	-460.
-4.3,	0.03,	-415.
-4.2,	0.063,	-410.
-4.1,	0.156,	-380.
-4.0,	0.250,	-350.
-3.9,	0.344,	-325.
-3.8,	0.437,	-275.
-3.7,	0.344,	-275.
-3.6,	0.250,	-220.
-3.5,	0.344,	-200.
-3.4,	0.437,	-140.
-3.3,	0.344,	-80.
-3.2,	0.250,	-60.
-3.1,	0.156,	-40.
-3.0,	0.250,	-10.
-2.9,	0.344,	10.
-2.8,	0.437,	0.
-2.7,	0.531,	-20.
-2.6,	0.625,	-50.
-2.5,	0.718,	-65.
-2.4,	0.812,	-80.
-2.3,	0.906,	-60.
-2.2,	1.00,	-40.
-2.1,	1.00,	-40.
-2.0,	1.00,	-30.
-1.9,	1.00,	10.
-1.8,	1.00,	0.
-1.7,	1.00,	30.
-1.6,	1.00,	40.
-1.5,	1.00,	50.
-1.4,	1.00,	30.
-1.3,	0.833,	20.
-1.2,	0.666,	0.
-1.1,	0.5,	0.
-1.0,	0.5,	0.
-0.9,	0.5,	0.
-0.8,	0.5,	0.
-0.7,	0.5,	-10.
-0.6,	0.5,	-30.
-0.5,	0.666,	-10.
-0.4,	0.833,	40.
-0.3,	1.0,	40.
-0.2,	1.0,	20.
-0.1,	1.0,	10.
0.0,	1.0,	0.
0.1,	1.0,	10.
0.2,	1.0,	20.
0.3,	1.0,	40.
0.4,	0.833,	40.
0.5,	0.666,	-10.
0.6,	0.5,	-30.
0.7,	0.5,	-10.
0.8,	-0.8192,	0.
0.9,	-2.7335,	0.
1.0,	-5.1927,	0.
1.1,	-8.6359,	0.
1.2,	-14.4348,	0.

Table 6.2 : Measured Broadcast Television Transmitter, EIATX, as Presented in [14].

this transmitter are supplied in Figures 6.10 and 6.11, along with the amplitude and group delay masks prescribed in [15]. Note that these responses are well behaved, falling within the prescribed amplitude and group delay masks for the most part, and can be considered as representative of broadcast transmitters. Also provided in [14] is a television receiver model with the following characteristics:

(i) a tuner with negligible deviations of a flat response and zero phase,

(ii) a SAW IF filter with linear (zero) phase with a Nyquist slope amplitude response around the picture carrier, -6 dB response at the chroma subcarrier, and -13 dB response at the equivalent 4.2 MHz video frequency,

(iii) an inphase second detector,

(iv) with a sound trap with a 3 dB bandwidth of approximately 1 MHz and a minimum phase response, and

(v) finally, with a single amplitude equalizer lifting the video response at 4.5 MHz including its minimum phase response.

The amplitude and group delay responses of this receiver are supplied in Table 6.3 and plotted in Figures 6.12 and Figures 6.13, respectively. Although not directly applicable to commercial home receivers, the group delay tolerance masks for television demodulators are presented in Figure 6.14, for wideband and bandwidth-limited modes. Undoubtedly, the tolerances on commercially available home receivers will be worse (greater) than those applicable for

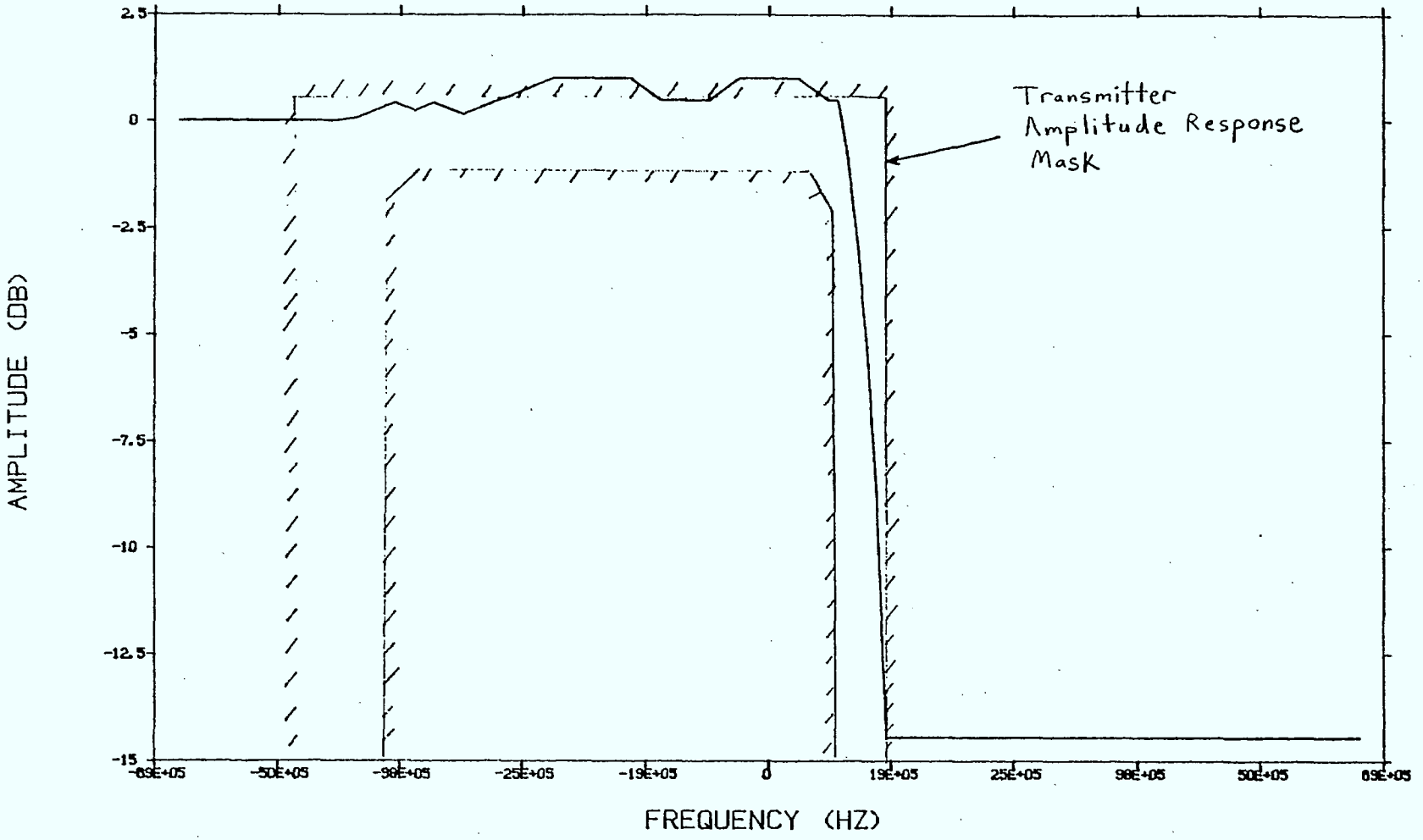


Figure 6.10 : Amplitude Response of EIATX

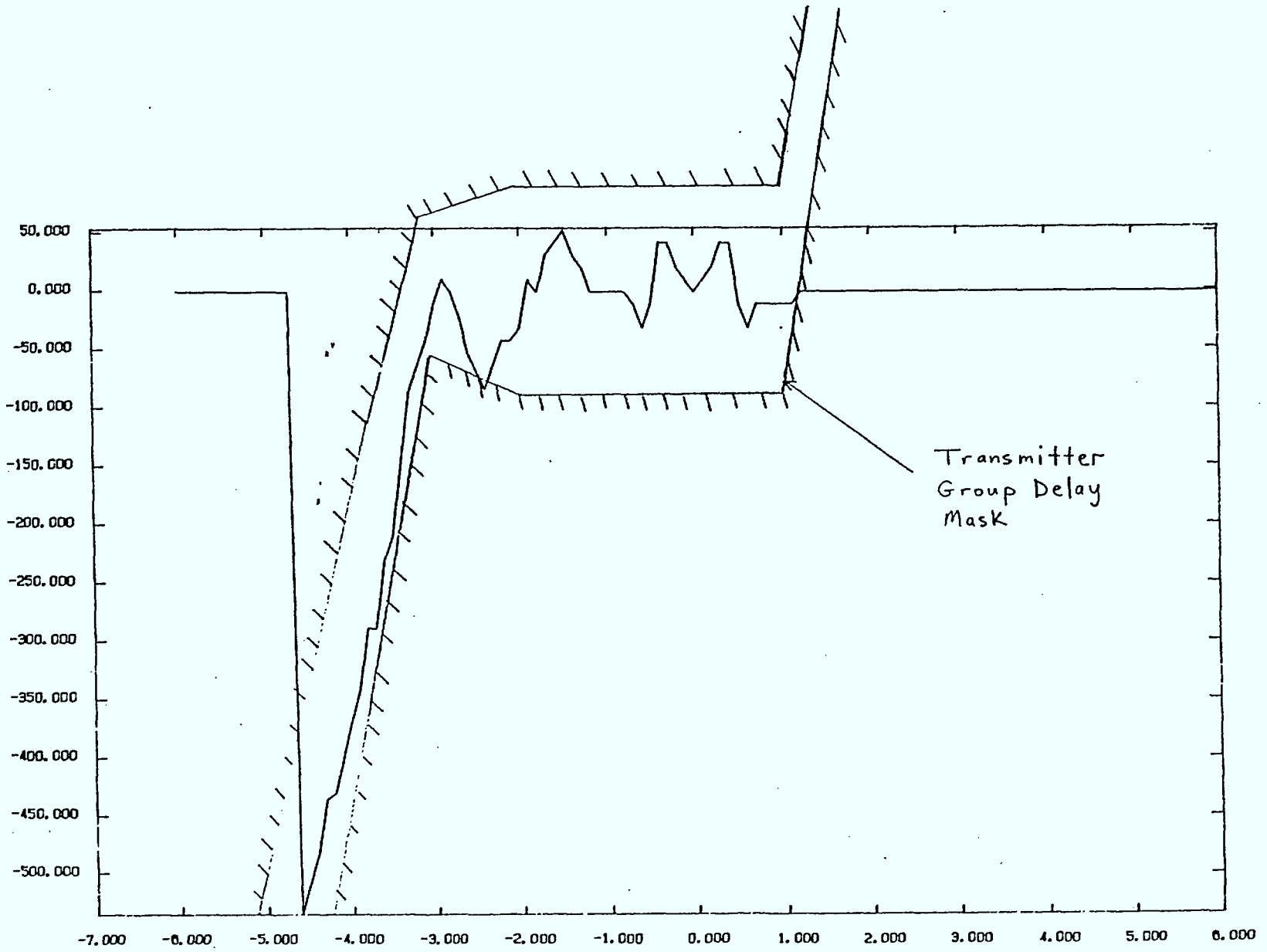


Figure 6.11 : Group Delay Response of EIATX

FREQUENCY (MHZ)	AMPLITUDE (DB)	GROUP DELAY (NANOSEC)
-5.6,	-45.37,	65.201
-5.5,	-43.90,	79.317
-5.25,	-39.08,	133.052
-5.,	-30.21,	227.471
-4.75,	-24.83,	368.501
-4.60,	-27.49,	446.732
-4.50,	-68.97,	472.454
-4.40,	-22.51,	465.926
-4.30,	-15.10,	428.882
-4.20,	-10.99,	372.372
-4.10,	-8.45,	309.488
-4.00,	-6.52,	249.668
-3.75,	-3.65,	136.081
-3.60,	-2.62,	92.258
-3.50,	-1.90,	70.738
-3.25,	-0.60,	35.563
-3.00,	0.02,	17.015
-2.75,	0.35,	7.395
-2.50,	0.50,	2.545
-2.25,	0.53,	0.233
-2.00,	0.53,	-0.741
-0.75,	0.06,	-0.317
-0.70,	0.03,	-0.278
-0.65,	-0.06,	-0.241
-0.60,	-0.18,	-0.207
-0.55,	-0.35,	-0.175
-0.50,	-0.58,	-0.145
-0.45,	-0.85,	-0.118
-0.40,	-1.17,	-0.093
-0.35,	-1.56,	-0.072
-0.30,	-1.99,	-0.053
-0.25,	-2.49,	-0.037
-0.20,	-3.06,	-0.024
-0.15,	-3.68,	-0.013
-0.10,	-4.38,	-0.006
-0.05,	-5.16,	-0.001
-0.00,	-6.02,	0.
0.05,	-6.98,	0.
0.10,	-8.05,	0.
0.15,	-9.23,	0.
0.20,	-10.56,	0.
0.25,	-12.04,	0.
0.30,	-13.72,	0.
0.35,	-15.63,	0.
0.40,	-17.83,	0.
0.45,	-20.40,	0.
0.50,	-23.48,	0.
0.55,	-27.28,	0.
0.60,	-32.23,	0.
0.65,	-39.23,	0.
0.70,	-51.25,	0.
0.75,	-60.00,	0.
1.00,	-70.00,	0.
1.20,	-80.00,	0.

Table 6.3 :Television Receiver Response, EIARX, as Presented in [14].

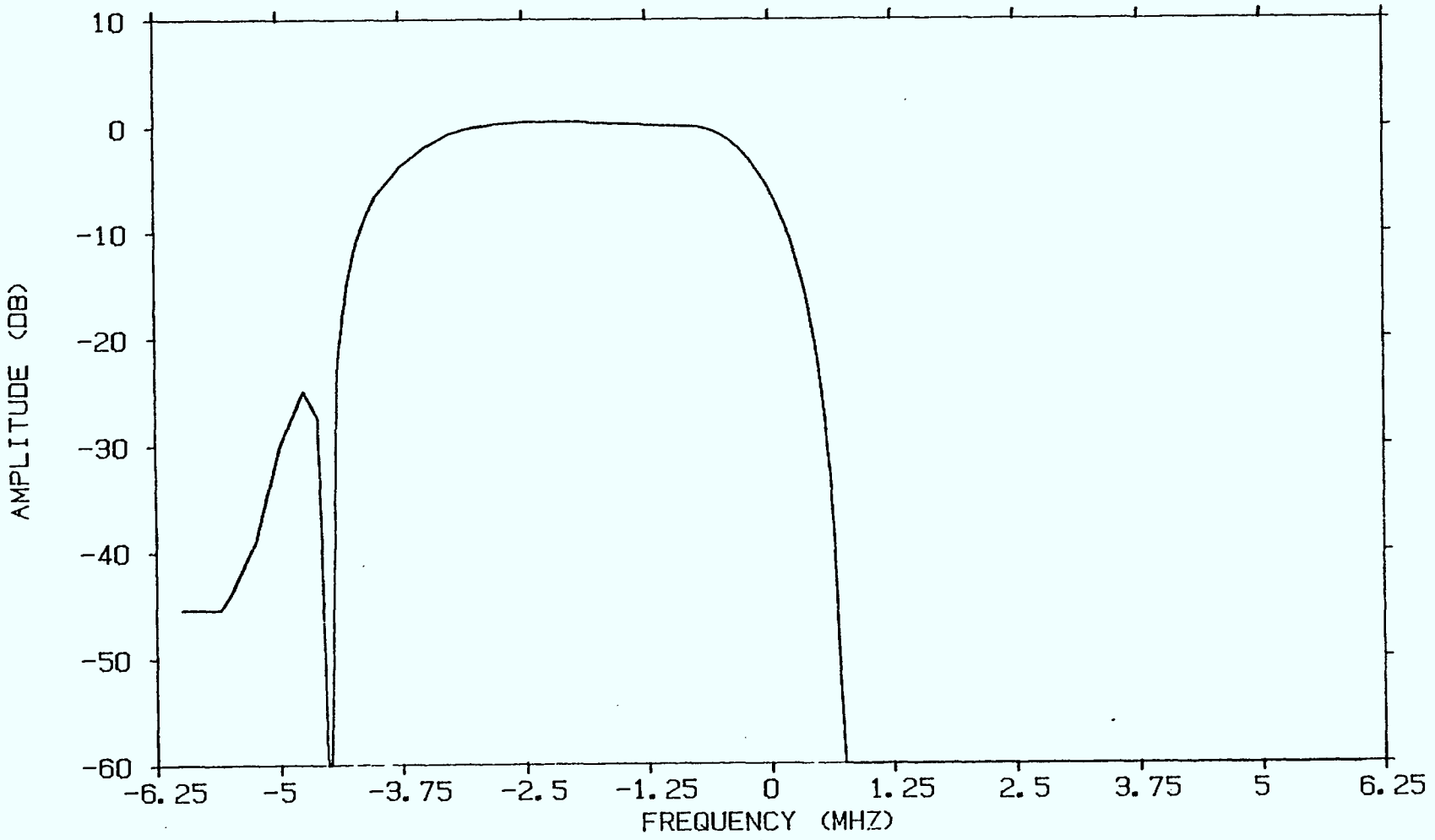


Figure 6.12 : Amplitude Response of EIARX

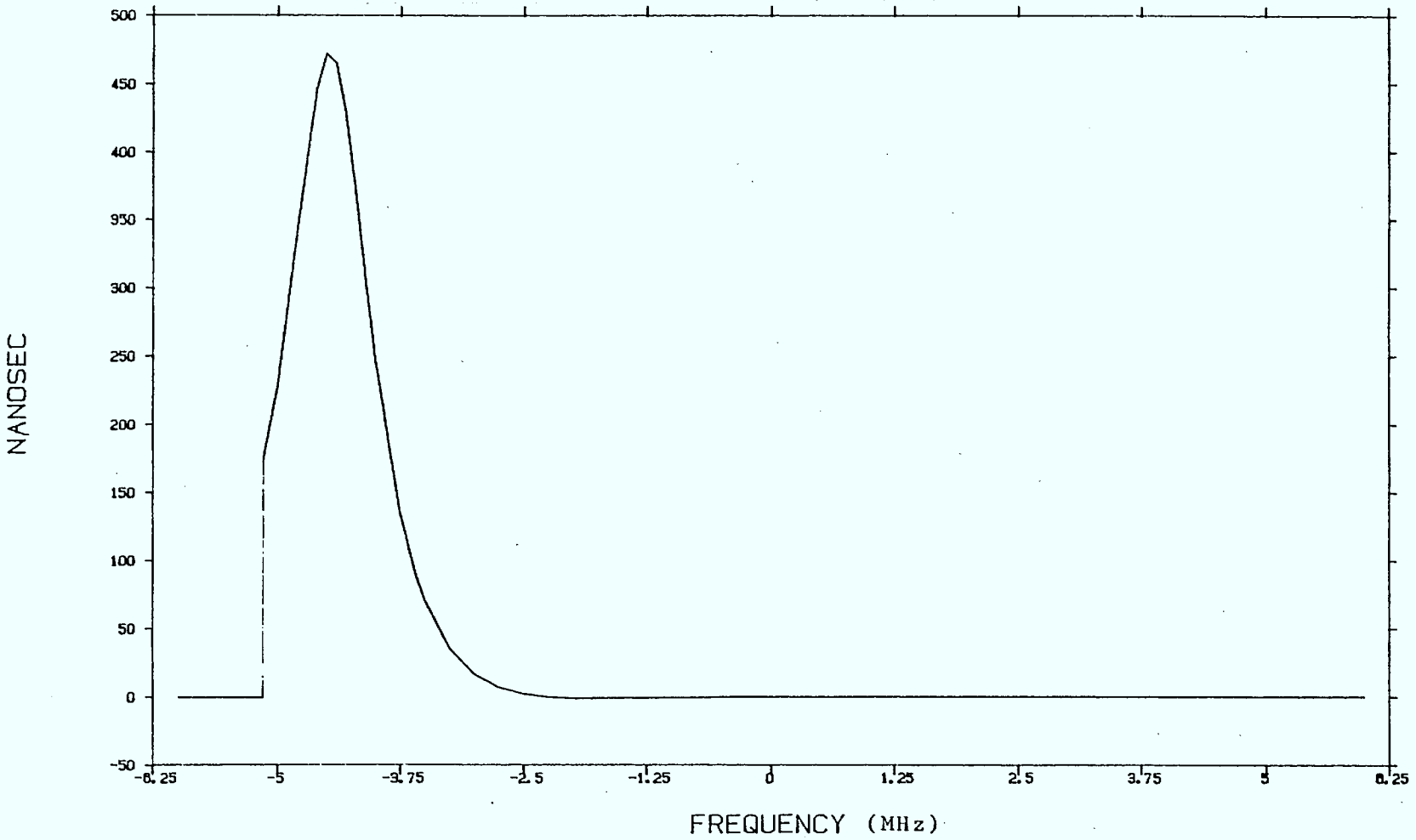


Figure 6.13 : Group Delay Response of EIARX

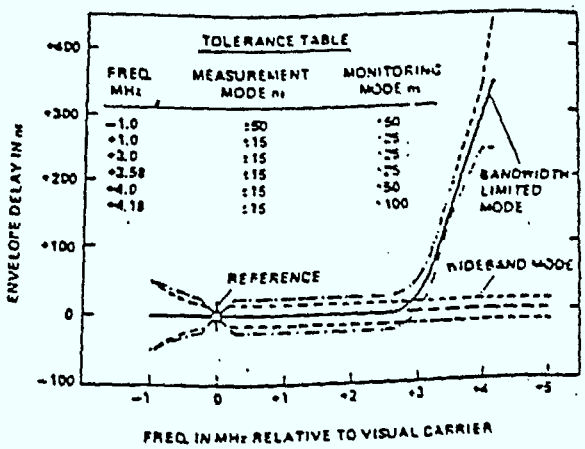


Figure 6.14 : Group Delay versus Sideband Frequency Response for Station Demodulators. (from [16]).

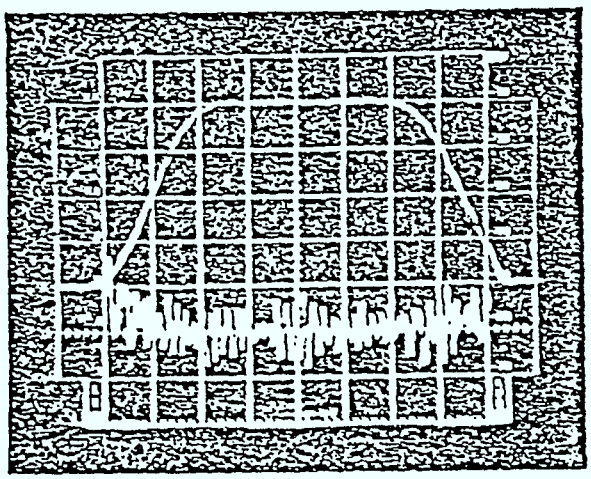


Figure 6.15 : Saw Filter Group Delay on Bottom Trace for a Tektronix Demodulator (from [16]).

station demodulators. This raises serious doubts about the accuracy of the essentially zero group delay characteristic assumed for the EIARX receiver response. As discussed in [16], the attenuation in the receiver Nyquist slope can result in departures from linear phase, even for surface acoustic wave (SAW) filters. A general characteristic of surface acoustic wave filters is the presence of echoes, in particular triple transit echoes. This gives rise to relatively fast ripple in the group delay response. This characteristic is illustrated in Figure 6.15, which shows the group delay response of Tektronix station demodulator. Note that ±30 nsec ripples are not uncommon, and that they can be as bad as ±50 nsec.

A more realistic candidate receiver response for the database can be formed by cascading the EIARX response with a typical SAW filter group delay response. The amplitude response of EIARX is quite realistic. This new candidate, POSTX, was generated by adding the group delay function

$$G(f) = 30 \sin(\frac{2\pi f}{0.2}) \quad \text{nsec},$$

where f is the frequency in MHz, to the EIARX response. The resulting group delay characteristic is illustrated in Figure 6.16.

In the Zenith simulation, responses are supplied for the pulse shape, transmit and receiver filters, and the resulting eye diagram parameters extracted. A minor change was made to the bit rate, r_o = 5.688889 MHz, to facilitate obtaining 18 samples per bit period. This is only a change of 0.67% and should have little consequence on the results. In generating our eye diagrams, only 11 samples per bit period are used. Because average bit error rate performance is established with our simulation, a large

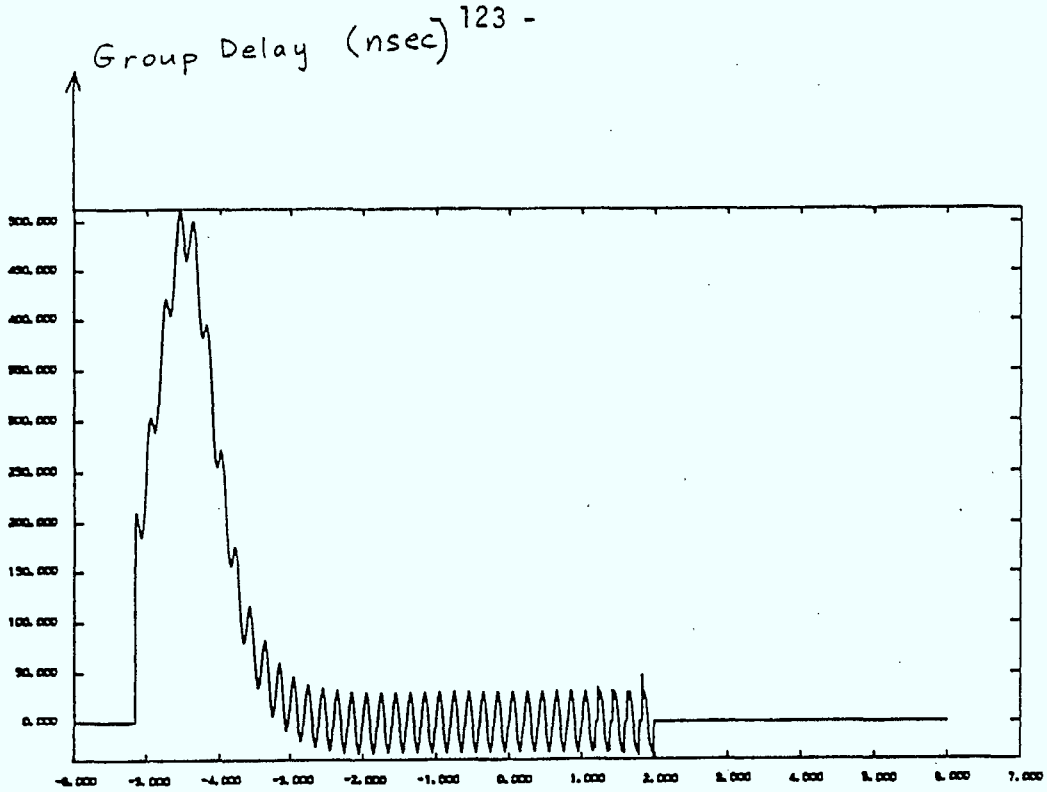


Figure 6.16 : Group Delay Response of Receiver Filter POSRX

number of bursts must be simulated, and for computational efficiency the number of samples per bit must be kept as small as possible.

The pulse shapes considered in the Zenith simulation are raised cosines with rolloff rates of 55%, 70% and 100%. The pulse amplitude was selected to be 100 IRE units. A large number of scenarios were considered in [14], and will not be repeated here. Attention will be restricted to the scenario illustrated in Figure 6.17. The characteristics of the video bandlimiting filter EIAVBL are tabulated in Table 6.4, and this filter is essentially a 4.35 MHz (3 dB BW) lowpass filter. The eye diagrams obtained with the Zenith simulation are provided in Figure 6.18. The eye opening, eye width, and overshoot parameters are difficult to read from this figure, which serves mainly as an indicator of the shape of the resulting eye diagrams. The eye diagrams for the RC-100, RC-70, and RC-55 pulse shapes obtained with the Telidon channel simulation program are presented in Figures 6.19, 6.20, and 6.21 respectively. The basic shapes of the eye diagrams agree well with those presented in Figure 6.18 (note there are clear differences in the perspective between these eye diagram plots). It is also clear that some type of normalization must have been applied to constrain the output pulse amplitudes to 100 IRE in the Zenith simulation. Note that the overall system gain at the picture carrier is 1 dB with the prescribed filters in Figures 6.17, which gives a peak to peak output data amplitude of 112.2 IRE. This phenomena has been observed on the eye diagrams produced by the Telidon channel simulation program. The extracted eye parameters from Figure 6.19, 6.20, and 6.21 are summarized in Table 6.5, and compared to the values presented in Table I of reference [14]. Concerning overshoots, we have considered the worst case overshoots (which are actually undershoots)

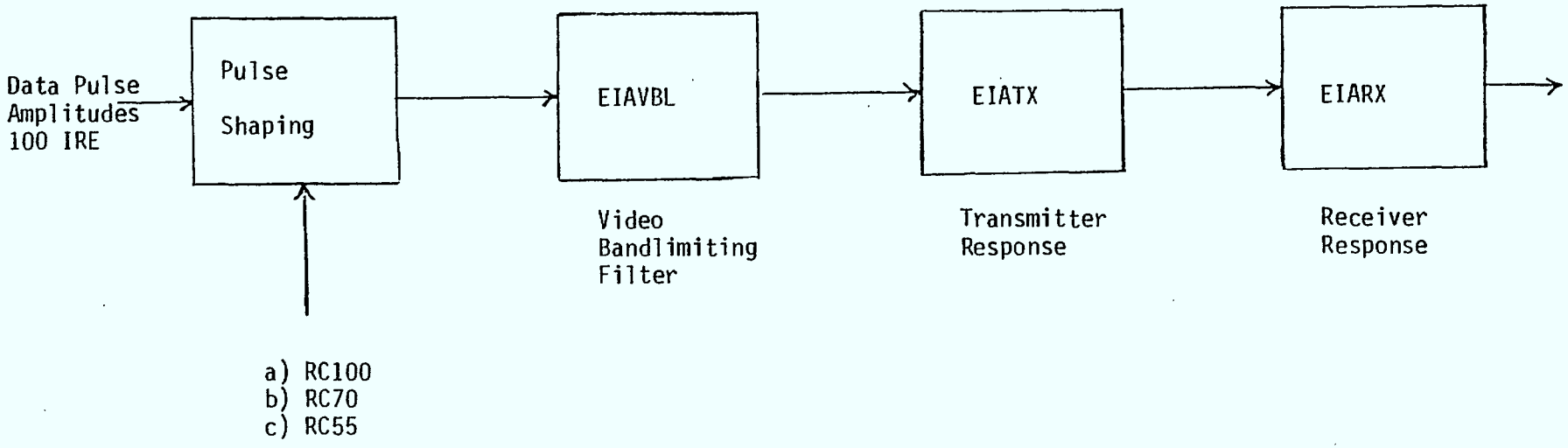


Figure 6.17 : Simulation Scenario from [14] that Will Be Used for Validation.

FREQUENCY (MHZ)	AMPLITUDE (DB)	GROUP DELAY (NANOSEC)
0.,	0.,	0.
4.15,	0.,	0.
4.20,	-0.285,	0.
4.25,	-1.041,	0.
4.30,	-1.868,	0.
4.35,	-2.784,	0.
4.40,	-3.806,	0.
4.45,	-4.967,	0.
4.50,	-6.305,	0.
4.55,	-7.890,	0.
4.60,	-9.827,	0.
4.65,	-12.327,	0.
4.70,	-15.847,	0.
4.75,	-21.873,	0.
4.80,	-99.,	0.
5.60.	-99.,	0.

Table 6.4 : EIA Video Bandlimiting Filter EIAVBL as Presented in [14].

RC-55

RC-70

RC-100

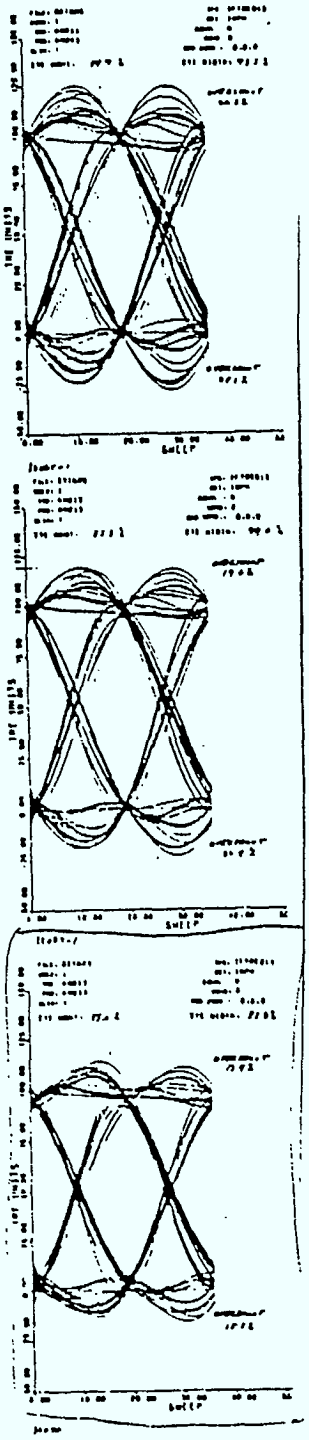
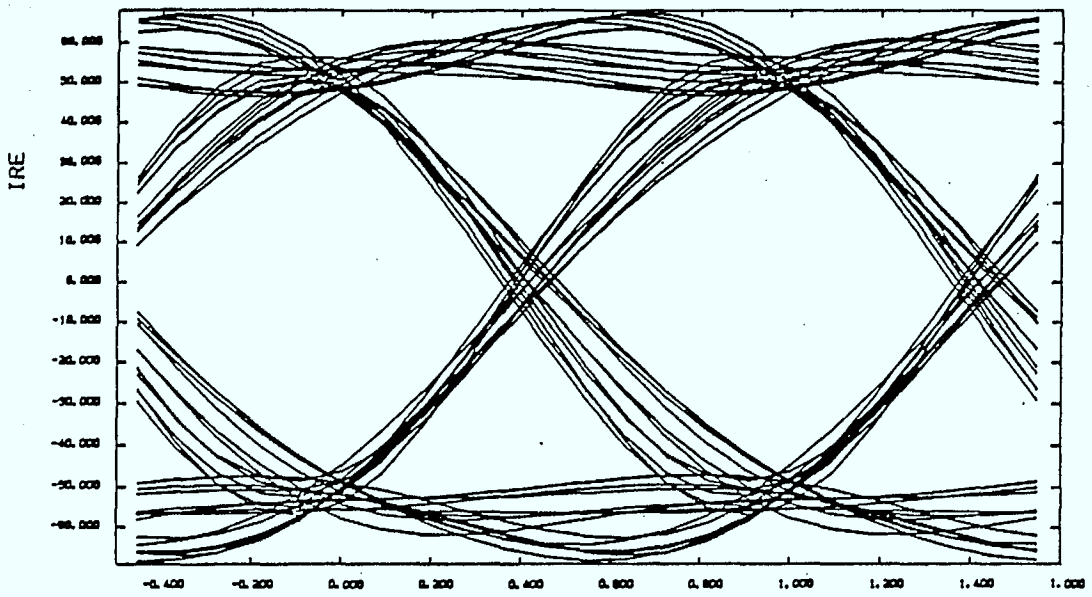
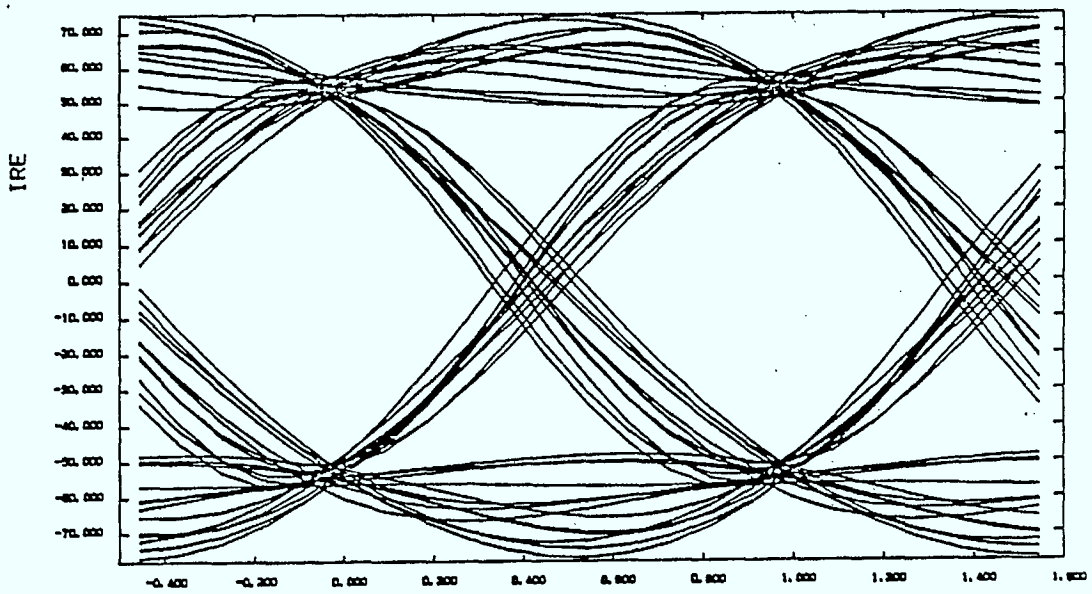


Figure 6.18 : Eye Diagrams Obtained for the Scenarios of Figure 6.17 with the Zenith Simulation program [14].



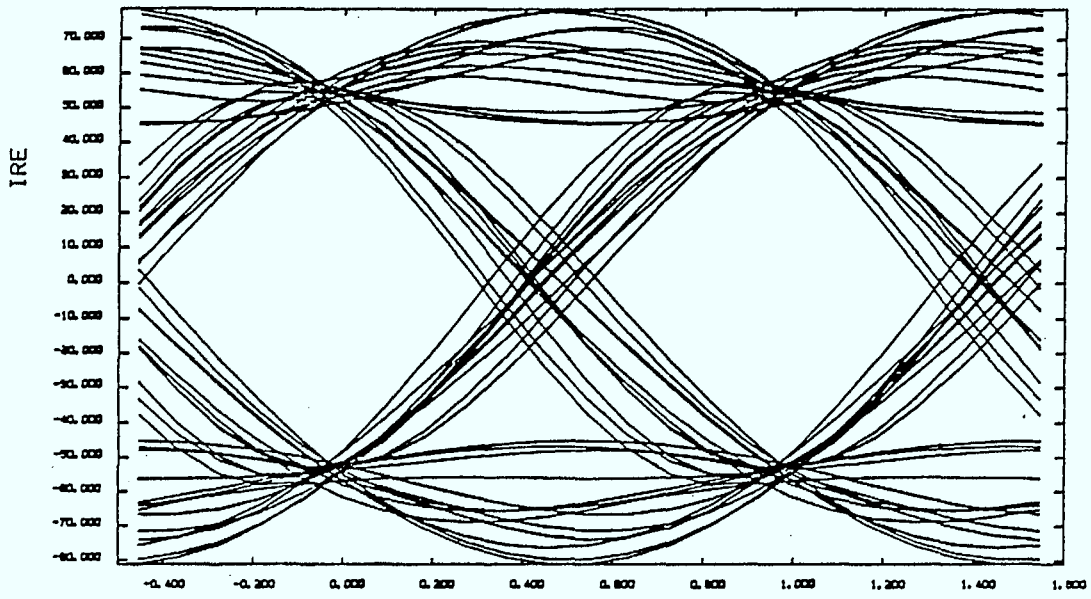
Output (peak-to-peak) Data Level = 112.2 IRE
Max. Eye Opening = 92 IRE (82%)
Overshoot = 16.1 IRE (14.3%)
Eye Width = 86%

Figure 6.19 : Eye Diagram for the Simulation Scenario Illustrated in Figure 6.17 with RC-100 Pulse Shaping



Output (peak-to-peak) Data Level = 112.2 IRE
Max. Eye Opening = 100 IRE (89%)
Overshoot = 24.4 IRE (21.7%)
Eye Width = 78%

Figure 6.20 : Eye Diagram for the Simulation Scenario Illustrated in Figure 6.17 with RC-70 Pulse Shaping



Output (peak-to-peak) Data Level = 112.2 IRE
Max. Eye Opening = 101.5 IRE (90.5%)
Overshoot = 28.4 IRE (25.3%)
Eye Width = 74%

Figure 6.21 : Eye Diagram for the Simulation Scenario Illustrated in Figure 6.17 with RC-55 Pulse Shaping

Pulse Shape	Eye Diagram Parameter	Measured Values From Simulated Eye Diagram	Computed Eye Data From Zenith Simulation
RC100	Eye Opening	82%	83%
	Overshoots	14.3%	15.3%
	Eye Width	86%	88.3%
RC70	Eye Opening	89%	88.9%
	Overshoots	21.7%	22.9%
	Eye Width	78%	79.4%
RC55	Eye Opening	90.5%	89.7%
	Overshoots	25.3%	28.1%
	Eye Width	74%	73.3%

Table 6.5: Comparison of measured eye parameters to computed eye data from [14] for the simulation scenarios of Figure 6.16.

at the low signal level (black). The parameters are in as good agreement as can be expected given the accuracy with which parameters can be measured from eye diagrams.

There is also the facility of simulating a multipath channel in the Zenith simulation. This multipath channel simulation employs only a single echo with randomly selected echo parameters. The echo amplitude has a Gaussian distribution with zero mean and variance $\sigma^2 = 0.09$, and an exponential echo delay distribution with a mean delay of 2 bit periods, and a uniform echo phase distribution. As a realistic multipath channel model, this approach has several weaknesses. One of the most glaring weaknesses is the lack of correlation between echo magnitude and echo delay. From physical reasoning, there should be a falloff of echo magnitude with echo delay. The VHF multipath simulation program presented in [2], [17] is a much more realistic multipath channel generator. Being a physical propagation model, this program provides insight into the behaviour of VHF multipath propagation as a function of frequency and environment. The severity of multipath components as a function of delay for various frequencies and reflector characteristics are presented and discussed in [2].

Furthermore, the multipath channel generator provides an indication of the video quality of the channel by providing a perceived desired to undesired signal strength ratio, PDUR. This PDUR ratio can be related to the subjective picture quality associated with the multipath channel [17]. Consequently, multipath channels generated by the Telidon propagation simulation program will be used in establishing system performance for multipath environments in the channel simulation program.

Before concluding this section, another transmitter characteristic for the database, denoted by POSTX, is defined. This response has been generated by simply drawing characteristic curves which fit within the transmitter amplitude and group delay masks presented in [15]. The group delay response of POSTX is presented in Figure 6.22. The group delay tolerances in the vestigial sideband region of the response are not defined in [15], so a realistic characteristic has been assumed in Figure 6.22. The amplitude response of POSTX is supplied in Figure 6.23 along with the transmitter amplitude response mask [15]. The amplitude and group delay ripple of the prescribed hypothetical transmitter, POSTX, has a relatively high rate.

To summarize the accomplishments of this section, it has been demonstrated that the Telidon channel simulation program gives similar results to those developed by other investigators, namely M. Pittarelli of DOC and P. Fockens and C. Eiler of Zenith Radio Corporation. The relatively minor differences observed can be attributed to the accuracy with which eye diagram parameters can be measured from eye diagram plots, and the fact that different data sequences, and different numbers of samples per bit and a different number of traces are used in generating eye diagrams. A database of potential transmit and receive filter characteristics have been defined. These are summarized below:

(1) NOMINALTX
(2) NOMINALRX
(3) EIATX
(4) EIARX
(5) POSTX
(6) POSRX

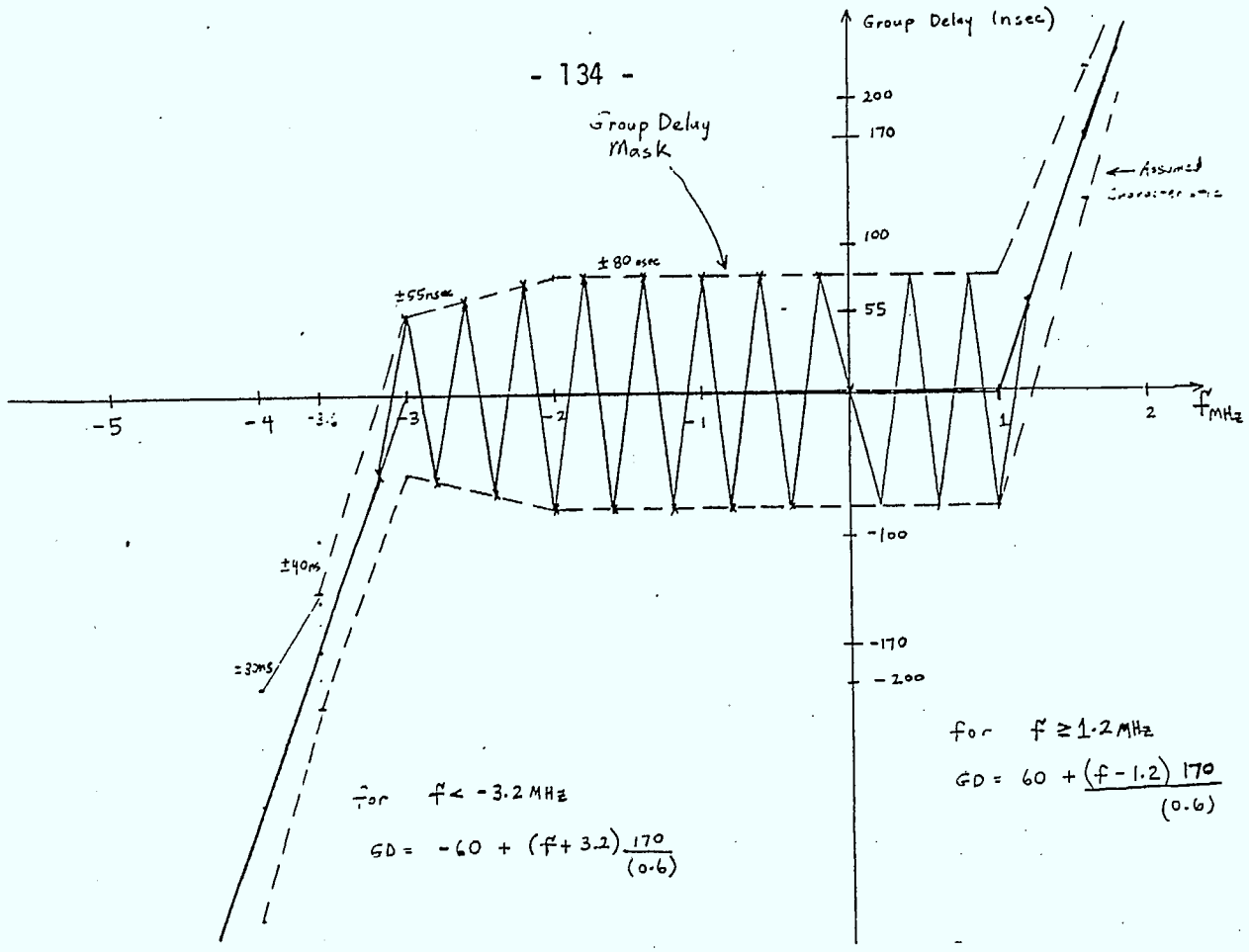


Figure 6.22 : Group Delay Response for POSTX Filter

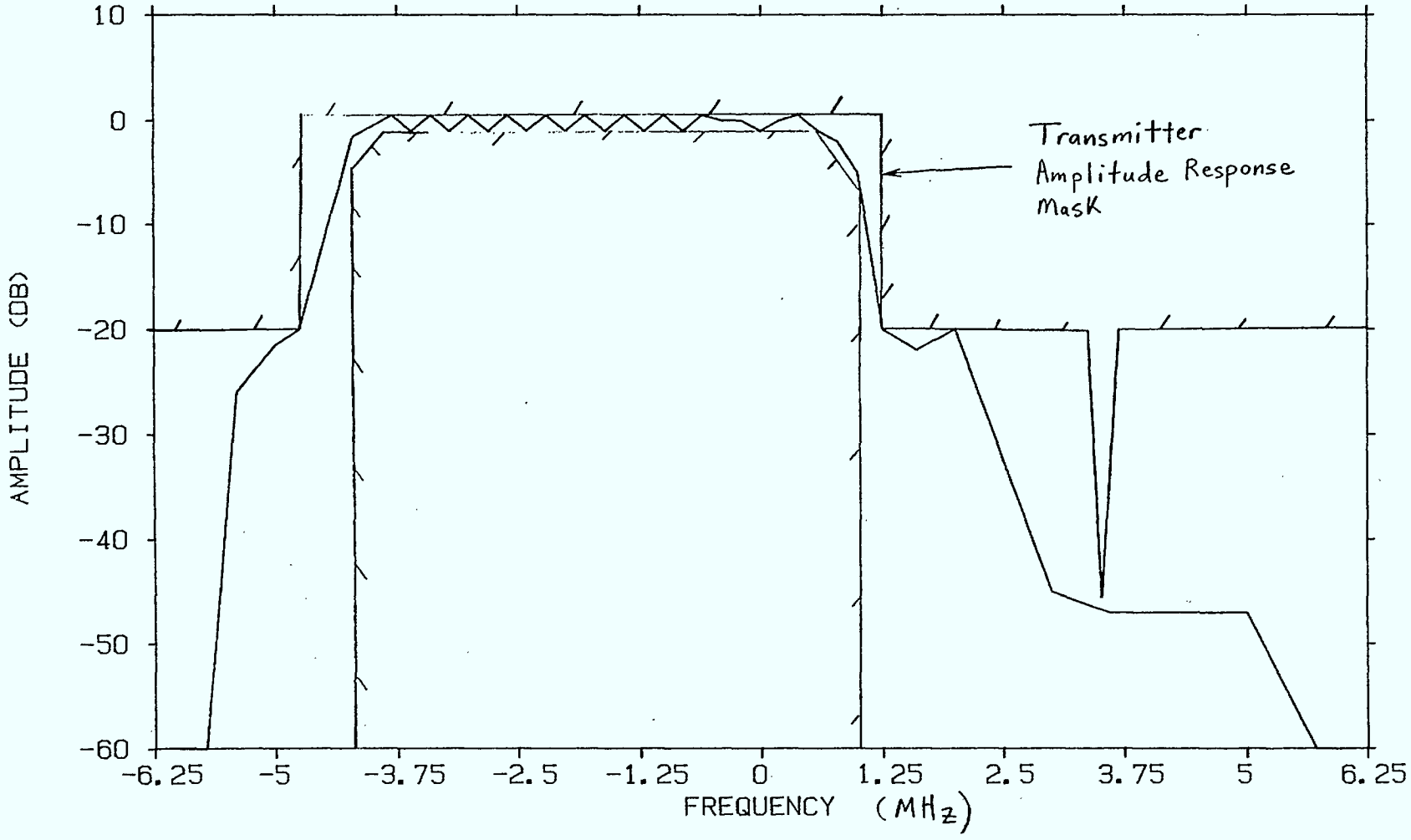


Figure 6.23 : Amplitude Response of POSTX Filter, showing Amplitude Response Mask.

The nominal transmitter and receiver characteristics are incorporated as baseline responses and should be indicative of performance obtained with well maintained transmission equipment and high quality station demodulator receivers.

6.2 Simulation Results With Database Transmit And Receive Filters

Simulation results for database transmit and receive filters are presented for thermal noise environments for for both ideal channel and multipath environments in this section. The four transmit and receive filter combinations considered are:

(1) NOMINALTX, NOMINALRX (baseline zero phase responses)

(2) EIATX, EIARX

(3) POSTX, EIARX

(4) POSTX, POSRX

The complex baseband lab measurement impulse response supplied by CRC personnel arrived too late to be included in the database for these simulation runs. As we will see in the next section, this complex baseband impulse response was incorporated in the simulation program and validation simulation runs performed for an ideal channel (back-to-back test setup). The BS-14 pulse shape, which consists of an RC-100 spectrum cascaded with a 4 MHz lowpass filter, was used in these simulation runs. The averaging slicing level decoder MA-MZ was used in the simulation runs considered in this section.

The multipath channels considered for these simulation runs are the PDUR - 13 dB and PDUR - 20 dB channels considered in the performance analysis runs conducted in [1]. The dominant echoes of these channels are illustrated in Figure 6.24. The PDUR-13 dB multipath channel is an example of a channel that provides a television picture whose quality is near the acceptability threshold. On the other hand, the video quality of the PDUR-20 dB channel should be fairly good despite the significant close-in echo. Relatively strong close-in echoes can be tolerated because the subjective degradation of picture quality is not very severe for close-in echoes. The longer the echo delay, the worse the effect on picture quality.

In the simulation program the transmit and receive filter frequency responses were normalized to have magnitudes of unity and one-half, respectively, at the picture carrier. This is required to normalize the peak to peak video swing to that of the baseline system with nominal transmitter and receiver responses. With this normalization one ensures that the transmitted carrier powers are the same for all systems and meaningful video signal to noise ratios can be defined. This is essential because the amplitude control constraints, be they power or peak signal level constraints, are defined with respect to the nominal system with a BS-14 pulse shape (an RC-100 pulse shape cascaded with a spectrum limiting 4.0 MHz lowpass filter).

To clarify this, consider the following simple example. Suppose we have a transmit filter whose gain at the carrier happens to be 1 dB. In this case, the peak video (picture) signal will be 1 dB greater than the nominal system. Consequently, one must add an extra dB of noise power in this case to establish the same video signal to noise ratio as the baseline system. However, by constrainting the

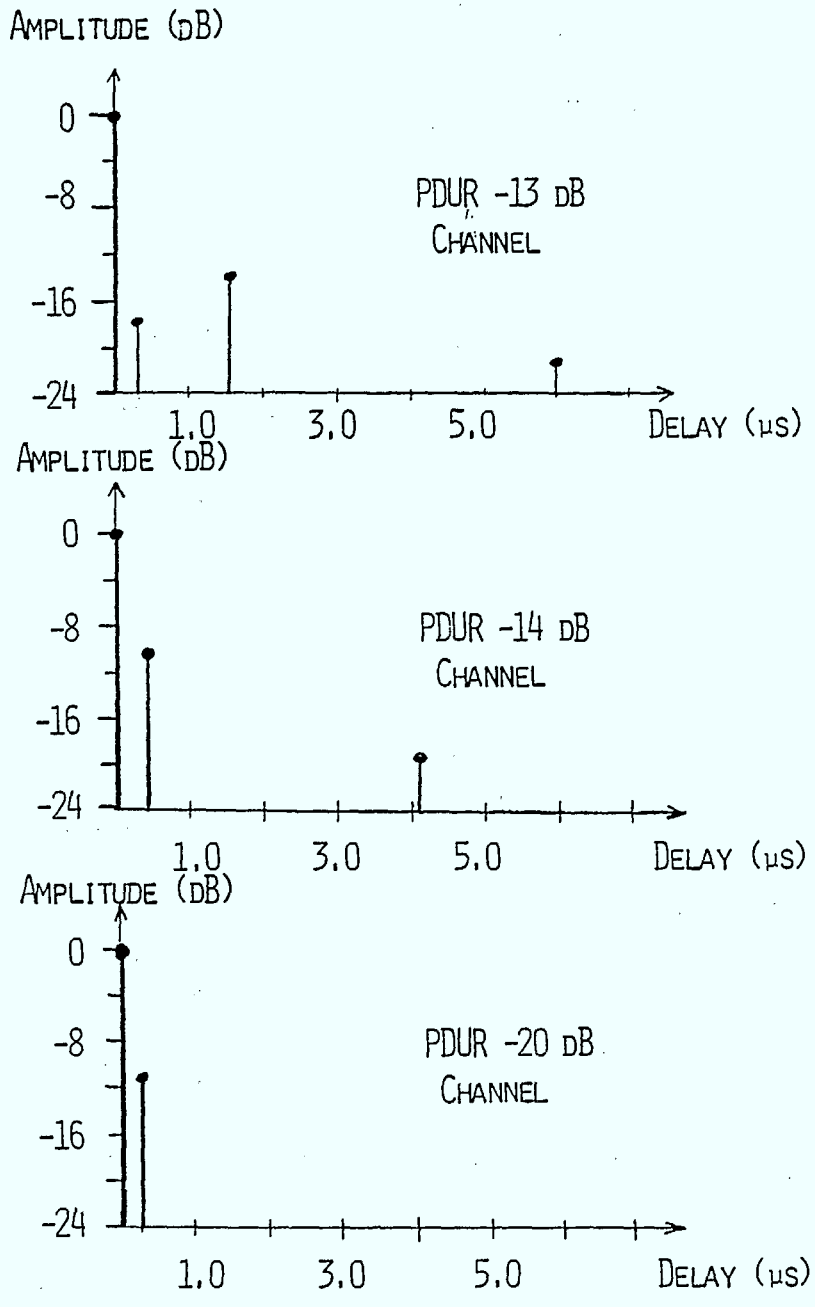


Figure 6.24 : DOMINANT ECHOES OF THREE MULTIPATH CHANNELS (from [1]).

power of the digital data signal at the output of the transmit filter to be the same as that of the nominal system, it is clear that the signal to noise ratio of the data signal will be 1 dB worse than that of the nominal system. It is clear that one cannot constrain the power of the data signal without applying a similar constraint on the carrier power.

The video signal to noise ratio is effectively a carrier to noise power ratio. Consequently, it is not the most accurate indicator of the performance for a digital communication system. For a digital communication system, performance can be predicted based on the ratio of the average power per bit to noise power ratio. Note that the average power per bit is given by an integral of the overall pulse spectrum, and as a consequence depends on the nature of the transmit and receive filters over the entire transmission band. In contrast, the video signal to noise ratio depends only on the transmit and receive filter characteristics around the picture carrier. Consequently, it would be relatively easy to construct transmit and receive filter combinations which provide the same video signal to noise ratio but have considerably different digital signal to noise ratios. The video signal to noise ratio is still useful because it provides a rough indication of system performance that can be fine-tuned with knowledge of the digital signal to noise ratio.

For the ideal channel runs, two amplitude constraints are considered. First, the power at the output of the transmit filter is constrained to be the same as that of the nominal system. The simulation results for this case are presented in Figure 6.25. There are several interesting observations that can be made. First of all, it is evident that performance with the measured transmitter characteristic from St. Louis, Missouri (EIATX) is considerably worse than

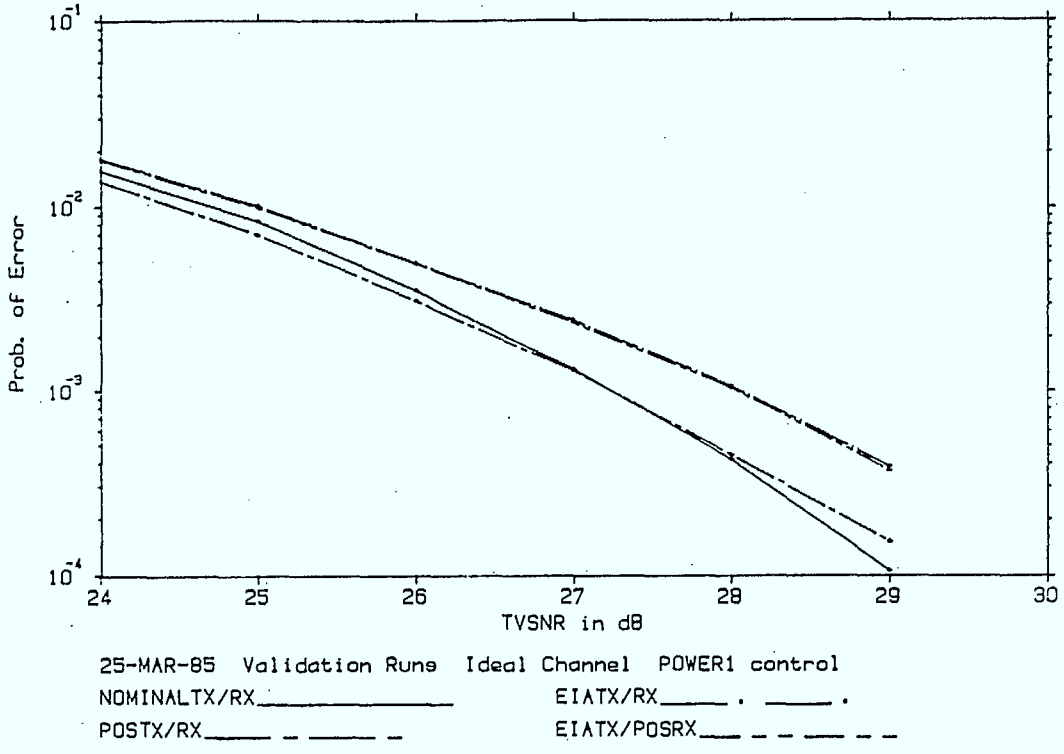
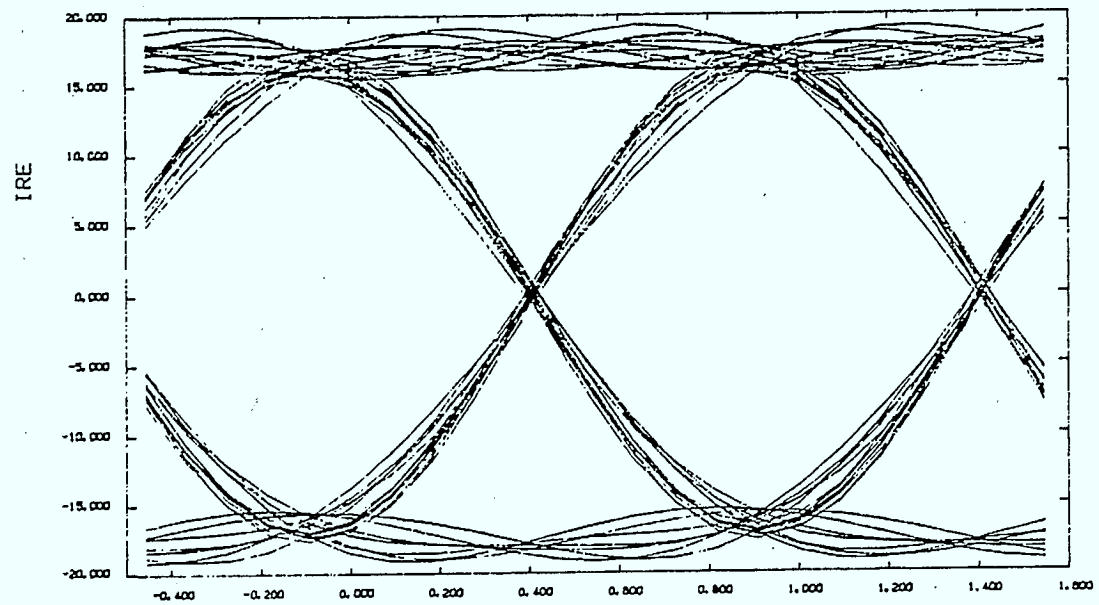


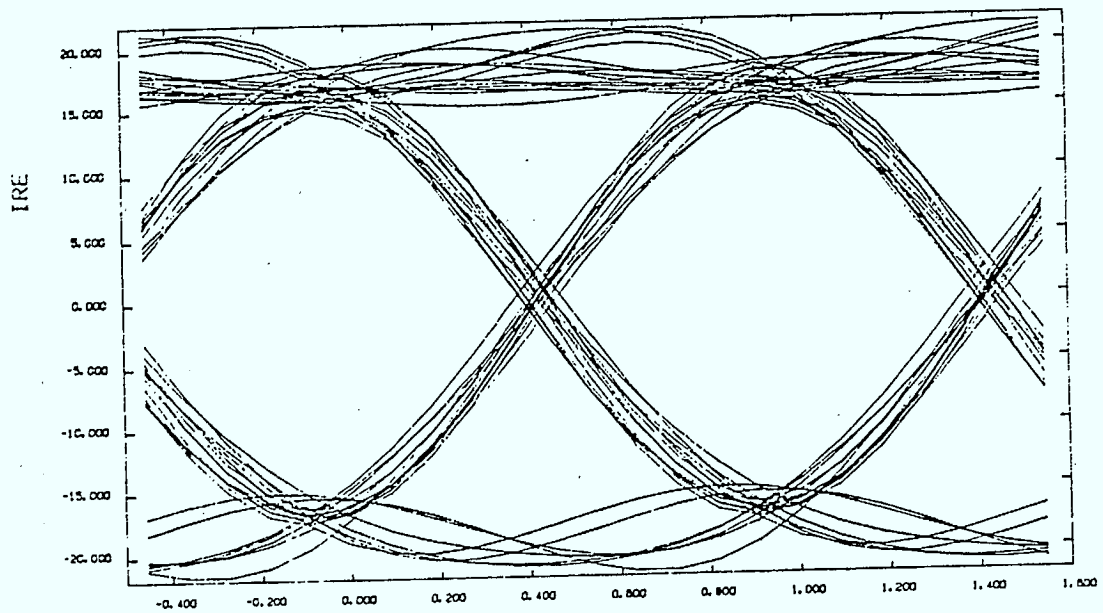
Figure 6.25 : Simulation Results with Transmitter Power Constrained

performance with a nominal transmit filter or the hypothetical transmitter response POSTX. The degradation is roughly 0.8 dB at a weighted video signal to noise ratio of 27 dB. Another interesting observation is that performance of the EIATX-EIARX and EIATX-POSRX combinations is virtually identical. This indicates that the ±30 nsec fast group delay ripple added to the EIARX response to form a more realistic receiver response does not introduces a noticeable performance degradation.

Probably the most surprising result is that performance with the combination POSTX-POSRX is actually slightly better than that obtained with the nominal transmitter and receiver responses (for SNR_{TV}=27 dB). At higher signal to noise ratios, performance is best with nominal zero phase transmitter and receiver responses. The eye diagrams for the nominal transmitter and receiver responses and for the POSTX-POSRX combination are supplied in Figure 6.26. It is evident that the eye opening is greater for the nominal filter combination, which is consistent with its superior performance at higher signal to noise ratios. Note also, that the thickness of the eye is greatest for the POSTX-POSRX combination. When the signal to noise ratio is relatively low, this can result in a performance improvement over a system with a narrower eye. When the noise is strong, ISI components other than those coresponding to significant eye closure can introduce significant terms in the probability of error expression. Furthermore, if there is a greater percentage of high components (broader eye) one can actually achieve slightly better performance (lower error rate). However, one is usually not to concerned with this scenario, because performance is usually not particularly good when the noise is strong enough for this phenomenon to occur. This is



(a) NOMINALTX - NOMINALRX



(b) POSTX - POSRX

Figure 6.26 : Eye Diagrams for the Nominal Transmitter and Receiver Filter and POSTX-POSRX Receiver/Transmitter Combinations

definitely the case here, and the system with nominal responses is superior at higher signal to noise ratios where system performance is approaching acceptability.

An examination of the eye diagram for the EIATX-EIARX combination, provided in Figure 6.27, clearly indicates why system performance is degraded with this filter combination. Note that in addition to significant eye closure, the eye diagram is skewed (asymmetrical). The amplitude response ripple of the EIATX filter is not excessive, being comparable to that of the POSTX response. Since there is a considerable performance difference between systems incorporating these responses, it must be being caused by the group delay response of EIATX. The group delay responses of the EIATX and POSTX responses are provided on the same graph for comparison purposes in Figure 6.28. The differences above f=1 MHz are of little consequence. It is interesting to note that the group delay excursions are, for the most part, smaller for the EIATX response than those of the POSTX response, which oscillates from one extreme of the group delay mask to the other.

The major degradation is invariably being caused by the slowly varying nature of the EIATX group delay characteristic. It appears that relatively fast, periodic group delay ripple that has equal magnitude excursions in the positive and negative directions tends to have a cancelling effect. This is consistent with the observation that the fast ripple superimposed on the EIARX receiver response, when forming the POSRX response, did not introduce any noticeable performance degradation. The baseband group delay of Figures 6.4 and 6.5 of the DOC

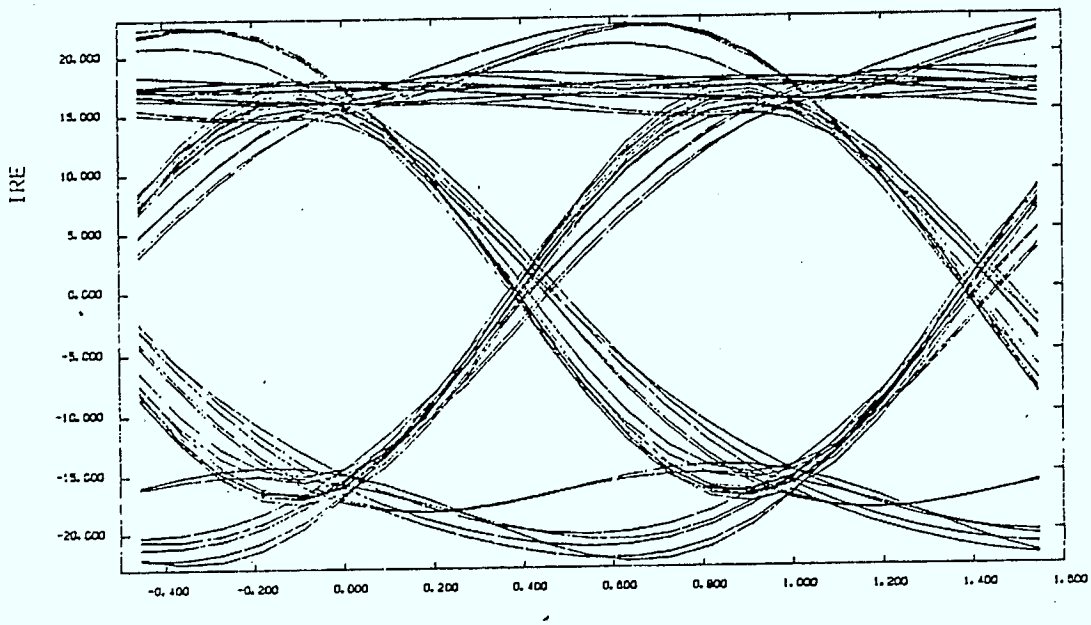


Figure 6.27 : Eye Diagram for EIATX-EIARX Transmitter/Receiver Filter Combination

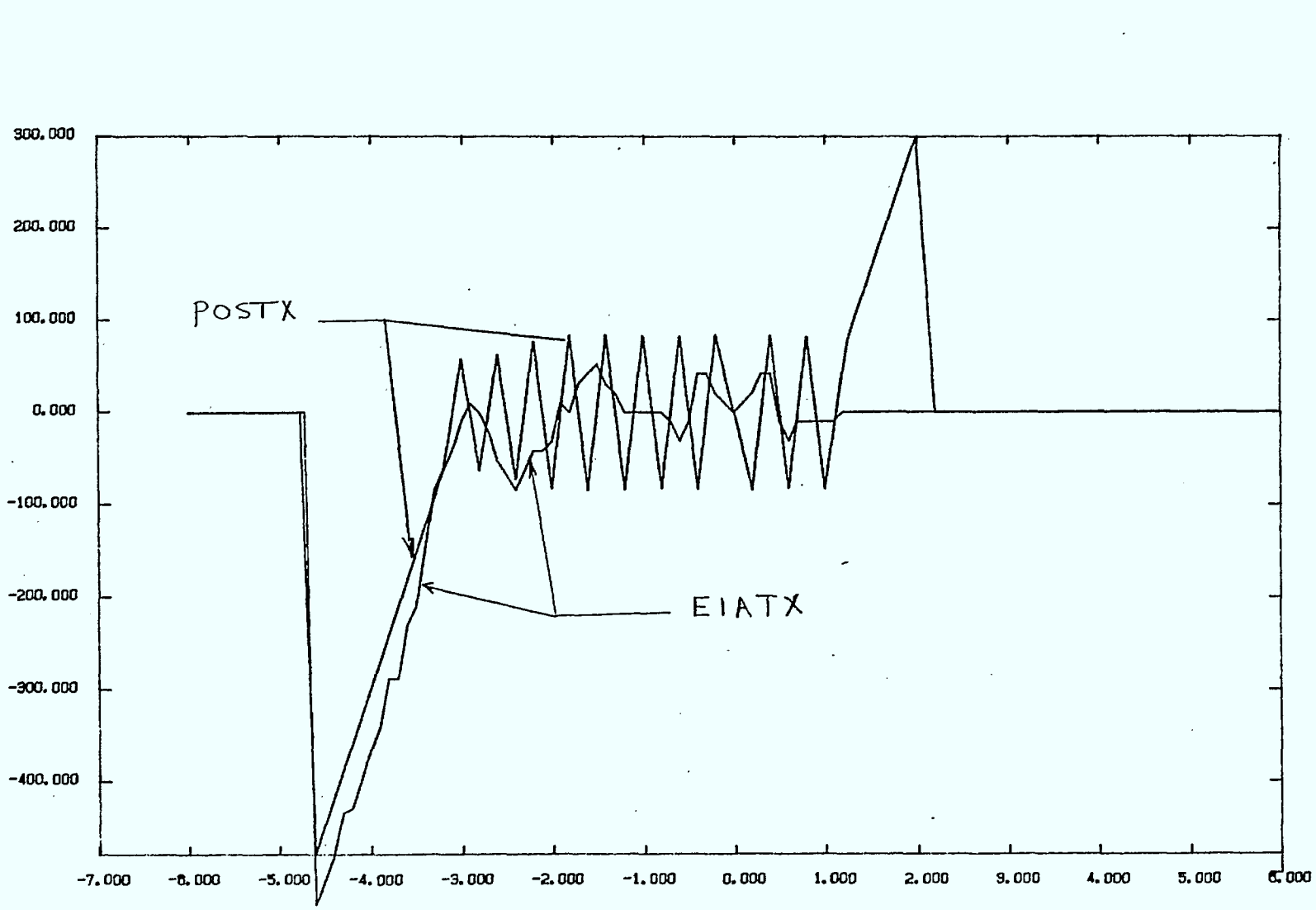
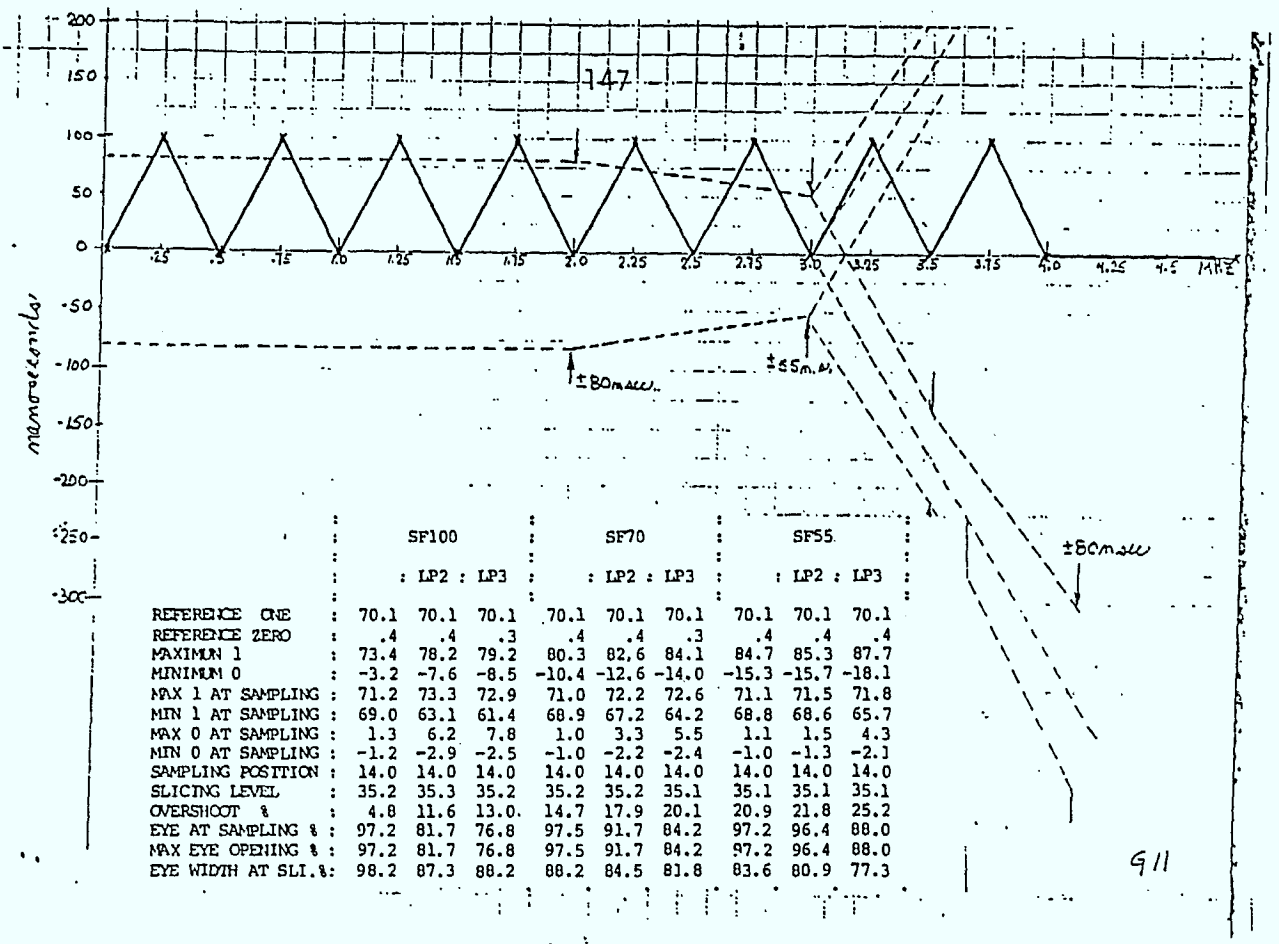


Figure 6.28 : Comparison of Group Delay Characteristics of the EIATX and POSTX Transmitter Filters.

simulation and their corresponding eye diagrams, Figures 6.8 and 6.9, also serve to demonstrate that slowly varying group delay can introduce appreciable distortion. Furthermore, in the baseband DOC simulation [13] it was found that the eye opening for the relatively fast rippling group delay function presented in Figures 6.29 is 71.6%. This is considerably better than the 44.9% eye opening for the slowly varying group delay function of Figure 6.5.

It is apparent from Figure 6.25 that significant performance degradations relative to the ideal baseline system can be introduced by typical transmitter and receiver responses (on the order of a decibel). It is also clear that the amount of degradation is strongly dependent on the group delay characteristics of the transmitter - receiver cascade. It is also interesting to note that the performance variations that can be introduced by typical transmitter and receiver characteristics are slightly greater than the performance improvement found for the recommended teletext pulse shape discussed in [1]. Given the degree of degradation that can be imposed by transmitters that are within the tolerance specs, it would be worthwhile measuring a number of channel responses for different transmitter sites in the next phase of the measurement program. This measurement data could then be incorporated in the simulation program, and typical and worst case performance degradations relative to the baseline system could be established.

The simulation results for the ideal channel case when the peak signal levels at the output of the receiver are constrained to be the same as those applicable for a nominal system with a BS-14 pulse shape are provided in



	SF100			SF70			SF55		
		LP2	LP3		LP2	LP3		LP2	LP3
REFERENCE ONE	70.1	70.1	70.1	70.1	70.1	70.1	70.1	70.1	70.1
REFERENCE ZERO	.4	.4	.3	.4	.4	.3	.4	.4	.4
MAXIMUN 1	73.4	78.2	79.2	80.3	82.6	84.1	84.7	85.3	87.7
MINIMUN 0	-3.2	-7.6	-8.5	-10.4	-12.6	-14.0	-15.3	-15.7	-18.1
MAX 1 AT SAMPLING	71.2	73.3	72.9	71.0	72.2	72.6	71.1	71.5	71.8
MIN 1 AT SAMPLING	69.0	63.1	61.4	68.9	67.2	64.2	68.8	68.6	65.7
MAX 0 AT SAMPLING	1.3	6.2	7.8	1.0	3.3	5.5	1.1	1.5	4.3
MIN 0 AT SAMPLING	-1.2	-2.9	-2.5	-1.0	-2.2	-2.4	-1.0	-1.3	-2.1
SAMPLING POSITION	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0	14.0
SLICING LEVEL	35.2	35.3	35.2	35.2	35.2	35.1	35.1	35.1	35.1
OVERSHOOT %	4.8	11.6	13.0	14.7	17.9	20.1	20.9	21.8	25.2
EYE AT SAMPLING %	97.2	81.7	76.8	97.5	91.7	84.2	97.2	96.4	88.0
MAX EYE OPENING %	97.2	81.7	76.8	97.5	91.7	84.2	97.2	96.4	88.0
EYE WIDTH AT SLI.%	98.2	87.3	88.2	88.2	84.5	81.8	83.6	80.9	77.3

Figure 6.29 : Rippling Group Delay Response and Resulting Eye Diagram Performance Factors (from [13]).

Figure 6.30. Note that in this case, there is roughly a 0.4 dB degradation for the POSTX-POSRX combination relative to the nominal system. The performance with the EIATX-EIARX and EIATX-POSTX combinations are once again virtually identical. The degradation that can be attributed to these transmitter and receiver combinations is approximately 1.5 dB in this case. An examination of the eye diagram when the EIATX transmitter characteristic is included (see Figure 6.27) indicates that the peak overshoots are considerably worse than for nominal system parameters. Consequently, the degree of degradation indicated in Figure 6.30 when the EIATX transmitter is included is quite realistic.

The average bit error rate performance for a thermal noise environment and a PDUR-13 dB multipath channel is presented in Figure 6.31. For these simulation runs, the power at the output of the transmit filter was constrained. In this case, performance of the POSTX-POSRX combination matches that of the baseline system. A comparison of the multipath simulation results to those for an ideal channel (see Figure 6.25) clearly indicates that multipath introduces a significant performance degradation. Video signal to noise ratios in the neighbourhood of 32 dB are required to obtain average error rates around $1x10^{-3}$ for this multipath channel, while this error rate is attained with video signal to noise ratios in the 27.5-28 dB range with an ideal channel. It is also evident that introducing the high rate group delay response ripple on the EIARX response did not result in any noticeable degradation. The other interesting thing to note from Figure 6.31 is that the degradation introduced by the transmitter and receiver responses persists and is not masked or totally dominated by multipath. There is roughly a half-dB degradation imposed by the EIATX response that persists in multipath.

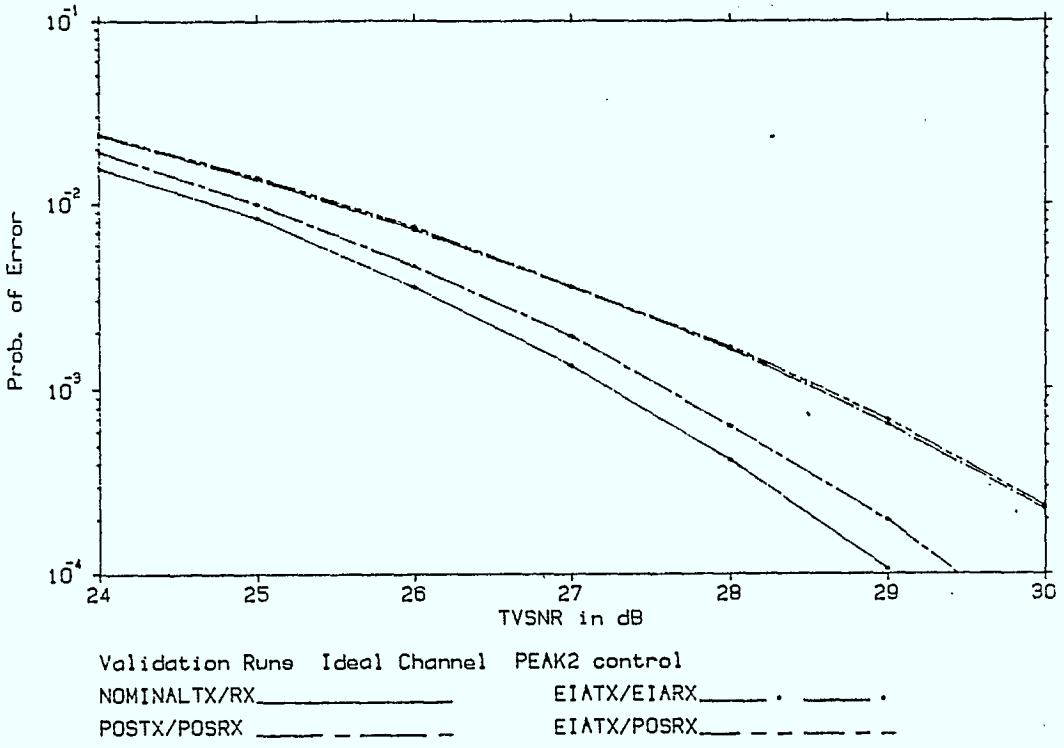


Figure 6.30 : Simulation Results with Output Peak Signal Levels Constrained

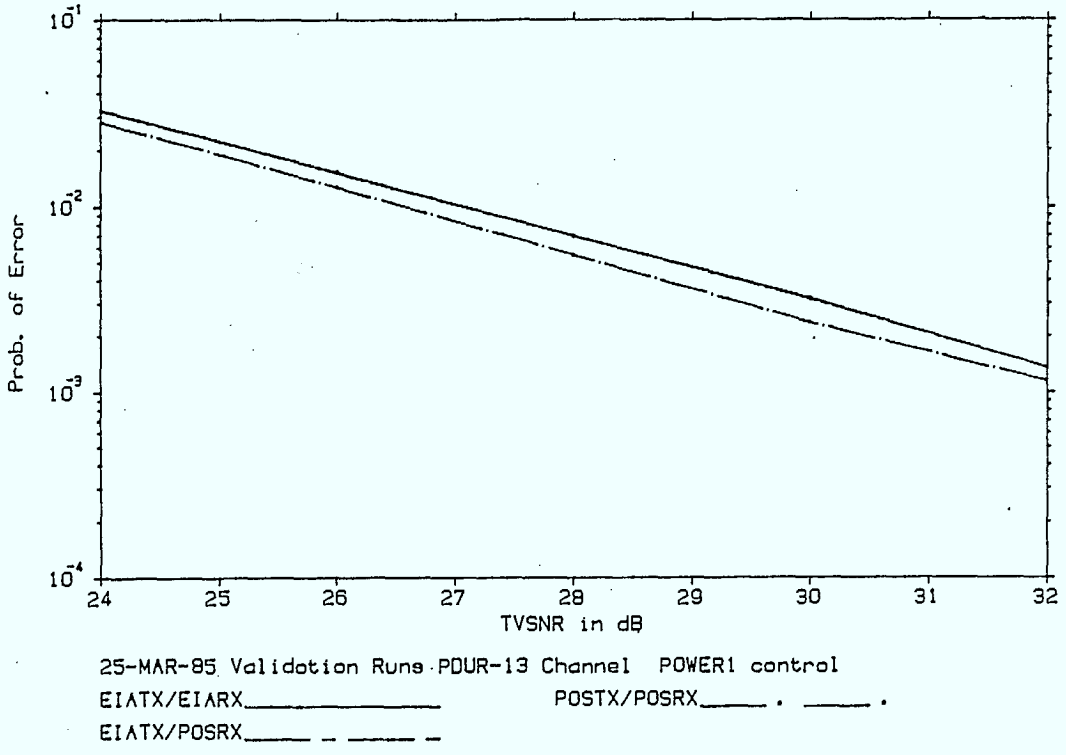


Figure 6.31 : Average BER for Thermal Noise Environment and PDUR-13 dB Multipath Channel

The simulation results for a thermal noise environment and a PDUR-20 dB multipath channel are presented in Figure 6.32. Once again, the fast group delay ripple on the POSRX response introduces no degradation relative to the EIARX response. The performance of the POSTX-POSRX combination matches the performance with nominal filter characteristics. The behaviour of the error rate curves is considerably different for this multipath channel. At low video signal to noise ratios, the degradation imposed by the EIATX response is relatively minor. However, above a video signal to noise ratio of 28 dB, the performance curve for the EIATX response essentially levels off, diverging from the error rate curve for the POSTX-POSRX combination. Consequently, the loss relative to the POSTX-POSRX combination becomes quite large at high video signal to noise ratios. With the EIATX transmitter response, performance levels off at an error rate of 5×10^{-2}. This is rather dismal performance, clearly indicating the need for some form of equalization to get satisfactory teletext performance over useable television channels.

6.3 Simulation Results Incorporating Lab Measurement Data

This section presents simulation results incorporating lab measurement data, for an ideal channel (back-to-back test setup) and thermal noise environment. We begin by describing the incorporation of lab measurement data in the simulation program.

6.3.1 Incorporating Lab Measurement Data in the Simulation Program

The complex baeband impulse response for an ideal (back-to-back) laboratory setup, which includes a Jerrold Commander IV channel 11 cable modulator and a Tektronix receiver which provides both inphase and quadurature

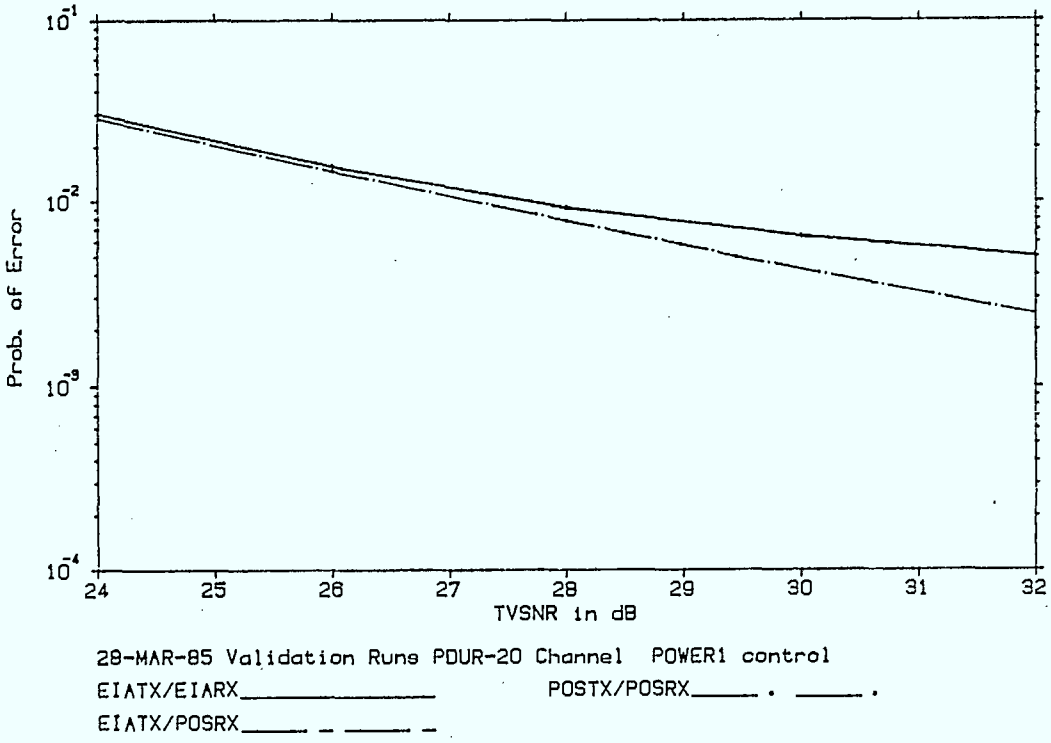
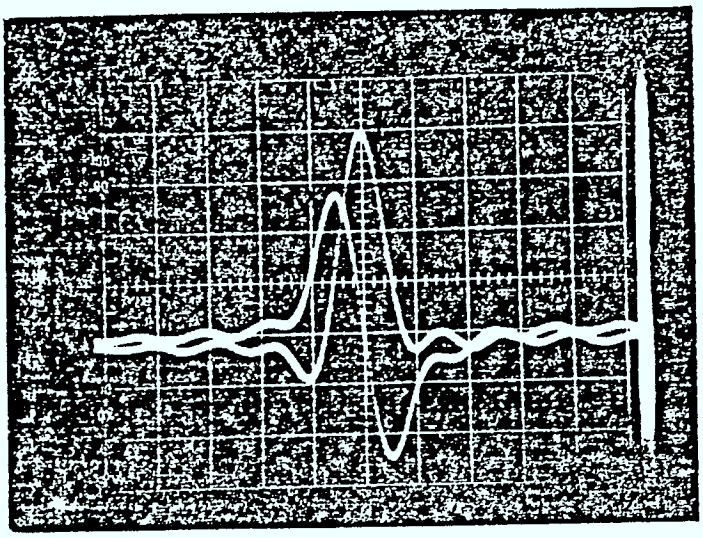


Figure 6.32 : Average BER for Thermal Noise Environment and PDUR-20 dB Multipath Channel

components, was measured with an improved complex baseband impulse response extraction system. Unfortunately, difficulties involved in modifying the system to extract the complex impulse response delayed the delivery of this measurement data. It was not until March 25, 1985 that we were confident that the extracted complex baseband impulse response was, in fact, correct.

A photograph of the 1-T responses of the inphase and quadrature components are supplied in Figure 6.33. The inphase and quadrature components of the complex baseband impulse responses are supplied in Table 6.6. From this table, we see that ratio of peak inphase component to peak quadrature component is approximately 1.34. This is reasonable agreement with the 1.56 factor obtained with the Telidon channel simulation with nominal transmit and receive filter characteristics and a BS-14 pulse shape.

The impulse response sample spacing is $1/2r_o$, where r_o is the data rate (5.727272 Mbps), and is truncated to 41 samples. The amplitude and group delay responses of this end-to-end vestigial sideband frequency response are supplied in Figure 6.34. It is apparent from the smoothness and the relatively minor group delay deviations over the channel bandwidth that the equipment used in the CRC laboratory is of high quality. The end-to-end amplitude and group delay responses with the EIATX-EIARX and POSTX-POSRX transmitter and receiver responses with a spectrum limited 100% raised cosine pulse shape spectrum are presented in Figure 6.35 and 6.36, respectively. Comparing these responses to that of the lab test setup presented in Figure 6.34, it is apparent that the database candidates have much worse group delay characteristics than



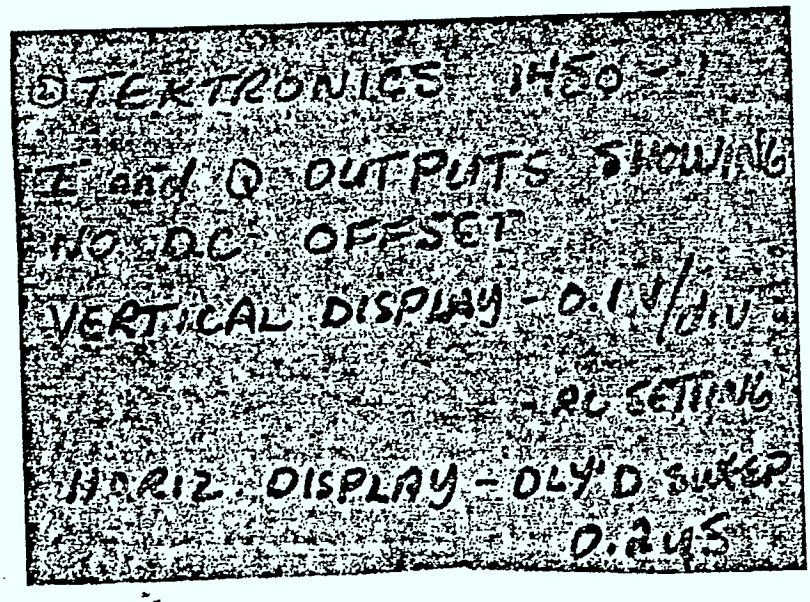
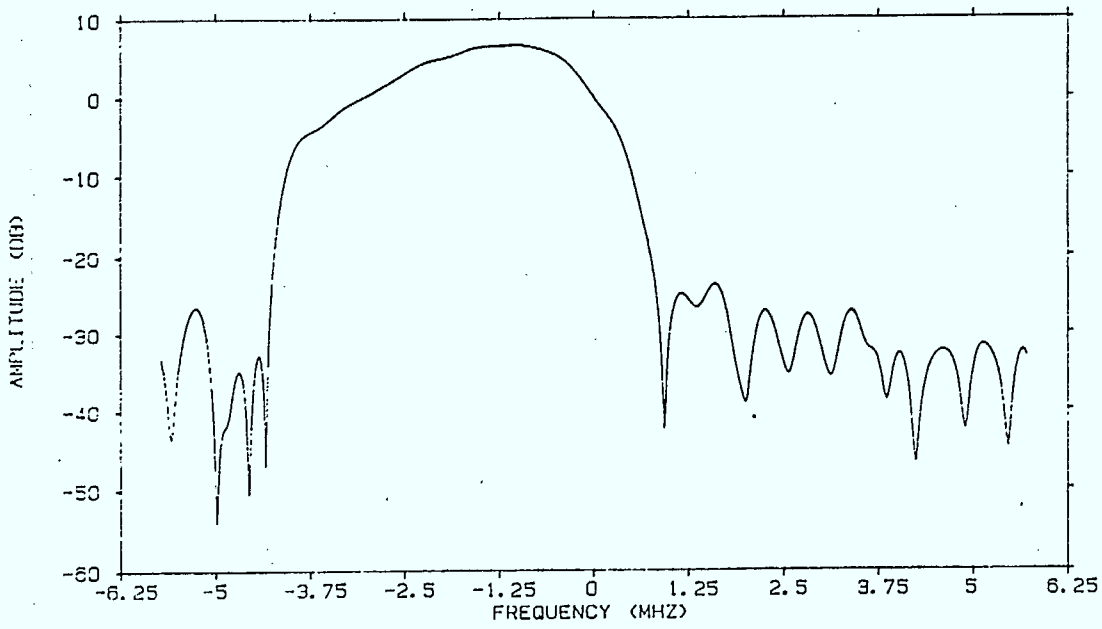


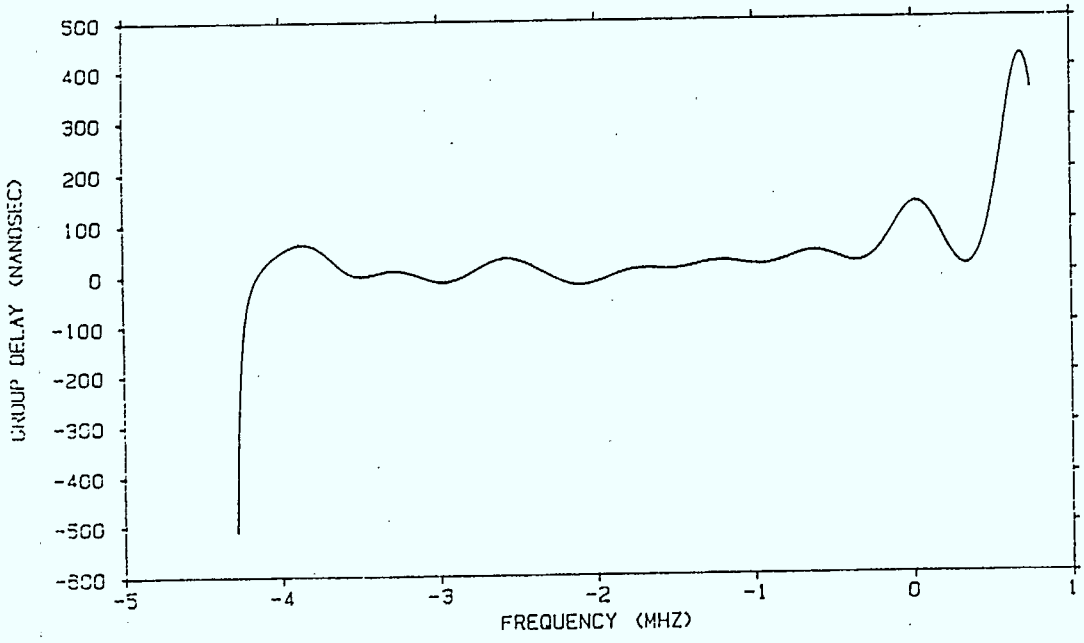
Figure 6.33 : Inphase and quadrature 1-T responses from Tektronics demodular.

INPHASE	QUADRATURE
-4.3157036E-03 ,	5.9935530E-03
-1.2142294E-02 ,	-1.3309010E-02
-1.6396290E-02 ,	5.0847948E-04
-9.6388714E-04 ,	-3.6691979E-03
-1.1581439E-02 ,	-1.6274550E-03
1.0805702E-03 ,	3.0008161E-03
-2.1433416E-03 ,	-8.5316030E-03
-1.9454805E-02 ,	-1.2555850E-02
-3.2689814E-03 ,	4.2204219E-03
-2.6191808E-03 ,	-7.8604091E-03
-1.0983869E-02 ,	-5.0394358E-03
-3.6759785E-04 ,	3.4948571E-03
-2.1809118E-02 ,	-1.5579980E-02
-1.4032471E-02 ,	3.0760661E-02
9.7104460E-03 ,	4.6257409E-03
-3.5533540E-02 ,	5.5084391E-03
-2.5781592E-02 ,	7.0734113E-02
-7.9629146E-02 ,	9.1865331E-02
-0.1748815 ,	0.4148560
0.3592850 ,	0.8417303
1.130174 ,	0.3578053
0.8685605 ,	-0.5672801
0.1454648 ,	-0.6215772
-5.2358494E-03 ,	-0.2546052
4.0689114E-02 ,	-0.1568539
-1.8970996E-02 ,	-0.1028123
1.7661629E-02 ,	-1.6470140E-02
1.9618671E-02 ,	-4.0581521E-02
-1.4319210E-02 ,	8.0511849E-03
1.7640254E-02 ,	1.4249750E-02
3.9482112E-03 ,	-1.4122540E-02
-1.7569816E-02 ,	3.8088779E-03
8.0415042E-04 ,	9.9164702E-04
-8.0142338E-03 ,	2.4466130E-03
-4.4260388E-03 ,	1.5511320E-02
1.0843749E-02 ,	-6.9276849E-03
-5.8697960E-03 ,	-6.0075689E-03
-3.2361506E-03 ,	6.1972998E-03
2.7512303E-03 ,	-8.1878733E-03
-1.2723930E-03 ,	-1.0185450E-03
6.1530946E-03 ,	8.4984787E-03

Table 6.6 : Laboratory Impulse Response Data

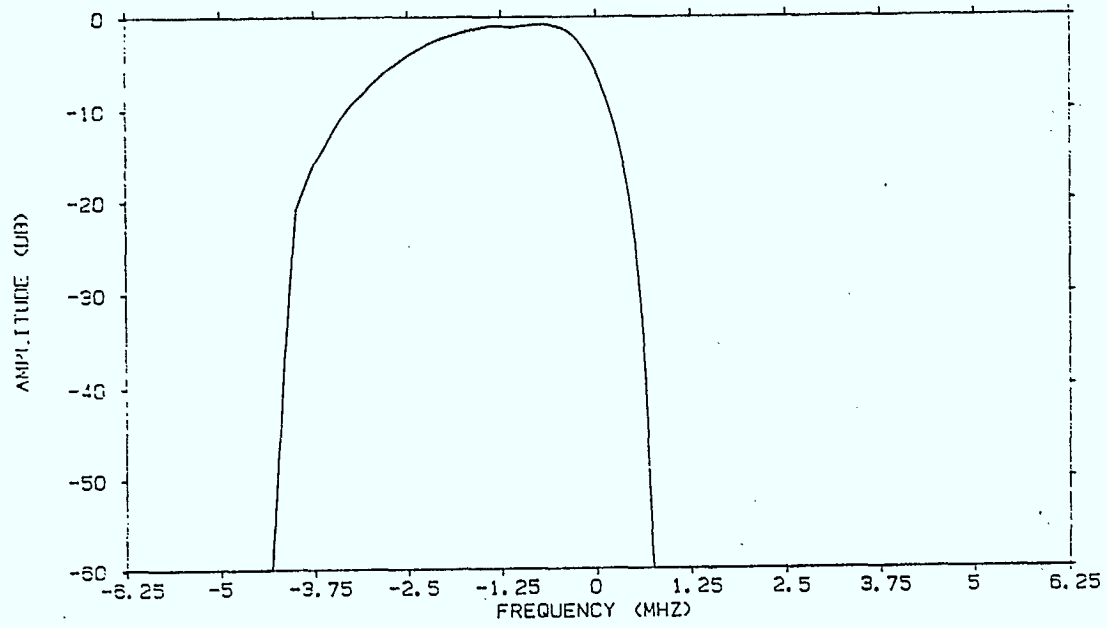


(a) Amplitude Response

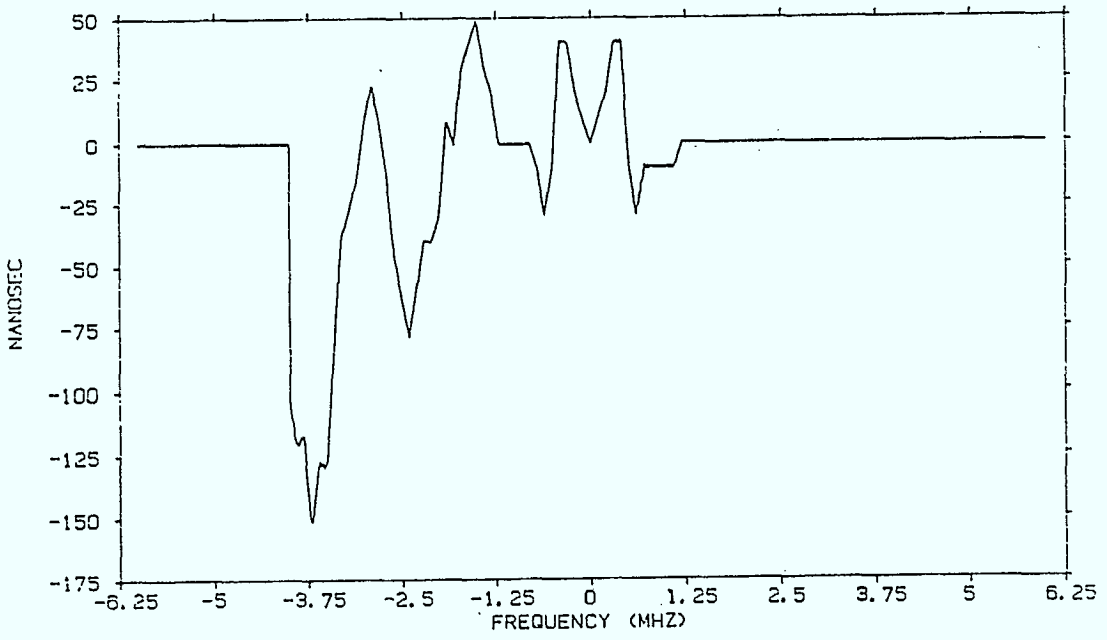


(b) Group Delay Response

Figure 6.34 : Amplitude and Group Delay Responses of the Frequency Response Corresponding to the Extracted End-to-End Complex Baseband Impulse Response with an Ideal Channel.

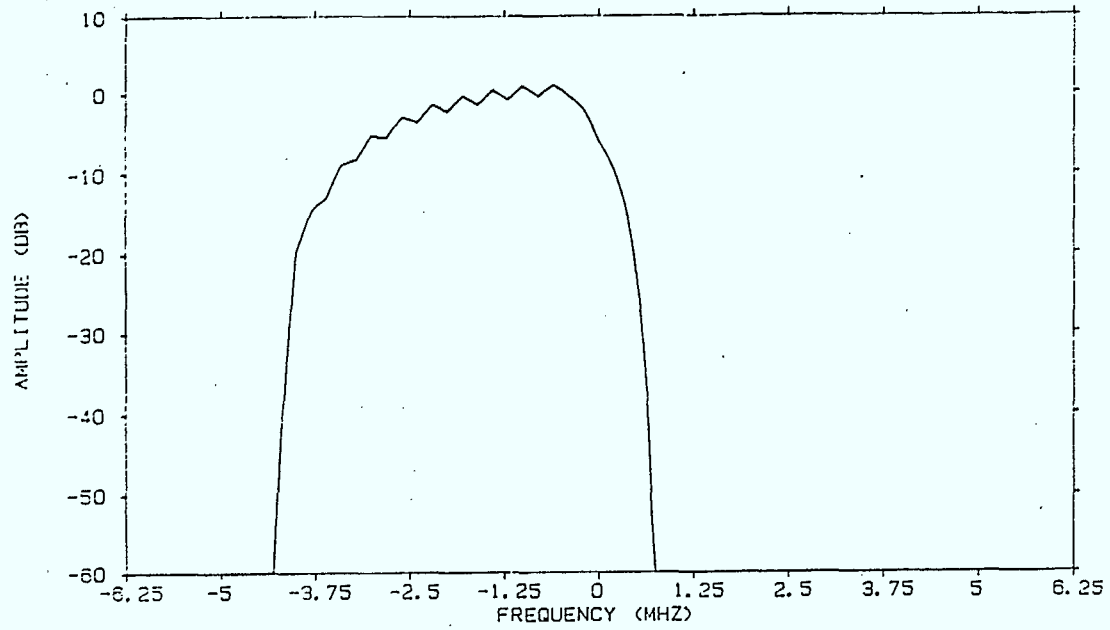


(a) Amplitude Response

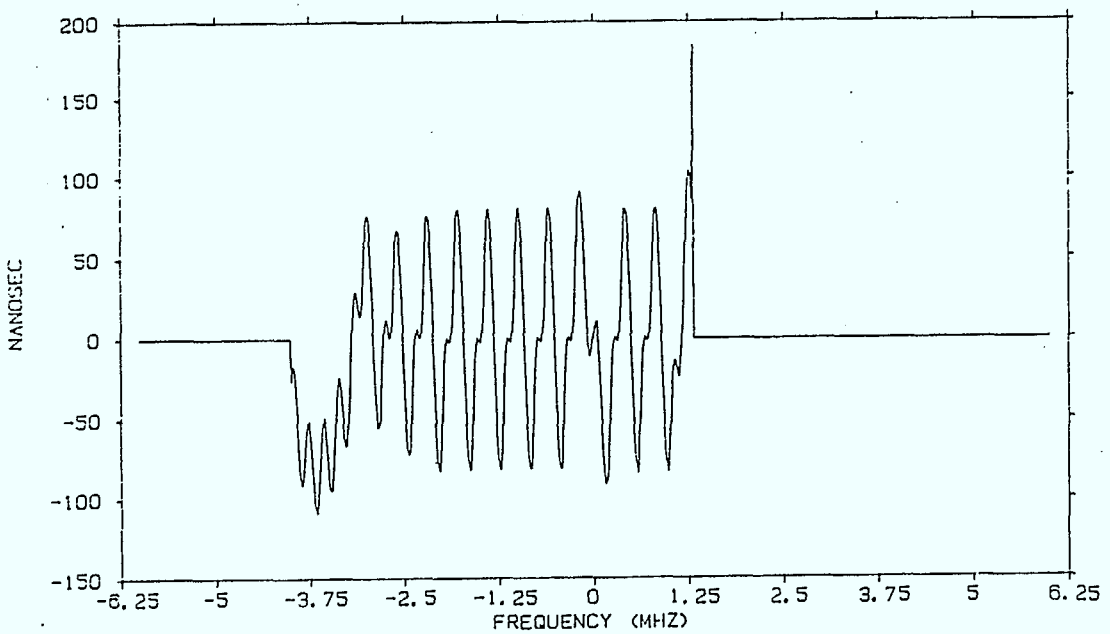


(b) Group Delay Response

Figure 6.35 : Amplitude and Group Delay Response of End-to-End Channel Response Using EIA Filters.



(a) Amplitude Response



(b) Group Delay Response

Figure 6.36 : Amplitude and Group Delay Response of End-to-End Channel Response Using POS Filters.

the high quality CRC lab measurement equipment, and are likely more representative of actual broadcast transmitters and commercially available home television receivers.

The end-to-end impulse response was extracted under ideal conditions (no externally injected noise, only quantization noise). To incorporate the lab measurement data into the simulation, one must factor out the RC-100 pulse shape. This is performed using the frequency domain spectral division approach described in [18]. The provided impulse response is zero stuffed to 1024 samples, converted to the frequency domain via an FFT, and the resulting end to end frequency response is divided by the pulse shape spectrum, and the resultant normalized to 0 dB at zero frequency.

This spectral division approach should be relatively good in this case because the RC-100 spectrum does not become extremely small over the channel bandwidth, and the extracted impulse noise is not corrupted by excessive amounts of measurement noise. Consequently, the noise enhancement property of the spectral division approach should not be too bad in this case. It should be emphasized that this spectral division approach is not recommended in all circumstances. For example, for factoring out the multipath channel characteristics from an end to end complex baseband impulse, with knowledge of the end to end impulse response for a line of sight channel, the system model illustrated in Figure 6.37 is appropriate. It is clear from this model that one could extract the complex baseband multipath channel characteristic using the least mean square impulse response extraction algorithms used to extract the end to end complex baseband impulse response (with knowledge of the transmitted data sequence and received signal). It is recommended that this approach be used in the next measurement program for extracting the multipath characteristics.

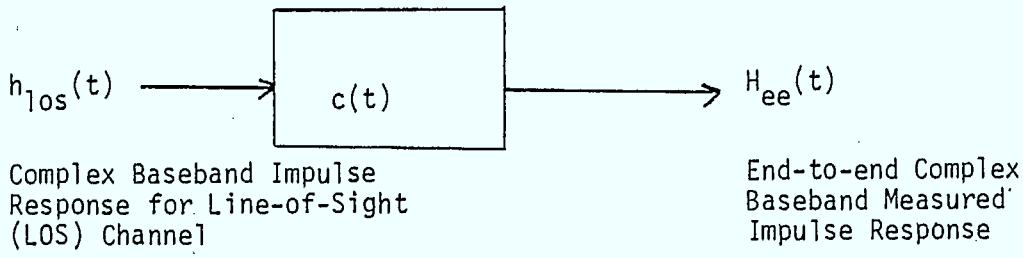


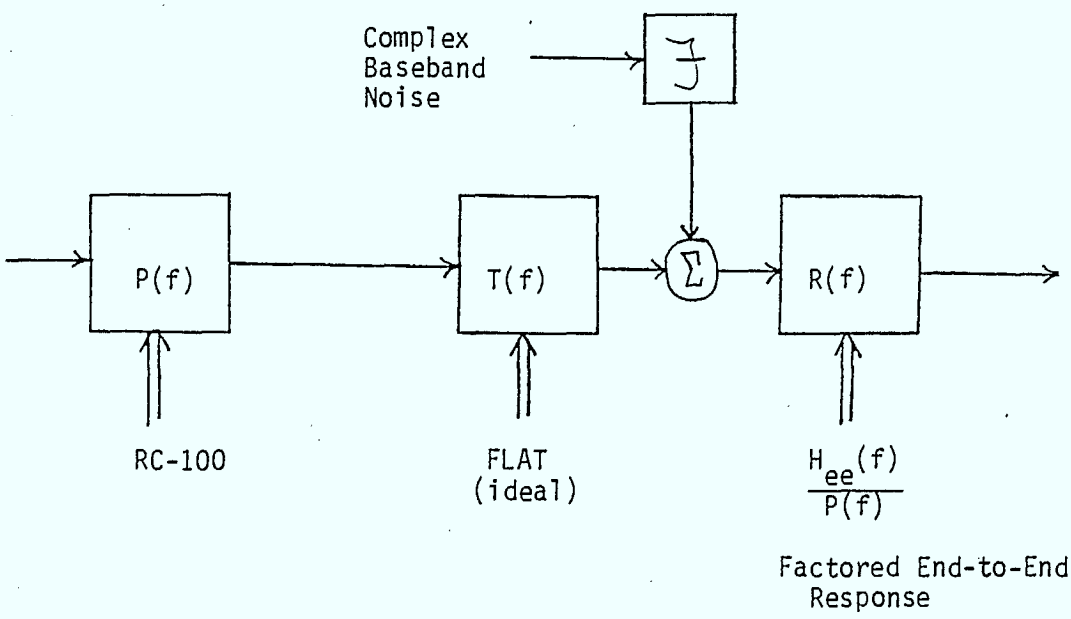
Figure 6.37 : System Model for Factorization of Multipath Channel from End-to-End Channel.

In the simulation program, the factored response is applied as the receiver response (see Figure 6.38) for two reasons. First of all, the characterisitcs of the narrower bandwidth receiver filter will dominate the cascaded or end-to-end response. Secondly, one needs a realistic vestigial sideband post noise filter.

6.3.2 Lab Measurement Validation Runs

The intention was to validate our implementation of the averaging slicing level MA-MZ decoder model against a hardware implementation of this decoder operating in a controlled, ideal (back to back) laboratory environment. The modulator and receiver used to perform these measurements are of high quality. The error rates for several signal to noise ratios could be measured and compared to simulated performance.

Unfortunately, an operational version of this decoder had not been received by CRC by the completion date of this contract. The performance of the decoders tested by CRC was dismal, being considerably worse than the peak detector decoder used in the field measurement program. Since performance of the averaging slicing level decoder should be considerably better than the peak detector decoder, a decoder fault of some description must be responsible for the disappointing performance. The author had the opportunity of examining the sampling instants provided by the symbol synchronization circuitry of this decoder in the laboratory at CRC. It appeared that there was a significant offset (approximately 25% of a bit period) from the optimal sampling instant. An examination of a typical eye diagram of the data communications channel (see Figure 6.26, for example) reveals that such an offset would introduce a substantial performance degradation. It



$H_{ee}(f)$ == measured end-to-end VSB channel frequency response

Figure 6.38 : Application of Filters Incorporating Lab Measurement Data

also appeared that the sampling clock would often miss sampling the first bit of the "word" sync byte. This is likely a consequence of the positioning of the clock sync window. This window appears to be positioned very near the end of the clock sync signal. It is conjectured that performance could be improved by shifting this window to a position earlier in clock sync signal.

In the absence of measurement data for the MA-MZ decoder, the simulated performance will be compared to baseline decoder performance quoted by the manufacturer. The quoted performance is supplied in Table 6.7 [19]. Our simulation program has been designed to use weighted video signal to noise ratios [2]. For thermal noise environments, the weighted signal to ratio can be suitably approximated by adding 6.1 dB to the unweighted video signal to noise ratio [20], [21]. The simulated performance with the lab measurement data is compared with the quoted performance spec in Figure 6.39. For weighted video signal to noise ratios less than 27 dB, simulated performance is in reasonable agreement with the specified performance. As the video signal to noise ratio is increased the two performance curves start to separate, with simulated performance being roughly 0.8 to 0.9 dB better than the performance quoted by the manufacturer. Given the relatively ideal implementations that can be incorporated in simulation programs, it is not surprising that simulated performance would surpass the measured performance of a hardware decoder implementation. Furthermore, and implementation loss on the order of 1 dB is not unreasonable.

In Figure 6.40, the simulated performance obtained with the measured lab impulse response is compared to the performance obtained with the baseline "zero phase" nominal transmitter and receiver responses. It can be seen that

Video SNR (unweighted)	BER
20	$3x10^{-3}$
21	$1x10^{-3}$
22	$5x10^{-4}$
23	$3x10^{-4}$
24	$1x10^{-4}$
25	$2x10^{-5}$
27	0

Table 6.7: Baseline averaging slicer decoder performance quoted by the manufacturer.

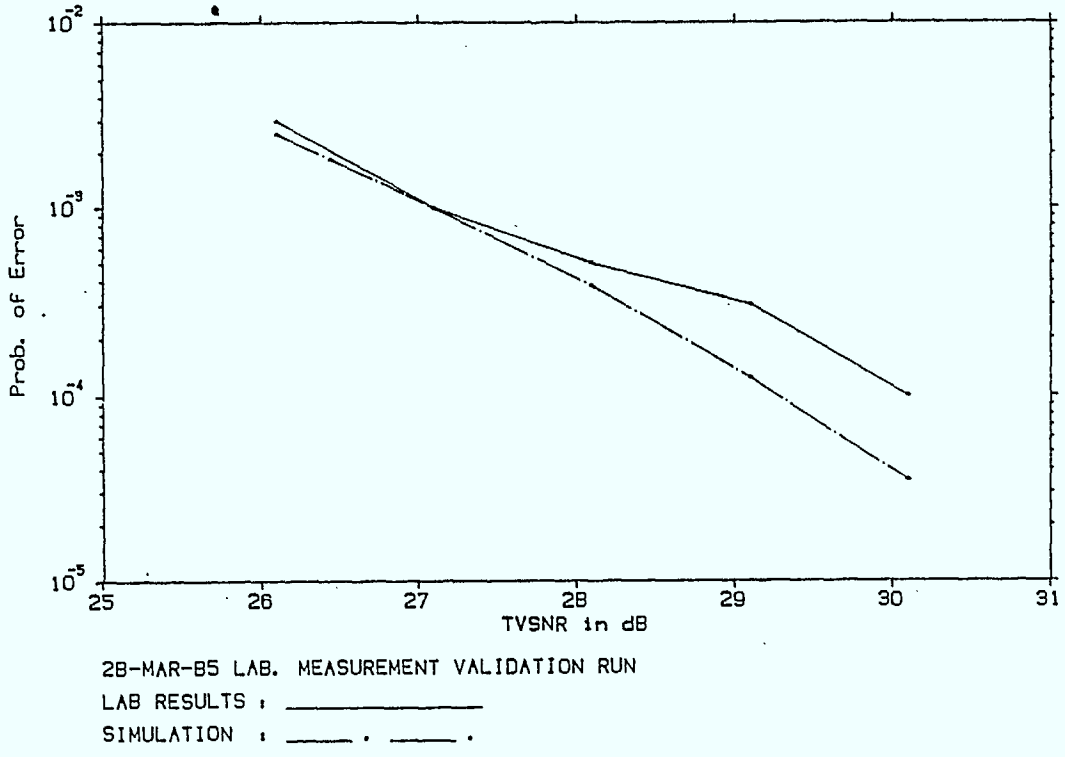


Figure 6.39 : Comparison of Performance with Lab Measurement Data vs. Quoted Performance Specifications

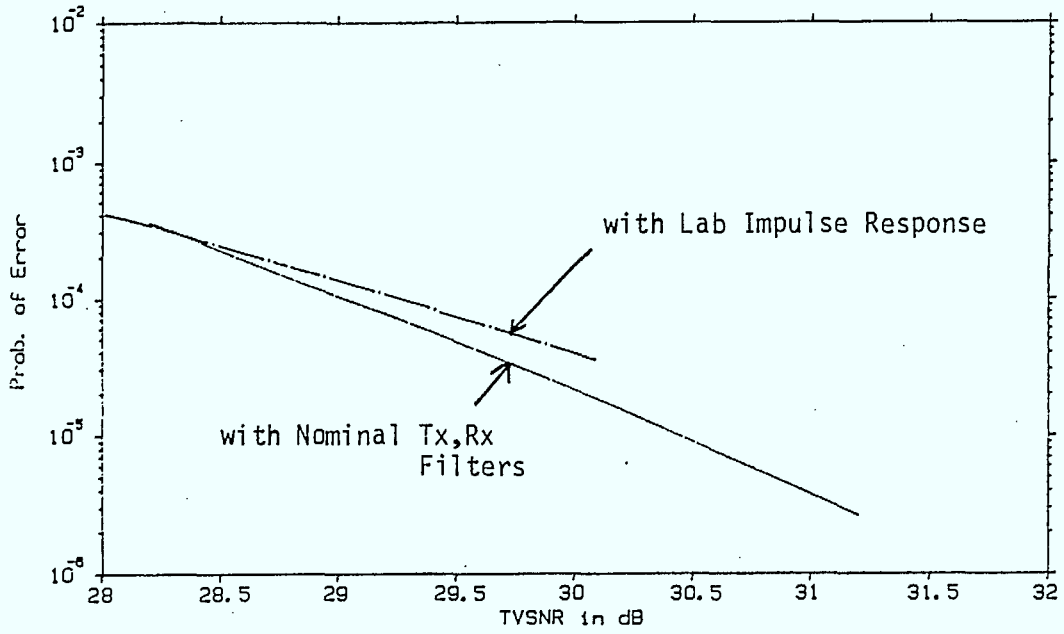


Figure 6.40 : Performance with Lab Measurement Complex Baseband Impulse Response Compared with Baseline System Performance in a Thermal Noise Environment.

the degradation relative to a baseline system (nominal zero phase transmitter and receiver filters is rather small. At a weighted video signal to noise ratio the degradation is approximately 0.4 dB. At lower video signal to noise ratios, the amount of degradation is relatively minor.

6.4 Validation Runs With Field Measurement Data

In this section, validation runs performed with field measurement data supplied by CRC are presented. The impulse response extracted in the field measurement program are inphase (real) impulse responses. The incorporation of these end to end channel impulse responses in the Telidon channel simulation program will be discussed in this section. Along with the end to end impulse response, the measured error sequence, an indication of the video signal to noise ratio, and the average bit error rate are provided for each measurement site. The measured error sequence files were used to assess the performance of alternative coding strategies in Section 3.

It was originally intended that some measurements would be performed with an averaging slicing level decoder (MA-MZ) to provide validation data for this contract. Difficulties getting the hardware version of this decoder to meet quoted performance specs for an ideal (back to back) laboratory environment meant that this field validation data could not be provided. Consequently, we must compare simulated bit error rate performance for the peak detector decoder for the prescribed channel and noise conditions to the measured performance in the field. Simulated performance will also be provided for the averaging slicing level decoder, and this should be considerably better than the measured performance with the peak detector decoder.

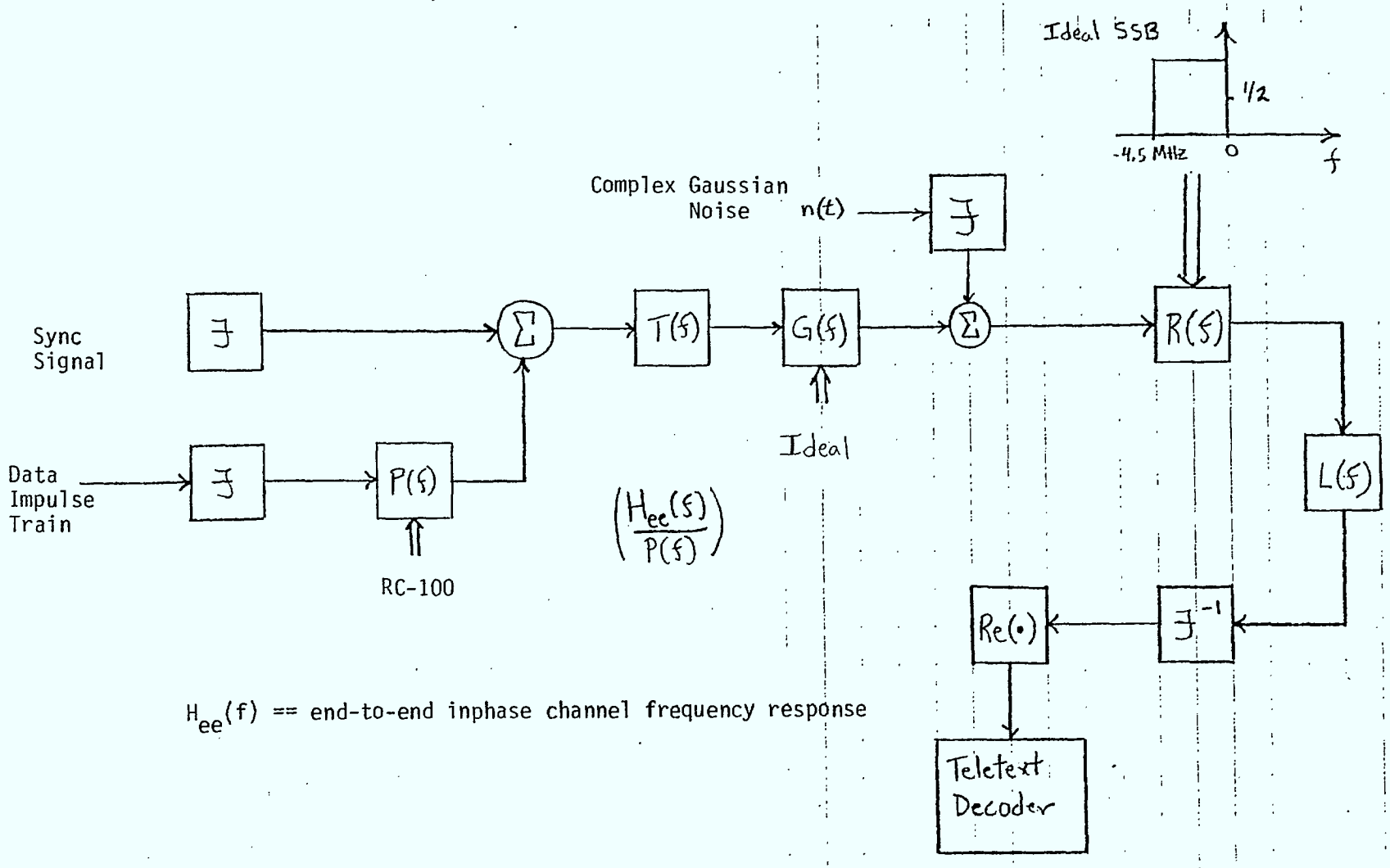
Although one could generate an error rate versus video signal to noise ratio curve for each measured channel, there is little incentive for doing so, given that measured performance is available at only one signal to noise ratio. What will be provided is an average error rate for both the peak detector and averaging slicing level decoders with a confidence interval for each decoder model. This will then be compared to the measured performance.

It must be emphasized that validation data for the averaging slicing level decoder would have been much better. In addition to providing better performance, this decoder is easier to simulate and not nearly so sensitive to model parameters. As indicated in [1], the performance of the peak detector decoder is extremely sensitive to peak detector reference voltages and time constants, especially in multipath environments. It is for this reason that the averaging slicing level decoder was recommended for the validation process, as one should expect a consistent trend with respect to simulated and measured performance with this decoder.

Before presenting the field measurement validation results, the procedure for incorporating measured end to end impulse responses in the simulation will be briefly described.

6.4.1 Incorporating End to End Inphase Impulse Responses in the Telidon Channel Simulation Program

The measured inphase impulse response incorporates the effects of the baseband pulse shaping filter (which has an RC-100 spectrum shape and no video bandlimiting filter), the transmit filter, the multipath channel as well as the post noise receiver filters. The complex baseband system model is illustrated in Figure 6.41. Because of the way



$\left(\frac{H_{ee}(f)}{P(f)}\right)$

$H_{ee}(f)$ == end-to-end inphase channel frequency response

Figure 6.41 : Complex Baseband System Model and Application of Filters for Field Measurement Simulation Runs.

that the teletext line has been constructed (inserting the teletext data signal on the television synchronization signals and line format), pulse shaping must be explicitly applied. Consequently, this pulse shaping response must be factored from the end-to-end channel frequency response. The resulting factored response could then be applied as the transmit filter and this would provide the desired end to end response. One also needs to apply a post noise receiver filter. Any ideal Nyquist slope receiver filter would suffice. Here, an ideal single sideband characteristic (see Figure 6.41) was used. This arrangement provides a white output noise spectrum, and in this case, the weighted video signal to noise ratio used in the Telidon channel simulation program is simply 6.1 dB higher than the measured unweighted video signal to noise ratio [20], [21].

The end to end impulse response has a sample period commensurate with a sampling frequency of twice the bit rate. The impulse responses provided were truncated to 41 samples (±10 bit periods). This impulse response was zero stuffed to 1024 samples, converted to the frequency domain, and had the pulse shape factored from the end to end frequency response by spectral division. The resulting frequency response is normalized to unity at zero frequency.

Data has been provided for four field measurement sites. The measured end to end inphase impulse responses and the average bit error rate performance for these channels are supplied in Tables 6.8, 6.9, 6.10, and 6.11. Also provided are the unweighted and weighted video signal to noise ratios that correspond to the measurement scenario. The amplitude and group delay responses for each of the measured end to end channels are supplied in Figures 6.42,

INPHASE COMPONENT

-.2236923E-01
-.8348424E-02
-.1769560E-02
-.5303008E-02
-.8028265E-02
-.4206716E-02
-.1632644E-01
-.2274064E-01
-.2305835E-01
-.2486162E-01
-.9361259E-02
-.1350015E-01
-.1326559E-01
0.5730013E-02
-.3024974E-01
-.2139551E-01
0.6076011E-01
0.2570291E-01
0.1974260
0.8884401
1.128193
0.5036963
-.4712345E-01
-.9450529E-01
-.4397043E-01
-.5218771E-01
-.4057406E-01
-.5194390E-01
-.4080389E-01
-.5991834E-02
-.1899499E-01
-.1080541E-01
0.3120703E-02
-.1200500E-01
-.1455988E-01
-.6837184E-02
-.1110943E-01
0.4813326E-02
0.2370642E-01
0.1967271E-01
0.7286116E-03

Table 6.8 : Inphase Component of IMP300 Impulse Response Data

INPHASE COMPONENT

-.1553730E-01
0.5864196E-02
-.1147265E-01
-.2453460E-01
0.8340698E-02
0.7055385E-02
0.5815180E-02
0.8692537E-02
-.4224109E-01
-.6779371E-01
-.1078998E-01
0.3959722E-01
0.4256962E-01
0.1940248E-02
-.2739207E-01
0.1615567E-02
-.8449979E-01
-.1733680
0.2833088
0.9319677
0.8345835
0.2516195
0.8299295E-02
0.2729506E-01
-.2377085E-01
-.6533907E-01
-.4755160E-01
-.2801615E-01
-.1837428E-01
-.3732338E-01
-.8259072E-02
0.3155557E-01
0.2399035E-01
0.2099344E-01
-.1188901E-02
-.1714252E-01
0.1794414E-01
0.2497968E-01
-.3765346E-01
-.6252232E-01
-.4659890E-01

Table 6.9 : Inphase Component of IMP301 Impulse Response Data

INPHASE COMPONENT

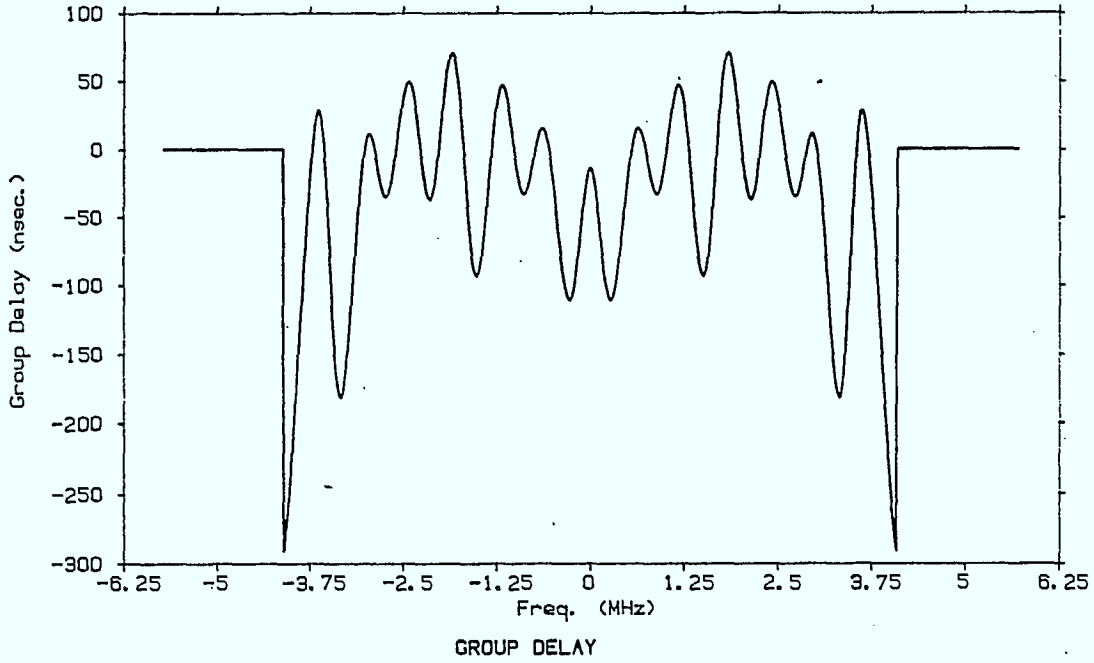
-.7037840E-02
-.7467135E-02
-.6279104E-02
-.5748043E-02
0.6739774E-02
0.1346745E-01
-.2397391E-01
-.3439511E-01
-.2294466E-01
-.2904676E-01
-.2504397E-01
-.2890456E-01
-.3155354E-01
0.1042423E-01
-.2256830E-01
-.5594078E-01
0.3345198E-01
0.2865441E-02
0.1763519E-01
0.6587334
1.184695
0.6963253
-.1707632E-01
-.1399613
-.2612834E-01
-.4382811E-01
-.4177679E-01
-.4457500E-01
-.3930119E-01
-.3896318E-02
0.2964652E-02
-.3635056E-02
0.2330829E-01
-.5754812E-02
-.1757185E-01
-.3209625E-01
-.5913913E-01
-.1806407E-01
0.4636684E-01
0.3461512E-01
0.8065485E-02

Table 6.10 : Inphase Component of IMP302 Impulse Response Data

INPHASE COMPONENT

-.1721820E-01
-.3391056E-01
-.2180789E-01
-.1840284E-02
.1000218E-02
.9253579E-02
-.1089394E-02
-.1823970E-01
-.5809505E-02
.5939834E-02
.4230615E-02
.4418497E-02
-.7113085E-03
.3260531E-02
.2787186E-01
-.3604354E-01
-.3151745E-01
-.2725667E-01
-.1357835
.2924577
1.053006
.8931832
.1233762
-.7954982E-02
.1597750
.6463420E-01
.5645690E-01
.3182684E-01
-.5955954E-01
.5715385E-02
.2587095E-01
-.3083639E-01
.1784462E-01
.3017441lE-01
-.5477904E-02
.2071556E-01
-.1051928E-01
-.3262407E-01
.6446829E-02
-.2250033E-02
.3050931E-04

Table 6.11 : Inphase Component of IMP362 Impulse Response Data



SNR_{TV} (unweighted) = 33 dB

SNR_{TV} (weighted) = 39.1 dB

Measured BER = 6.46×10^{-4}

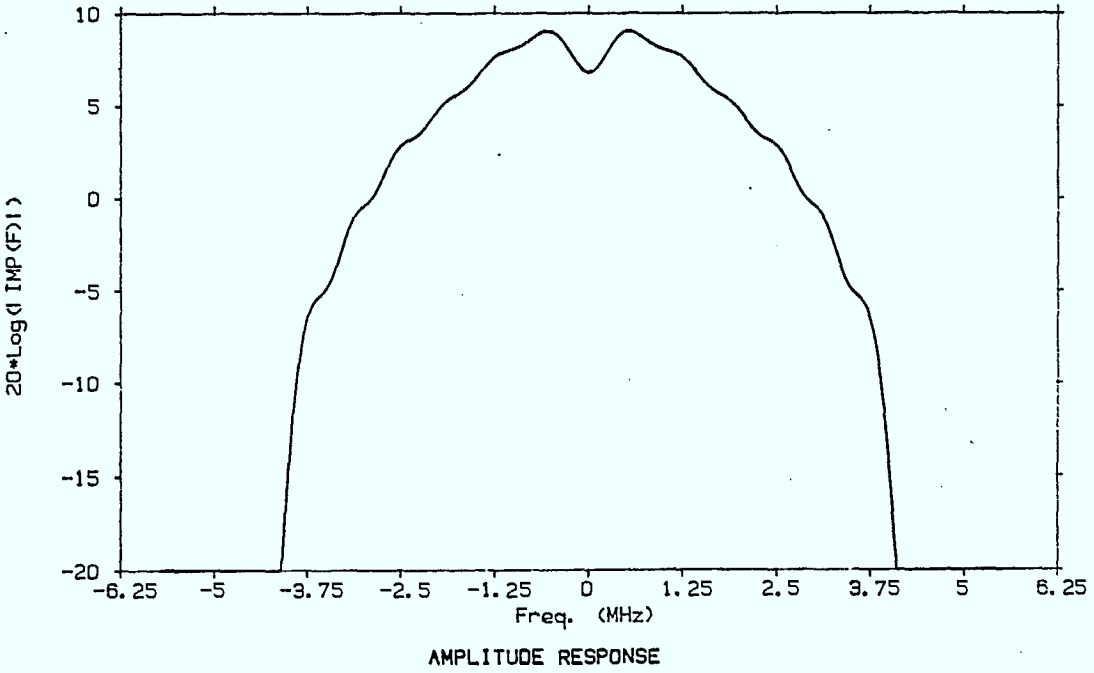
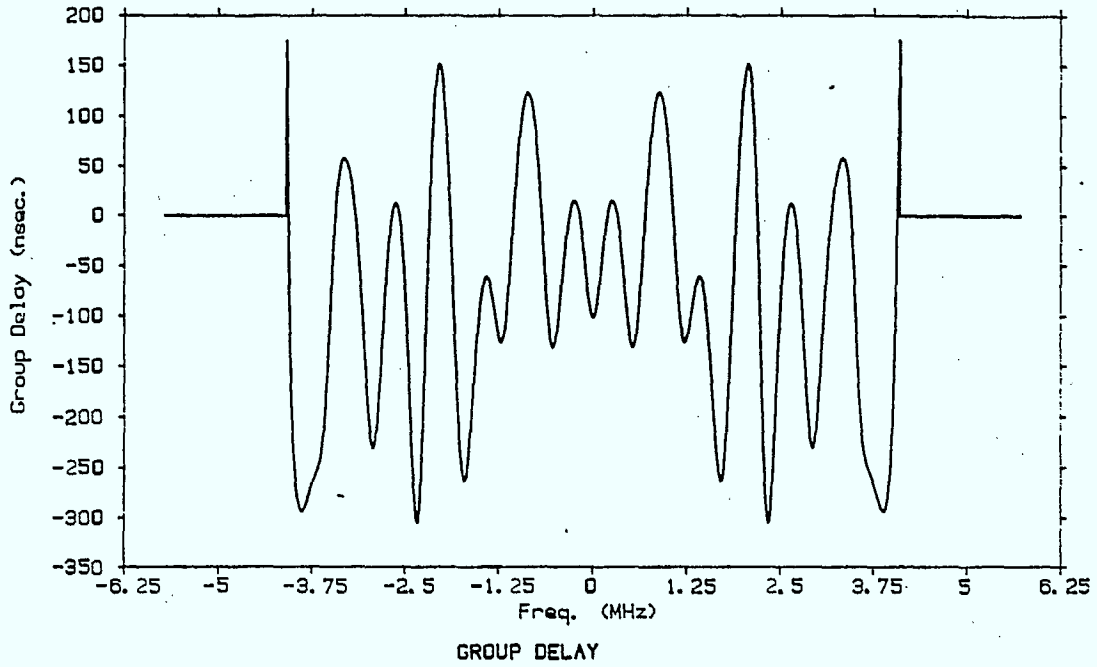


Figure 6.42 : Amplitude and Group Delay Responses of Channel Measurement IMP300.

6.43, 6.44, and 6.45. One surprising feature observed on all the amplitude responses (except that for the IMP362 channel) is a rather significant dip in the response around f=0 Hz (picture carrier). It is very improbable that multipath effects are introducing this phenomena, because one would expect ripples of this magnitude over the entire band if it were multipath induced. This notch could have a significant effect of system performance.

Before presenting the field measurement validation results, it is worth highlighting a few of the difficulties one has incorporating the measurement data from the current measurement program. The major difficulty stems from the fact that one would like to validate the simulation program for relatively high bit error rates (in the $1x10^{-3}$ to $1x10^{-4}$ range). This is necessary to keep the simulation processing requirements within reason. It is not realistic to consider simulation validation at error rates of $1x10^{-7}$ or video signal to noise ratios in the neighbourhood of 40 dB. Unfortunately, the region that is best suited for evaluating simulated performance corresponds to relatively poor video signal to noise ratios. Furthermore, because the sampling clock of the impulse response extraction system is not synchronized from burst to burst [11], one cannot reduce the corrupting effect of data autocorrelation sidelobes and noise on the measured impulse responses by impulse response averaging from burst to burst. Where the signal to noise ratios are commensurate with accurate impulse response estimation, system performance is really too good to be validated via simulation. Since one is really interested in examining performance near the threshold of acceptability (i.e. to evaluate the comparative performance of alternative coding strategies that provide a threshold extension), it is essential that the impulse response extraction system be able to average



SNR_{TV} (unweighted) = 26 dB

SNR_{TV} (weighted) = 32.1 dB

Measured BER = 1.72×10^{-3}

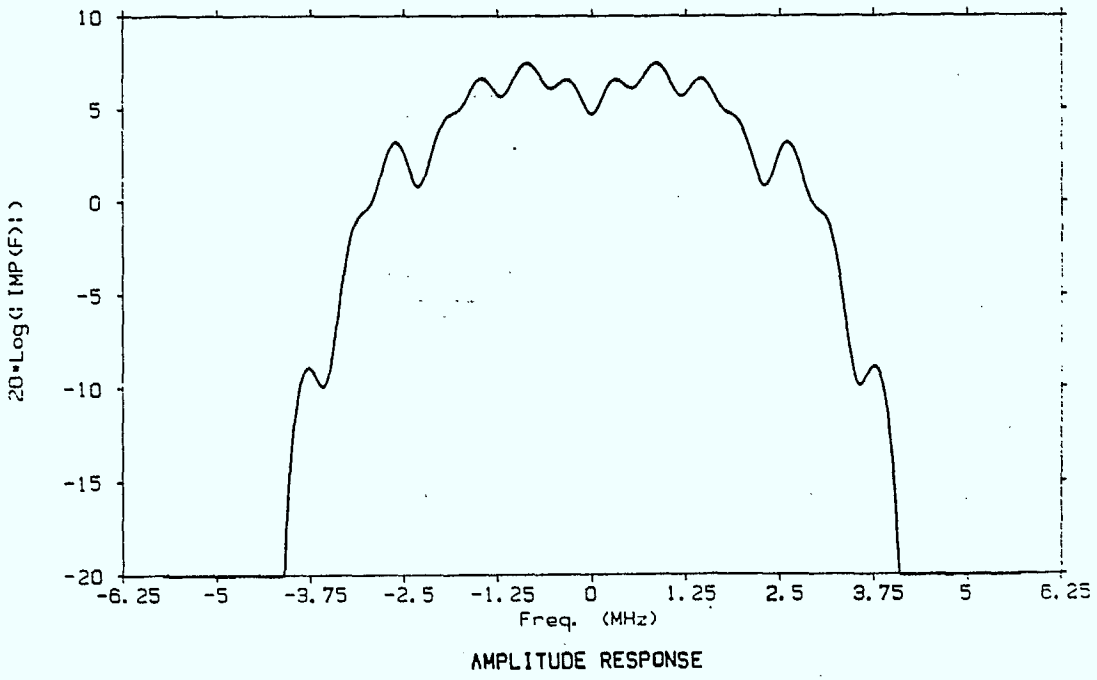
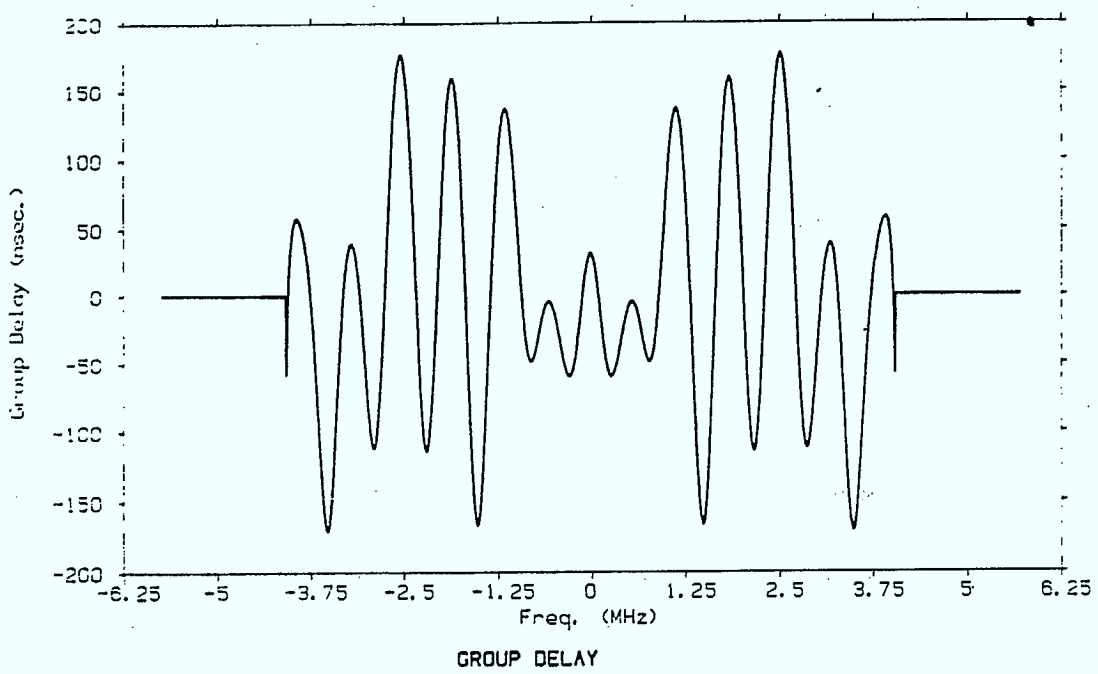


Figure 6.43 : Amplitude and Group Delay Responses of Channel Measurement IMP301



SNR_{TV} (unweighted) = 24 dB

SNR_{TV} (weighted) = 30.1 dB

Measured BER = 4.57×10^{-3}

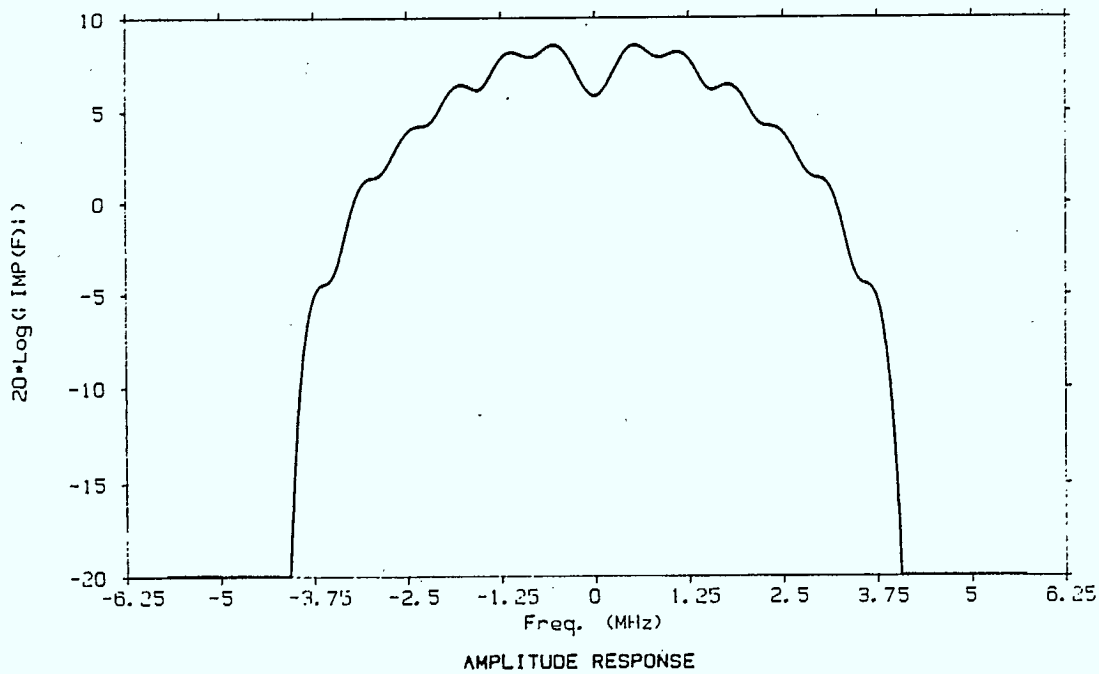
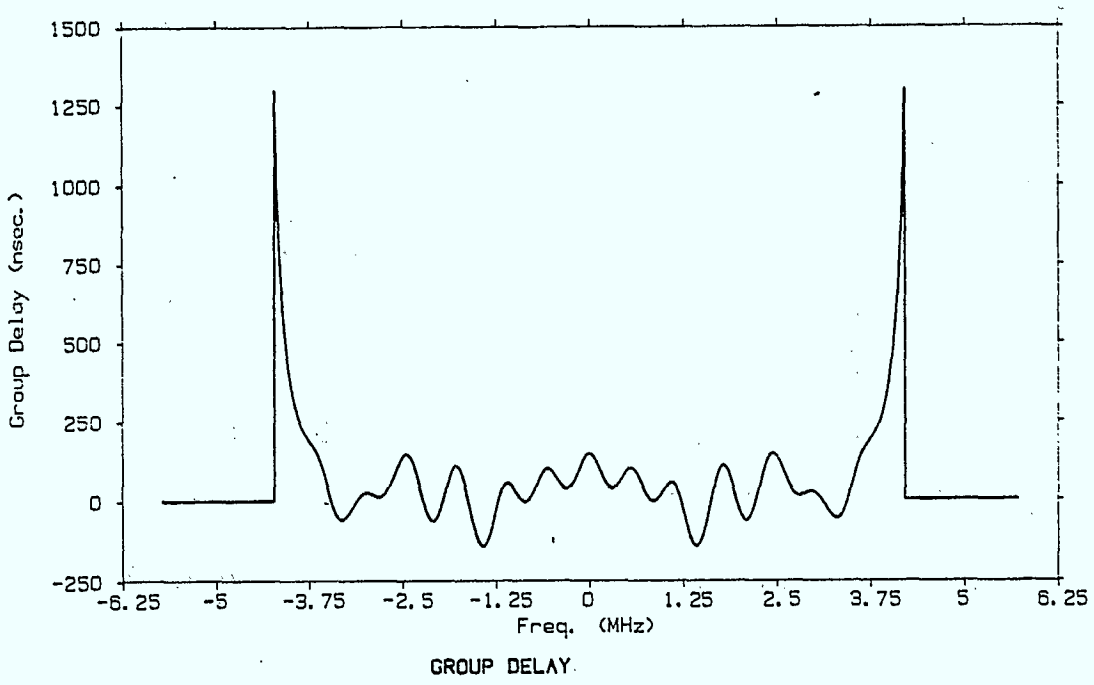


Figure 6.44 : Amplitude and Group Delay Responses of Channel Measurement IMP302



SNR_{TV} (unweighted) = 26 dB

SNR_{TV} (weighted) = 32.1 dB

Measured BER = 1.25×10^{-4}

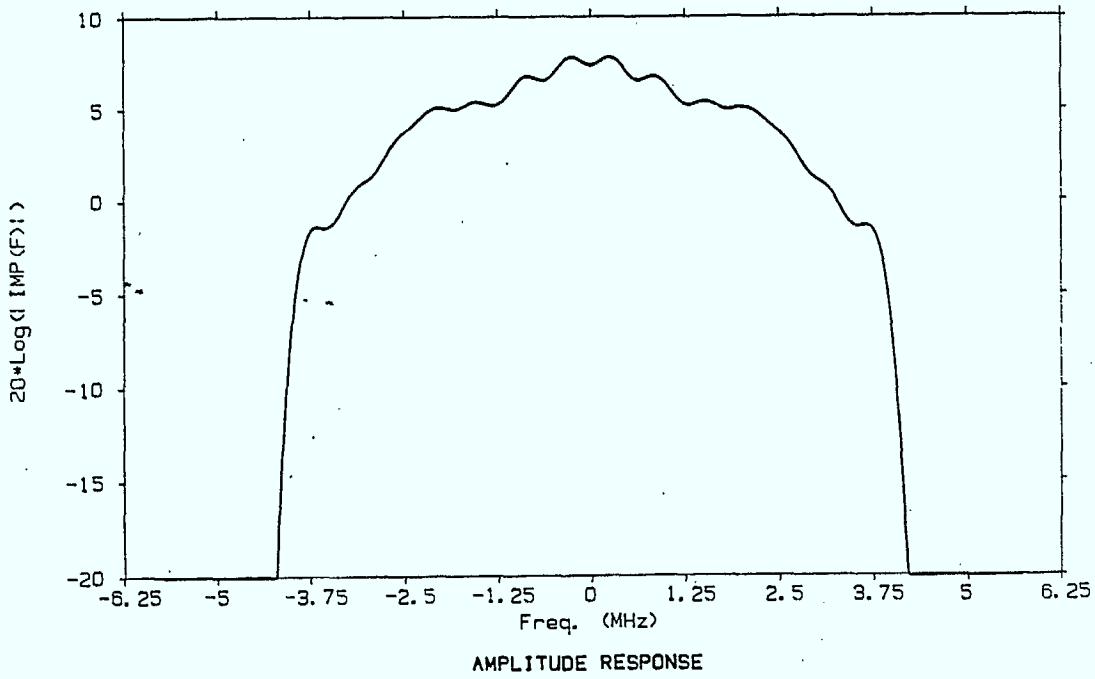


Figure 6.45 : Amplitude and Group Delay Responses of Channel Measurement IMP362

impulse responses from burst to burst to improve the quality of the channel impulse response in future measurement programs.

It was unfortunate that there had to be such a rush to supply measurement data for this contract. CRC personnel did not have adequate time to perform a thorough analysis of the measurement data before passing this information on to us. Consequently, the measurement data is rather preliminary at this stage. A case in point is the result for the channel IMP300, whose amplitude and group delay characteristics are presented in Figure 6.42. It is not sensible to expect that a channel of this quality would introduce the amount of degradation required to provide an error rate as high as 6.46×10^{-4} at a weighted video signal to noise ratio of 39.1 dB. Even for a relatively long simulation run (4000 bursts), no errors are observed. It is evident that some type of fault must have occurred during this measurement. Consequently this channel was not included in the field measurement validation runs discussed in the next section. Having a display on site to show the amplitude and group delay characteristics of the measured channel and expected error rate curves for typical scenarios would be useful for identifying situations such as this (where measured performance is far worse than one would expect). This prescreening procedure would be useful for ensuring that equipment problems do not colour the measurement results for a large number of sites.

6.4.2 Field Measurement Validation Results

The simulation results for the measured channel responses are provided in Table 6.12, for the PK-ZC and MA-MZ decoders. The performance of the MA-MZ decoder is considerably and consistently better than the simulated and

Channel	Weighted Video SNR	CRC Measured Performance (Peak Detector)	Simulated Performance (Peak Detector)	Simulated Performance (Averaging)
IMP301	32.1	1.72×10^{-3}	1.89×10^{-4}	9.47×10^{-5}
IMP302	30.1	4.57×10^{-3}	1.89×10^{-4}	$*7.58 \times 10^{-6}$
IMP362	32.1	1.25×10^{-4}	5.80×10^{-4}	$*7.58 \times 10^{-6}$

*only 2 errors observed in 4000 packets.

Table 6.12: Simulation results for measured channel responses and the peak detector and averaging slicing level decoders.

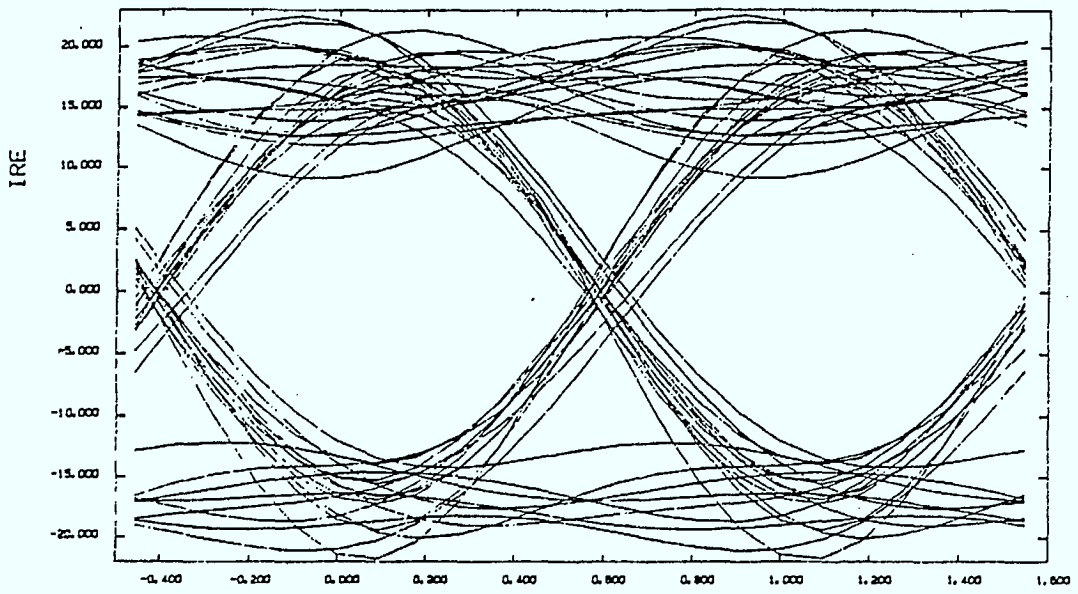
measured peak detector performance in all cases. The simulated performance of the averaging slicing level decoder is better by well over an order of magnitude than the measured peak detector performance. Comparing the simulated PK-ZC detector performance to measured performance with this decoder, one finds that simulated performance is better than measured performance by roughly an order or magnitude for the IMP301 and IMP302 channels. This sort of comparative performance is not really very good, as it exceeds what one would consider a reasonable implementation loss in development of a hardware decoder. Even more troublesome is the observation that the simulated performance with the IMP362 channel is roughly a factor of 4 worse than the measured performance. The lack of consistency is troublesome. The eye diagram for this channel is illustrated in Figure 6.46. Note that the eye closure is rather substantial. It would be interesting to compare this eye diagram to the eye diagram measured at this site to see if they are consistent.

A number of attempts have been made to justify the discrepancy. First of all, it was confirmed by A. Vincent that the accuracy of the measured video signal to noise ratio is roughly ±1 dB. Increasing the video signal to noise ratio on the IMP362 channel by 1 dB resulted in a 2.4×10^{-4} error rate, which is still high by a factor of 2. Attempting to smooth the group delay spike observed in this channel's response gave no significant performance improvement, and this can be attributed to significant attenuation of the overall channel response at the spike location. It was verified that it was the group delay is responsible for the distortion, as making the response zero phase (effectively zeroing the group delay) while maintaining the same amplitude response resulted in an order of magnitude improvement in bit error rate. An

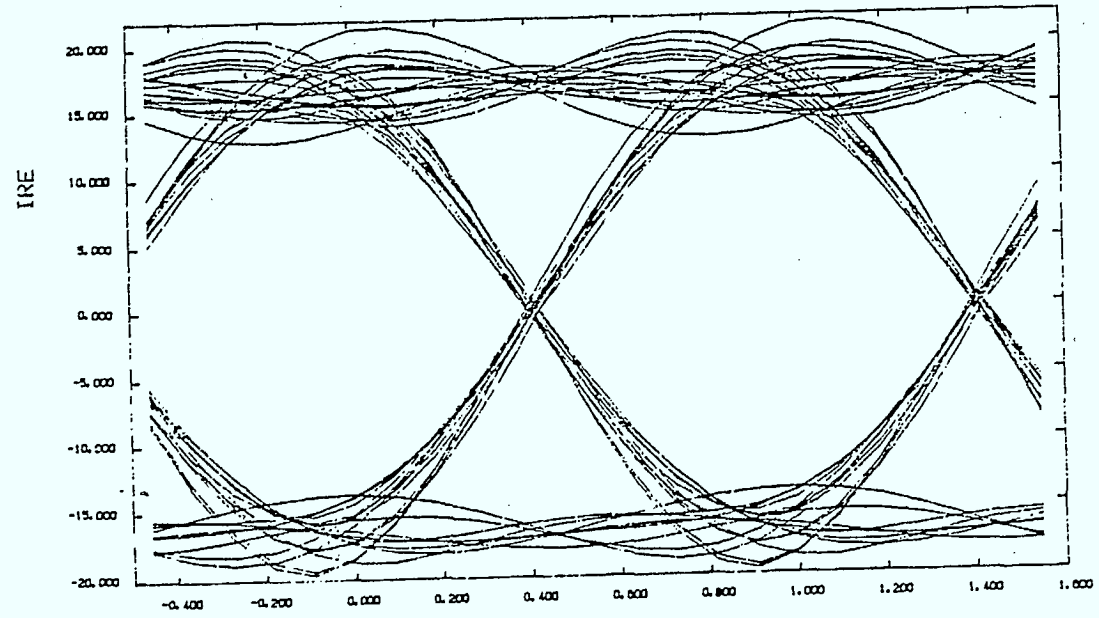
inspection of the resulting eye diagram presented in Figure 6.46 for this scenario justifies this magnitude of performance improvement.

The consequences of using noisy impulse response estimates on simulated system performance was also investigated. The system model considered was that of an ideal channel with nominal transmit and receive filters and a BS-14 pulse shape. It is rather difficult to predict the nature of the impulse response noise components, because they are dependent on both the characteristics of the noise process influencing the received data signal and the autocorrelation properties of the data signal used for impulse response. Since it is difficult to separate the effects of data and noise with least squares estimation, a simpler estimation technique that yields similar performance to least squares estimation (when the data autocorrelation sidelobes are not excessive) was analyzed. From the analysis performed, it was concluded that the effect that can be attributed to noise alone (assuming a perfectly white data signal) is not sufficient to introduce the degree of performance degradation between the simulated and measured performance obtained with the IMP362 channel.

A typical data autocorrelation function for a data sequence used for teletext impulse response extraction system is presented in Figure 6.47 [22]. Rather significant sidelobes, introduced by deterministic bytes, are evident. These are likely to have a rather significant effect on the quality of the extracted impulse rsponse. Another obvious point about the implementation of an impulse response extraction system that averages impulse responses from burst to burst, is that different random data sequences should be used for each burst. In this case, averaging will tend to reduce the corrupting influence of data



(a) with group delay



(b) with group delay removed

Figure 6.46 : Eye Diagrams for the IMP362 Channel

38

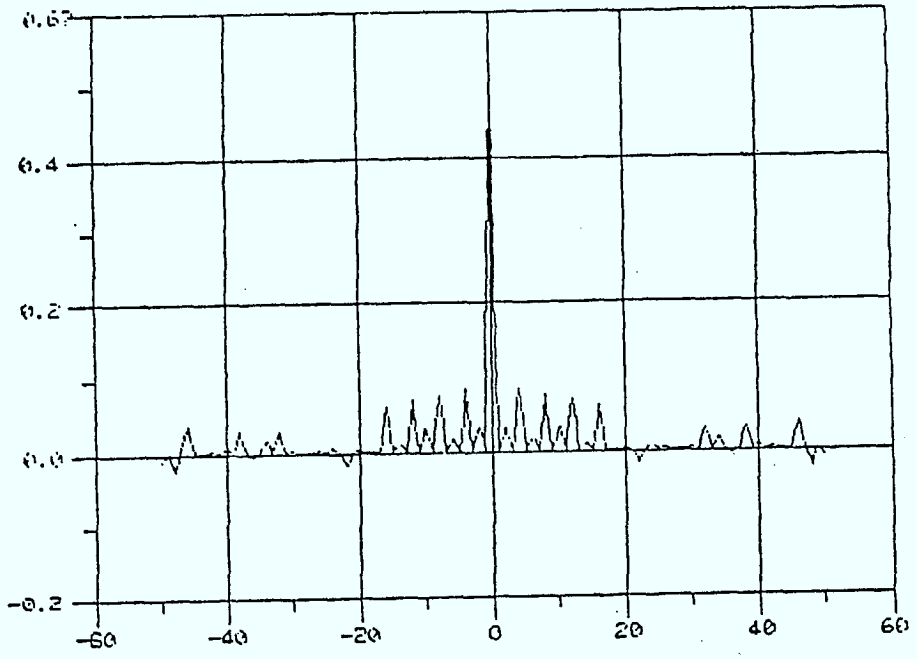


Figure 6.47 : Autocorrelation Function of the Data Sequence Used for Impulse Response Extraction in the CRC System (from [22]).

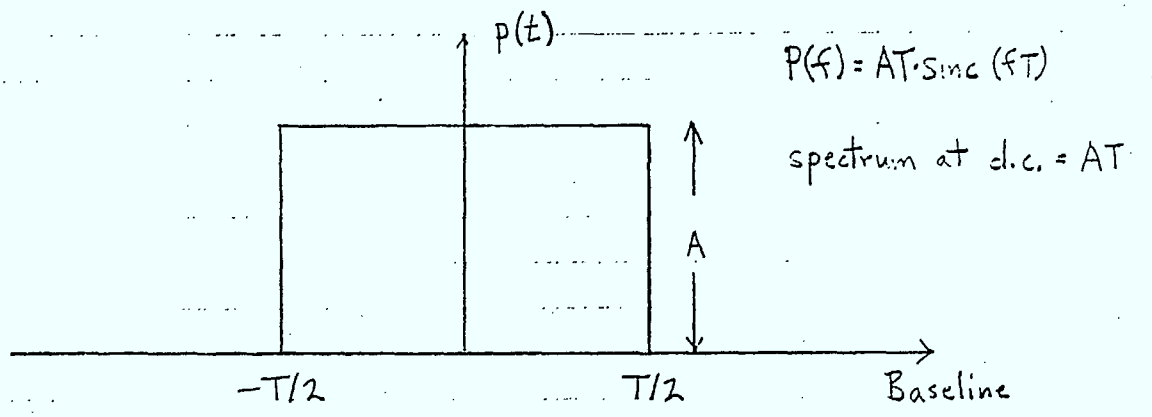
sequence autocorrelation sidelobes. Even for relative high signal to noise ratios, data autocorrelation properties can have a significant impact on impulse response extraction accuracy. In [23], which deals with identifying BPSK transmission systems based on their extracted impulse responses (fingerprints), it was found that a 3 second observation period was required to ensure that the data would not introduce noticeable corruption on extracted impulse responses for systems with 3 kHz bandwidth. Although this system is not identical to the teletext impulse response extraction system, it is similar enough to recommend an appropriate processing time. Using the empirical rule of 1 sec observation time for a 1 kHz bandwidth system, one finds that 250 μsec of data should be used for 4 MHz television channels. Assuming 264 bits per burst, the above empirical rule suggests using 5.4 bursts for impulse response extraction. Based on this, it is suggested that a minimum of 6 scans be used for impulse response extraction, independent of the signal to noise ratio.

The cause of the observed discrepancy with the IMP362 channel is not entirely clear. One possible explanation for the discrepancy is the pronounced dip in the amplitude responses at the picture carrier in the IMP301 and IMP302 channels, and the lack of such a dip in the IMP362 channel. The consequence of this is that the resulting energy per bit for a given video signal to noise ratio is greater for the IMP301 and IMP302 channels than it is for the IMP362 channel. Removing the dip in the IMP301 and IMP302 responses will likely result in a 2 dB or 3 dB reduction in digital signal to noise ratio for a specified video signal to noise ratio. Note that this would result in a considerable reduction in BER performance for these channels. This tends to suggest that observed relationship

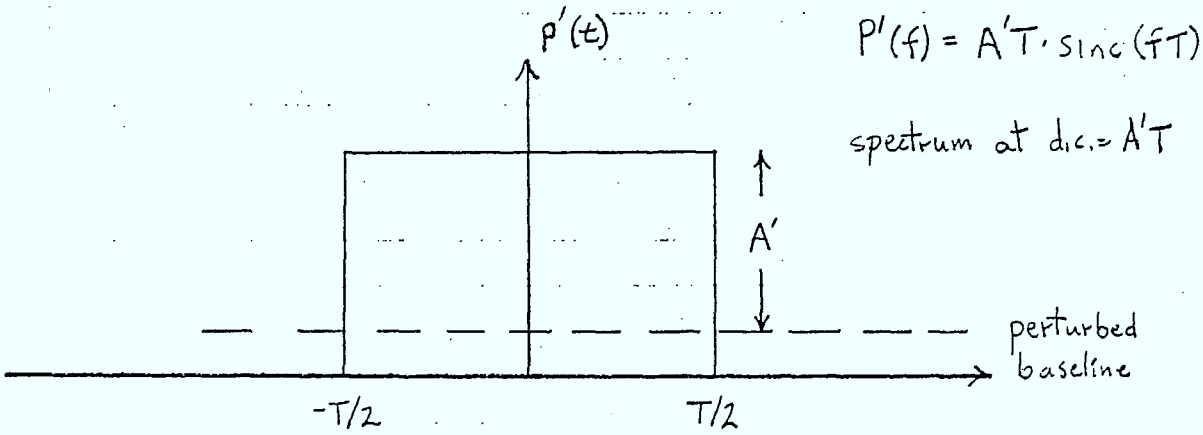
between the simulated and measured peak detector decoder performance may, in fact, be closer to the truth. From the outset [23], it was feared that the simulated peak detector decoder performance might not match measured performance exactly because of the sensitivity of this decoder to peak detector reference voltages and time constants.

The consequence of removing the dip at the picture carrier of the IMP302 channel was investigated for the peak detector decoder. This was done by simply flattening the response over the dip region. It was found that the error rate in this case was 3.87×10^{-3}, which is a factor of 20 worse than performance with the dip in the channel. It is felt that this is more indicative of what the actual system performance should be. This is relatively close to the measured error rate of 4.57×10^{-3}. Consequently, what was thought to be a discrepancy may in fact be closer to the truth. Time did not permit flattening the dip on the IMP301 response and examining performance.

The source of these pronounced dips at the picture carrier is somewhat of a mystery. It seems very unlikely that this is being introduced by the transmitter or receiver, or for that matter, multipath. It is interesting to note, however, that there is dc processing inherent in the impulse response extraction software, as the slicing level (dc-offset) must be estimated and removed from the data before an impulse response is extracted. It is quite likely that estimation errors in the dc-offset are the source of the observed problem. To demonstrate this, consider a simple rectangular pulse as illustrated in Figure 4.48(a), with an amplitude of A relative to its baseline. The frequency response in this case is given by AT sinc fT. The spectrum at zero frequency in this case is given by (AT).



(a) rectangular pulse with unperturbed baseline



(b) rectangular pulse with perturbed baseline

Figure 4.48 : Illustration Demonstrating the Importance of the Baseline on the DC Spectral Component.

However, if we move the baseline upward, and reduce the effective amplitude as indicated in Figure 4.48(b), the dc component of the resultant spectrum has been effectively reduced (only A'T in this case).

This brings up an interesting point. When estimating the slicing level in software, it might be wise to avoid using the deterministic bytes of the packet, as they may have more bits of a specific polarity than the other, and thus introduce an bias error in the slicing level. Even with random data generated by long PN sequence generators, it is possible to get data sequences with a dc-bias (more bits of a specific polarity than another). It is absolutely crucial that one remove the dc-bias introduced by the data sequence.

With knowledge of the actual data sequence used for impulse response extraction, it should be relatively easy to remove this bias given specific knowledge of the number of one's and zero's in the data stream.

It is recommended that this dc correction be incorporated in the present impulse response extraction software, and that the field measurement data be run through this improved impulse response extraction scheme to see if the severe dips in the response at the picture carrier are removed (or at least reduced).

It should be mentioned in closing that this perturbation in the slicing level may well be the explanation for poor impulse response estimates observed for unweighted video signal to noise ratios in the neighbourhood of 24 dB to 26 dB.

Strong noise could cause the slicing level to fluctuate from its appropriate value, and the resulting offset is

likely to introduce noticeable errors in the extracted impulse response. It must be emphasized that when we were analysing the effect of noise perturbed impulse responses on system performance, we were assuming ideal slicing level estimation. Although not proved here, it is believed that errors in estimating the dc-level will introduce significant error.

It is recommended that the effects of slicing level perturbation be investigated via simulation. It would be relatively easy to compare performance with a priori knowledge of the slicing level (ideal case) to that obtained with estimated slicing levels for video signal to noise ratios of interest. Furthermore, the effects of the dc-bias introduced by the data sequence itself could be quantified with the same simulation program.

The major accomplishments of the validation procedure are summarized in the next section.

6.5 Channel Validation: Summary and Conclusions

The Telidon channel simulation program has been validated against simulation programs developed by other investigators. A small database of potential transmitter and receiver responses has been generated. It has been shown that appreciable performance degradations can be attributed to typical transmitter and receiver characteristics. The simulated performance of the averaging slicing level decoder has been successfully validated against quoted manufacturer's performance, incorporating a measured complex baseband lab impulse response.

Validation of the Telidon channel simulation program against field measurement results has not been as successful. It is encouraging that simulated performance is relatively close to measured performance. However, it is relatively difficult to draw any concrete conclusions, as more questions have arisen than were answered. The preliminary nature of the measurement data of the current program are partially responsible for this. It is recommended that before any further validation runs are conducted that:

(1) the impulse response extraction system be improved to average impulse responses from burst to burst to provide higher quality channel measurements, and

(2) the averaging slicing level decoder be used in future measurement programs.

The first step in the validation process should be to compare simulated performance to measured performance for the averaging slicing level decoder in a controlled (back to back) laboratory experiment.

7.0 SUMMARY

This concludes the report on the enhancement and validation of the existing Telidon broadcast channel simulation system. This study lead to the following results.

The decision to transfer the software to the MCS computing facilities including a DEC VAX 11/750 and FPS5105 array processor was a wise one. In particular, the conversion of the software to the array processor reduced computation time by a factor of fifteen with respect to the original Honeywell CP-6 system. This reduction in computing time, greatly aided the testing and design of further enhancements to the system, and in validating the existing software.

The implementation of the proposed Telidon coding techniques was verified against the theoretical performance predictions. Verification was done on an ideal white noise channel, and theory and simulation were in very close agreement over the range tested (BER$>10^{-4}$). Evaluation of computer simulation and field measured error sequences showed the dependence of the coding gain on the receiver hardware, in particular the slicing level and bit timing algorithms. In fact with poor algorithms there was shown to be severe degradation of the coding performance due to packet-wise correlation of errors.

The simulation program was enhanced by the introduction of an approximation to an improved hardware receiver. This decoder used a modified averaging technique for estimating the slicing level and a modified zero-crossing technique for symbol synchronization. The modified averaging technique was found to close to ideal on a white noise

channel. The modified zero-crossing technieque was an improvement over an existing technique, but leaves room for further improvement.

Also introduced was an adaptive slicing technique and a correlator based symbol synchronization technique. The adaptive slicer was found to only provide an advantage on those multipath channels which had only wide multipath spreads. On other channels, the adaptive slicer did not perform much better than the modified averaging technique. On the other hand, the correlator based bit timing recovery technique was shown to be close to ideal. Not only did it provide an improved BER performance, but it also reduced the correlation between errors and thus enhanced coding performance as well.

There were two aspects to the task of validation of the Telidon channel simulation program. In the first case, the simulated performance has been successfully validated against quoted manufacturer's performance, incorporating a laboratory measured complex baseband impulse response. In the second case, validation against full measurements was not as good but several questions and recommendations have been made about the measurement procedure.

In summary, the enhancements to the Telidon simulation system have demonstrated a slicing technique and a bit timing recovery technique which are nearly ideal in a white noise channel. However in severe multipath channels, neither these improved techniques nor the data encoding provide much relief to the deterioration in performance encountered. In these cases, the only resort would be a type of channel equalization as suggested in [1].

REFERENCES

[1] K.W. Moreland et. al., Telidon System Study Final Report, Miller Communications Systems Ltd., DSS File No: 21ST.36100-2-4380, MCS File No: 8342, July 1984.

[2] K.W. Moreland et. al., Telidon System Study 2nd Interim Report, Miller Communications Systems Ltd. DSS File No: 21ST.36100-2-4380, MCS File No: 8342, June 1983.

[3] A.V. Oppenheim and R.W. Schafer, Digital Signal Processing, Prentice Hall, 1975.

[4] A.B. Carlson, Communication Systems, 2nd ed., McGraw-Hill, 1975.

[5] W.C. Lindsay and M.K. Simon, Telecommunications Systems Engineering, Prentice Hall, 1973.

[6] Broadcast Specifications BS-14, Dept. of Communications, Govt. of Canada, Ottawa, Issue 1, June 1981.

[7] M. Sablatash and J. Storey, Determination of Throughputs, Efficiencies, and Optimal Block Lengths for an Error-Correcting Scheme for the Canadian Broadcast Telidon system, Can. Elec. Eng. J, Vol. 5, No. 4, 1979.

[8] B. Mortimer, A Study of the Use of Error Correcting Codes in Broadcast Telidon, DSS Contract No. OSU801-00095, February 1982.

B. Mortimer, A Description of the Carleton Code, DSS Contract No: OSU801-00095, September 1981.

[9] B. Mortimer, M. Moore, More Powerful Error-Correction Scheme for the Broadcast Telidon System, DSS Contract No: OSU82-00164, March 1983.

B. Mortimer, Two Byte Data Block and Bundle Codes for the Broadcast Telidon System, DSS Contract No: OSU82-00164, November 1982.

[10] B. Mortimer et. al., A Study of theUser of Error-Correcting Codes in Canadian Broadcast Telidon: One Byte Codes, DSS Contract No: OSU81-00095, August 1981.

[11] "Minutes of the Telidon Channel Measurement Meeting", February 6, 1985.

[12] K. W. Moreland, "Complex Impulse Response Measurements", Miller Communications Systems Ltd. Technical Memorandum, March 12, 1985.

[13] M. Pittarelli, "Teletext Computer Simulations: DRAFT", Department of Communications, Ottawa, April 1983.

[14] P. Fockens and C. Eiler, "Anaysis of System Models: Chapter III of EIA Report", Zenith Radio Corporation, November 1983.

[15] "Radio Standards Specification: RSS-154, Issue 2", Department of Communications' September 1, 1980.

[16] T. M. Gluyas, "Televison Demodulator Standards", IEEE Trans. on Consumer Elect., Vol. CE-23, No. 3, August 1977.

[17] K. W. Moreland et.al. "Telidon System Study: 3rd Interim Report", Miller Communications Systems Ltd. Report, DSS File No: 21ST.36100-2.4380, MCS File No. 8342, September 1983.

[18] K. W. Moreland et.al. "Telidon System Study: 1st Interim Report", Miller Communications Systems Ltd. Report, DSS File No: 21ST.36100-2.4380, MCS File No. 8342, March 1983.

[19] Private communication with A. Vincent.

[20] "C.R.I.R. XIIIth Plenary Assembly, Volume XII-Geneva 1974", International Telecommunication Union, Geneva, 1975, pg. 37.

[21] T. Fujio, "A Universal Weighted Power Function of Television Noise And Its Application To High-Definition TV System Design", IEEE Trans. on Broadcasting, Vol. BC-26, No. 2, June 1980, pg. 43.

[22] A. Vincent, "Measurement of The Impulse Response of Televison Channels for Broadcast Telidon", Communications Research Center, Ottawa, Ontario, January 1984.

[23] M. Moher, "Signal Identification Report: BPSK HF Radio Identification", MCS File No. 8323, March 26, 1984.

[24] R. K. Tiedemann and K. W. Moreland, "Proposal to Undertake Further Study and Simulation of the Telidon Transmission Study: Draft", MCS No. 85039, October 18, 1984.

Appendix A

Analysis of C Code and Bundle Code at High BER

Prepared by: Dr. B. Mortimer
Department of Mathematics and Statistics
Carleton University
Ottawa, Canada

A.1 PERFORMANCE OF CODE C WITH ERASURE DECODING OF INDEPENDENT ERRORS

Code C corrects any pattern of errors confined to a single byte and any pattern which corrupts two bytes making a parity failure in each. So the least weight error event giving a decoding failure or error is a triple error. We count the number of ways different error events can occur. We denote for example by 2-2 the event of four errors with two in each of two bytes. Some patterns are falsely corrected as double errors in a weight 6 codeword. Some are rejected as uncorrectable but detectable errors.

The most common error patterns of this type are shown in Table A.1 (A_6 = 8600, see [9]).

For the case 2-2-1 of 5 errors, the partition into decoding failures and errors was arrived at by an averaging procedure. There are approximately

$$28.27\binom{8}{2}\binom{6}{2} \cdot \frac{26}{128}$$

weight 8 codewords of shape 2-2-4. Each gives 4 patterns 2-2-1 which will be corrected by adding 3 more errors. Each weight 6 codeword (which must have shape 2-2-2) yields 6 patterns 2-2-1 which are corrected by adding one more error.

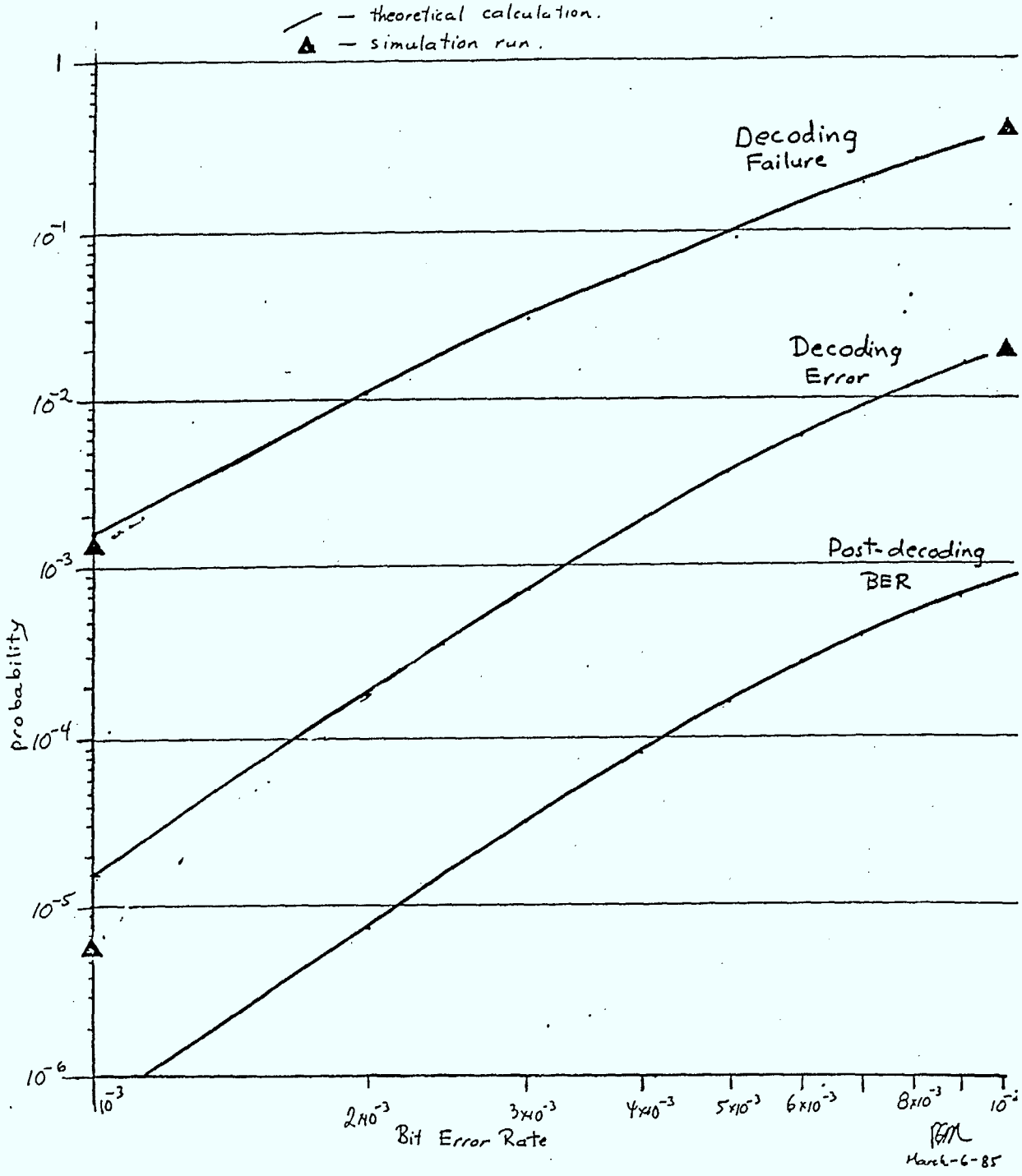
Using these approximations the performance of the C Code in terms of the fraction of packets rejected (Decoding Fault) and the fraction of packets with undetected errors (Decoding Error) is illustrated in Figure A.1 for BER over

Table A.1 - Occurences of Common Error Patterns

No. of Errors	Byte Pattern	Result	No. of Patterns		Probability	No. of Output Bit Errors
3	1-1-1	DF	$\binom{28}{3}8^3$	1,677,312	$p^3(1-p)^{221}$	
	2-1	DF	$28.27\ \binom{8}{2}8$	169,344		
	3	CD				
4	1-1-1-1	DF	$\binom{28}{4}8^4$	83,865,600	$p^4(1-p)^{220}$	
	2-1-1	DE	$\binom{28}{4}26\binom{8}{2}8^2$	17,611,776		~10
	2-2	DF	$\binom{28}{2}\binom{8}{2}^2-3A_6$	270,800		
		DE	$3A_6$	25,800		6
	3-1	CD				
	4	CD				
5	1-1-1-1-1	DF	$\binom{28}{5}8^5$	3.2204 E9	$p^5(1-p)^{219}$	
	2-1-1-1	DF	$\binom{28}{3}25\binom{8}{2}8^3$	1.1741 E9		
	2-2-1	DF		61,331,631		
		DE		309,585		~8
	3-1-1	DF	$\binom{28}{2}26\binom{8}{3}8^2$	3.52235 E7		
	3-2	DF	$28.27.\binom{8}{3}\binom{8}{2}$	1,185,408		
	4-1	DF	$28.27.\binom{8}{4}8$	423,360		
	5	CD				

CD correct decoding = packet correct after decoding.
DF decoding failure = packet rejected after decoding.
DE decoding error = packet contains junk after decoding.

Figure A.1 Performance of Code C with Erasure Decoding and Independent Errors



the range from 10^{-3} to 10^{-2}. The actual numerical results and a comparison to Carleton simulation results are shown in Table A.2.

BER	Probability of Correct Decoding	Decoding Failure	Decoding Error	Bit Error after Decoding
.1E-03	.998434393	1.55145339E-03	1.41531779E-05	6.31466387E-07
2E-03	.989367686	.0104506395	1.21674205E-04	8.10565313E-06
3E-03	.969510766	.0297515209	7.37703231E-04	3.29135934E-05
4E-03	.938560289	.0595700453	1.86966573E-03	8.34172091E-05
5E-03	.89795819	.0983821979	3.65961178E-03	1.63217059E-04
6E-03	.850058202	.143859125	6.08267315E-03	2.71383285E-04
7E-03	.797571393	.193897947	9.03066046E-03	4.02908647E-04
8E-03	.743205746	.244451041	.0123432126	5.50698307E-04
9E-03	.689441431	.294721232	.015837337	7.06587812E-04
.01	.63840241	.342266393	.019331197	8.62464405E-04

Errors correctable

Errors detected but uncorrectable

Errors either undetected or falsely corrected

BER after decoding resulting from decoding errors

1.004e-3	.9985549	1.4392e-3	5.9165e-6
0.9995e-2	.6184272	.3621675	.0194053

Table A.2 Performance of Code C with Erasure Decoding & Independent Errors

Above: theoretical calculation

Below: results of two simulation runs

A.2 PACKET REJECTION RATE FOR THE BUNDLE CODE

We are calculating the probability that a packet is rejected when the Single-Bundle code is decoded with a packet replaced with an erasure, i.e. 28 bytes all with parity failures, whenever

(i) a prefix code detects an error it can't correct.

(ii) a horizontal Code C codeword detects an error it can't correct.

We assume that a prefix code, i.e. 5 Hamming [8,4] codewords, is either correctly decoded or the error is detected as uncorrectable (option (i) above). This is valid if essentially any change in the information in the prefix such as would occur if there was a decoding error would put the packet out of context or make it incomprehensible. Thus such a change would be recognized by the decoder and the prefix code would fail for reasons beyond the actual structures of the code.

The constituent code used in the bundle is Carleton C2 or to us Code C. This code is decoded in such a way that in the case of two byte parity failures a correction always occurs; so no decoding failures can happen in such a case. For example, if for reason (i) or (ii) a packet is replaced with erasures and if the other bytes all have the correct parity, then whether they are right or wrong, "corrections" will be made in the 2 erased bytes of each vertical codeword.

The possibilities for rejecting a bundle are as follows:

I. 0 prefix failures & > 2 horizontal failures

II. 0 prefix failures & 0 horizontal failures and at least one decoding error in a horizontal codeword, which is detected by the vertical codewords but is uncorrectable. (In fact such decoding errors are generally corrected vertically.)

III. 1 prefix failure & > 1 horizontal code failures (in the other h-1 packets).

IV. > 2 prefix failures.

The probabilities are as follows:

probability Prefix correctly decoded = PreCor = $(q^8+8p.q^7)^5$

probability Code C correctly decoded = Code CCD (from section A.1)

probability Code C decoding failure = Code CDF (from section A.1)

$$\text{Prob(I)} = (\text{PreCor})^h[1-(1-\text{Code CDF})^h - h.\text{Code CDF}(1-\text{Code CDF})^{h-1}]$$ "not 0 and not 1 hor. DF's"

$$\text{Prob(II)} = (\text{PreCor})^h[(1-\text{Code CDF})^h-(\text{Code CCD})^h]$$

"no decoding failure - exclude case of all packets correct"

$$\text{Prob(III)} = h(1-\text{PreCor})(\text{PreCor})^{h-1}[1-(1-\text{Code CDF})^{h-1}]$$

"not 0 DF's in other packets"

$$\text{Prob(IV)} = 1-(\text{PreCor})^{h}-h(1-\text{PreCor})(\text{PreCor})^{h-1}.$$

"not 0 and not 1 prefix failure"

The rejection rate for packets (or bundles) is the sum of these four and is shown in Table A.3. Of course Prob(II) is an approximation - an upper bound.

Table A.3 - Bundle Rejection Rate Performance at High BER

BER	Rejection Rate	I	II	III	IV
10^{-2}	.98	8.1e-1	1.5e-3	1.6e-1	1.5e-2
$9x10^{-3}$	.96	8.1e-1	3.1e-3	1.3e-1	9.9e-3
$8x10^{-3}$	.91	7.9e-1	6.0e-3	1.1e-1	6.4e-3
$7x10^{-3}$	.81	7.1e-1	1.0e-2	8.0e-2	3.8e-3
$6x10^{-3}$	.65	5.8e-1	1.4e-2	5.6e-2	2.1e-3
$5x10^{-3}$	.44	3.9e-1	1.6e-2	3.4e-2	1.0e-3
$4x10^{-3}$	2.3e-1	2.0e-1	1.3e-2	1.6e-2	4.3e-4
$3x10^{-3}$	7.6e-2	6.2e-2	7.4e-3	5.6e-3	1.4e-4
$2x10^{-3}$	1.2e-2	9.1e-3	2.3e-3	9.9e-4	2.8e-5
10^{-3}	4.5e-4	2.2e-4	2.0e-4	3.9e-5	1.8e-6

Validation of the Error Correction Software

For the purpose of determining the validity of the Error Correcting Software written for the Telidon Simmulation project I have:

(i) read through the Fortran source code,
(ii) read through the pseudo-code decoder for Code C,
(iii) calculated the theoretical performance levels for Code C and the Bundle Code at high Bit Error Rates and compared the results with those produced by the Error Correction Software.

I am satisfied that the programmers completely understand the workings of the codes and the decoding algorithms. I am also satisfied that the software performs in the way that it is described as acting.

Brian Mortimer

Brian Mortimer
NSERC University Research Fellow
Department of Mathematics and Statistics
Car;eton University,
Ottawa, Ontario, Canada.

APPENDIX B

PSEUDOCODE DESCRIPTION OF DECODING ALGORITHM

Decoding Algorithms for the Code C and Bundle codes
(Version used in the program CODING)

CODE C

Look-up tables for calculations.

Look-up tables are needed to perform calculation in the finite field - GF(128).

The generator polynomial for the field is

$$(x + 1)(x^7 + x^3 + x^0) = 0$$

$$x^8 + x^7 + x^4 + x^3 + x + 1 = 0$$

The first eight elements, alpha^i s, are

alpha^1 0 0 0 0 0 0 0 1
alpha^2 0 0 0 0 0 0 1 0
alpha^3 0 0 0 0 0 1 0 0
alpha^4 0 0 0 0 1 0 0 0
alpha^5 0 0 0 1 0 0 0 0
alpha^6 0 0 1 0 0 0 0 0
alpha^7 0 1 0 0 0 0 0 0
alpha^8 1 0 0 0 0 0 0 0

The elements of the field that are generated by the above polynomial all have odd weight. Each element in the field has a corresponding even weight representation as we require that

$$x^7 + x^3 + x^0 = 0.$$

To convert from on representation to the other perform the following:

even = odd (+) 10001001

(2)

The complete tables used are provided at the end of this Appendix. A simple description of their functions is provided here:

Tables - CL(x) 0 <= x <= 255

Functions of CL(x)

a) If x is odd weight (ie. has an odd number of ones in it) then CL(x) < 128.
If x is even weight then CL(x) >= 128.
Thus, the table CL(x) may be used to determine the weight of the argument. (In fact this table is used to determine the location of the parity errors

b) Look-up table to determine the log (with respect to the finite field) of the argument. Both even- and odd-weight representations of the argument may be used.
If an even-weight representation is employed then CL(x)=CL(x)-128.

Example: if $x = \text{alpha}^{i}$

where alpha^{i} is an element of GF(128)

then CL(x) = i.

- CXO(x) 0 <= x <= 127

Function of CXO(x)

a) Performs the exponential operation and yields an odd-weight element.

Example: for i given above

$CXO(i) = \text{alpha}^{i}$

where alpha^{i} is in the odd representation for

the field.

- CXE(x) 0 <= x <= 127

Function of CXE(x)

a) Performs the exponential operation and yields an even-weight element.

1 - Determine the location of the parity errors

Method - Store the location of all of the errors that are of even weight.

Pseudo code -

```
DO i = 1 , number-of-error-bytes
   IF weight of error-byte(i) is odd THEN
      location-parity-error(i) = location-error(i)
   END IF
END DO
```

2 - Determine the Product syndrome

Method - 'Exclusive Or' all of the error bytes together.

Pseudo code -

```
product-syndrome = 0
DO i = 1 , number-of-error-bytes
   product-syndrome=product-syndrome (+) error-byte(i
END DO
```

3 - Determine the Carleton syndrome

Method - Calculate:

$$\left[\ \text{MOD2 SUM}\ \right]_{i=1}^{\text{number of errors}} \left(\text{error}_i * \text{alpha}^{8*i} \right)$$

where error_i = error in byte i (denoted: location-error(i))

Pseudo code -

```
Carleton-syndrome = 0
DO i = 1 , number-of-error-bytes
    temp = CL(error-byte(i)) mod 128
    temp = CXO([Carleton-syndrome + 8*location-error(i)] mod 12
    Carleton-syndrome =Carleton-syndrome (+) temp
  END DO
```

4 - Decode the packet

```
Method - Solve the equations specified by the
           product and Carleton syndromes. The
           byte-parity bits aid in this process.
           The look-up tables describes in section 1
           are used for the calculations.

   Let W(x) denote the weight of x (number of nonzero bits)

       Example:   W(01101110) = 5

a) IF W(Carleton-syndrome)=W(product-syndrome)=(the
   number of parity-errors) = 0

   THEN
      The decoded packet is the same as the input packet
      GOTO  ACCEPT-THE-PACKET

b) IF    W(Carleton-syndrome) .NE. 0
       & W(Product-syndrome)  .NE. 0
       & The number of parity errors = 0

   THEN
      There is the possiblity of a single byte error
      with even weight.  Calculate the following:

       X = CL(Carleton-syndrome) mod 128
           -  CL(Product-syndrome) mod 128

       DO k = 0 , 2
         X = X + k*127
         IF   X is a multiple of 8
            & 6 <= X <= 33
         THEN
            Correct the byte indicated by X
            by exclusive or-ing it to the
            product-syndrome
            GO TO ACCEPT-THE-PACKET
         END IF
       END DO

       The calculation results in an error
       out of range of the codeword.

       GO TO  REJECT-THE-PACKET

 c)  IF  W(Carleton-syndrome) .NE. 0
       & W(product-syndrome)  .NE. 0
       & The number of parity errors = 1

     THEN
```

```
There is a possiblity that a single byte
is in error and that the byte in error is
that indicated by the parity error.
Calculate the Carleton syndrome
assuming that the only error is given by
the product syndrome, and that the error
byte is that flagged by the parity error.

temp = ( CL(product-syndrome)
         + 8*location-parity-error(1)) mod 127

test-Carleton-syndrome = CX0(temp)
IF  test-Carleton-syndrome = Carleton-syndrome
THEN
   Correct the byte indicated by the
   parity error by exclusive or-ing it
   with the product-syndrome
   GO TO  ACCEPT-THE-PACKET
END IF
The expected and actual syndromes did not
match

GO TO REJECT-THE-PACKET

d)   IF  (W(Carleton-syndrome) .NE. 0).OR.(W(product-syndrome) .NE. (
    &    The number of parity errors = 2

 THEN
    It is possible that there are 2 bytes in error at
    the locations specified by the parity flags. Solve
    the equation below:
```

$$E_j = \frac{\text{product-syndrome} * \text{alpha}^{8i} + \text{Carleton-syndrome}}{\text{alpha}^{8i} + \text{alpha}^{8j}}$$

```
    alpha8i = (error(location-parity-error(1))*8) mod 127
    alpha8j = (error(location-parity-error(2))*8) mod 127
    reciprocal=127-[CL( CX0(alpha8i) (+) CX0(alpha8j))] mod 128
    IF product-syndrome = 0
    THEN
       psynd-alpha8i = 0
    ELSE
       temp = [[CL(product-syndrome)]          + alpha8i]
                                    mod 128              mod 127
       psynd-alpha8i = CX0(temp)
    END IF
    numerator=[CL( psynd-alpha8i (+) Carleton-syndrome)] mod 128
    Ej = CX0([numerator+reciprocal+127] mod 127)
    Ei = Ej  (+)  product-syndrome
```

Correct the byte at location-parity-error(1) by exclusive or-ing the byte there with the value Ei. Similarly correct the byte at location-error(2) by exclusive or-ing the byte there with the value Ej.

GO TO ACCEPT-THE-PACKET

e) REJECT-THE-PACKET
Update rejection counter, get next packet and start again.

f) ACCEPT-THE PACKET
Update error counters, get next packet and start again

THE BUNDLE DECODER

Envoked under one of the following conditions:
a) The Hamming code in the prefix rejected the current packet
b) The Carleton 2 code rejected the current packet
c) The Carleton 2 code 'corrected' the current packet but the packet is not error free

1 - Tests to see if a Bundle is ready for decoding (ie if a complete Bundle (14 packets have been specified). If a Bundle is ready go to (3).

2 - Update the current Bundle as follows:

(i) If the packet was rejected by the prefix decoder, insert a line of erasures into the Bundle and increment the packets missed counter (RETURN)

(ii) If the packet was rejected by the Carleton 2-byte decoder, insert the undecoded packet into the Bundle and flag the location of this packet. Increment the packets rejected counter (RETURN).

(iii) If the packet was accepted by the Carleton 2-byte decoder as error free, insert the decoded packet into the Bundle (RETURN).

3 - Decode the Bundle as follows:

(i) If more than one packet is missed (Prefix rejection), the Bundle is deemed uncorrectable. The decoding stops and the Bundle rejection is noted for updating in a later routine (RETURN).

(ii) If the only error in the bundle is a rejected or missed line, the Bundle decoder can guarantee correction (RETURN).

(iii) If there was only one packet rejection and no packets missed, the rejected packet is replaced with a line of erasures.

(iv) Transpose the Bundle so that new Carleton 2 byte codewords are formed by interleaving two bytes from each of the 14 vertical packets. The

procedure followed is as in [1].
Decode the new Carleton 2 byte codewords one at a time and count the residual errors.

If any of the new codewords are rejected by the Carleton 2 byte decoder REJECT ALL 14 PACKETS. (RETURN)

If all of the new codewords are accepted by the Carleton 2 byte decoder - update the error counters. (RETURN)

(v) Go to (2) and save the packet that caused the initial call to the subroutine.

[1] B. Mortimer, M. Moore, "More Powerful Error-Correction Scheme for the Broadcast Telidon System",DSS Contract OSU82-00164, March 1983.

Code C Log. look-up Table

mod 128

CL(0)=	255	127
CL(1)=	0	0
CL(2)=	1	1
CL(3)=	159	31
CL(4)=	2	2
CL(5)=	190	62
CL(6)=	160	32
CL(7)=	103	103
CL(8)=	3	3
CL(9)=	135	7
CL(10)=	191	63
CL(11)=	15	15
CL(12)=	161	33
CL(13)=	84	84
CL(14)=	194	194
CL(15)=	201	73
CL(16)=	4	4
CL(17)=	253	125
CL(18)=	136	8
CL(19)=	121	121
CL(20)=	192	64
CL(21)=	79	79
CL(22)=	16	16
CL(23)=	243	115
CL(24)=	162	34
CL(25)=	11	11
CL(26)=	85	85
CL(27)=	166	38
CL(28)=	105	105
CL(29)=	174	46
CL(30)=	223	91
CL(31)=	51	51
CL(32)=	5	5
CL(33)=	210	82
CL(34)=	155	155
CL(35)=	60	60
CL(36)=	137	9
CL(37)=	44	44
CL(38)=	154	154
CL(39)=	153	25
CL(40)=	196	68
CL(41)=	17	17
CL(42)=	67	67
CL(43)=	244	116
CL(44)=	27	27
CL(45)=	164	36
CL(46)=	116	116
CL(47)=	13	13
CL(48)=	150	22
CL(49)=	86	86
CL(50)=	152	24
CL(51)=	167	39
CL(52)=	53	53
CL(53)=	106	106
CL(54)=	147	19
CL(55)=	175	47
CL(56)=	89	89
CL(57)=	223	95
CL(58)=	71	71
CL(59)=	52	52
CL(60)=	238	110
CL(61)=	6	6
CL(62)=	142	14
CL(63)=	211	83
CL(64)=	92	92
CL(65)=	254	126
CL(66)=	30	30
CL(67)=	61	61
CL(68)=	230	102
CL(69)=	138	10
CL(70)=	87	87
CL(71)=	45	45
CL(72)=	178	50
CL(73)=	123	123
CL(74)=	248	120
CL(75)=	206	78
CL(76)=	114	114
CL(77)=	194	66
CL(78)=	41	41
CL(79)=	68	68
CL(80)=	150	22
CL(81)=	81	81
CL(82)=	187	59
CL(83)=	171	43
CL(84)=	76	76
CL(85)=	18	18
CL(86)=	216	88
CL(87)=	198	70
CL(88)=	149	149
CL(89)=	249	121
CL(90)=	104	40
CL(91)=	109	109
CL(92)=	245	117
CL(93)=	37	37
CL(94)=	24	24
CL(95)=	165	37
CL(96)=	104	40
CL(97)=	42	42
CL(98)=	119	119
CL(99)=	241	113
CL(100)=	13	13
CL(101)=	219	91
CL(102)=	157	29
CL(103)=	101	101
CL(104)=	81	81
CL(105)=	236	108
CL(106)=	154	26
CL(107)=	55	55
CL(108)=	168	40
CL(109)=	31	31
CL(110)=	58	58
CL(111)=	205	77
CL(112)=	107	107
CL(113)=	43	43
CL(114)=	146	20
CL(115)=	73	73
CL(116)=	176	48
CL(117)=	112	112
CL(118)=	99	99
CL(119)=	238	110
CL(120)=	224	96
CL(121)=	97	97
CL(122)=	72	72
CL(123)=	226	98
CL(124)=	53	53
CL(125)=	201	73
CL(126)=	239	111
CL(127)=	99	99
CL(128)=	7	7
CL(129)=	131	3
CL(130)=	143	15
CL(131)=	63	63
CL(132)=	212	84
CL(133)=	33	33
CL(134)=	93	93
CL(135)=	232	104
CL(136)=	128	0
CL(137)=	127	127
CL(138)=	31	31
CL(139)=	129	1
CL(140)=	65	65
CL(141)=	130	2
CL(142)=	231	103
CL(143)=	32	32
CL(144)=	139	11
CL(145)=	34	34
CL(146)=	38	38
CL(147)=	213	85
CL(148)=	46	46
CL(149)=	233	105
CL(150)=	179	51
CL(151)=	94	94
CL(152)=	124	124
CL(153)=	132	4
CL(154)=	249	121
CL(155)=	8	8
CL(156)=	207	79
CL(157)=	64	64
CL(158)=	115	115
CL(159)=	144	16
CL(160)=	195	67
CL(161)=	65	65
CL(162)=	42	42
CL(163)=	200	80
CL(164)=	69	69
CL(165)=	145	17
CL(166)=	151	23
CL(167)=	116	116
CL(168)=	82	82
CL(169)=	133	5
CL(170)=	135	7
CL(171)=	135	135
CL(172)=	172	44
CL(173)=	9	9
CL(174)=	77	77
CL(175)=	250	122
CL(176)=	17	17
CL(177)=	234	106
CL(178)=	217	89
CL(179)=	45	45
CL(180)=	199	71
CL(181)=	95	95
CL(182)=	110	110
CL(183)=	180	52
CL(184)=	246	118
CL(185)=	35	35
CL(186)=	28	28
CL(187)=	140	12
CL(188)=	35	35
CL(189)=	214	86
CL(190)=	185	57
CL(191)=	39	39
CL(192)=	165	37
CL(193)=	10	10
CL(194)=	50	50
CL(195)=	173	45
CL(196)=	120	120
CL(197)=	251	123
CL(198)=	242	114
CL(199)=	78	78
CL(200)=	14	14
CL(201)=	134	6
CL(202)=	220	92
CL(203)=	83	83
CL(204)=	158	30
CL(205)=	126	126
CL(206)=	102	102
CL(207)=	189	61
CL(208)=	88	88
CL(209)=	146	18
CL(210)=	237	109
CL(211)=	76	76
CL(212)=	155	27
CL(213)=	117	117
CL(214)=	56	56
CL(215)=	152	24
CL(216)=	169	41
CL(217)=	66	66
CL(218)=	22	22
CL(219)=	196	68
CL(220)=	59	59
CL(221)=	209	81
CL(222)=	204	76
CL(223)=	43	43
CL(224)=	168	104
CL(225)=	215	87
CL(226)=	183	55
CL(227)=	36	36
CL(228)=	149	21
CL(229)=	40	40
CL(230)=	75	75
CL(231)=	186	58
CL(232)=	177	49
CL(233)=	36	36
CL(234)=	113	113
CL(235)=	247	119
CL(236)=	91	91
CL(237)=	141	13
CL(238)=	229	101
CL(239)=	29	29
CL(240)=	31	31
CL(241)=	96	96
CL(242)=	74	74
CL(243)=	202	74
CL(244)=	30	30
CL(245)=	182	54
CL(246)=	147	93
CL(247)=	111	111
CL(248)=	98	98
CL(249)=	221	109
CL(250)=	202	74
CL(251)=	20	20
CL(252)=	240	112
CL(253)=	48	48
CL(254)=	100	100
CL(255)=	218	90

Code C Even Exp. look-up Table

CXE(0)=	1 0 0 0 1 0 0 0	136
CXE(1)=	1 0 0 0 1 0 1 1	139
CXE(2)=	1 0 0 0 1 1 0 1	141
CXE(3)=	1 0 0 0 0 0 0 1	129
CXE(4)=	1 0 0 1 1 0 0 1	153
CXE(5)=	1 0 1 0 1 0 0 1	169
CXE(6)=	1 1 0 0 1 0 0 1	201
CXE(7)=	0 0 0 0 1 0 0 1	9
CXE(8)=	0 0 0 1 0 0 1 0	18
CXE(9)=	0 0 1 0 0 1 0 0	36
CXE(10)=	0 1 0 0 1 0 0 0	72
CXE(11)=	1 0 0 1 0 0 0 0	144
CXE(12)=	1 0 1 1 1 0 1 1	187
CXE(13)=	1 1 1 0 1 1 0 1	237
CXE(14)=	0 1 0 0 0 0 0 1	65
CXE(15)=	1 0 0 0 0 0 1 0	130
CXE(16)=	1 0 0 1 1 1 1 1	159
CXE(17)=	1 0 1 0 0 1 0 1	165
CXE(18)=	1 1 0 1 0 0 0 1	209
CXE(19)=	0 0 1 1 1 0 0 1	57
CXE(20)=	0 1 1 1 0 0 1 0	114
CXE(21)=	1 1 1 0 0 1 0 0	228
CXE(22)=	0 1 0 1 0 0 1 1	83
CXE(23)=	1 0 1 0 0 1 1 0	166
CXE(24)=	1 1 0 1 0 1 1 1	215
CXE(25)=	0 0 1 1 0 1 0 1	53
CXE(26)=	0 1 1 0 1 0 1 0	106
CXE(27)=	1 1 0 1 0 1 0 0	212
CXE(28)=	0 0 1 1 0 0 1 1	51
CXE(29)=	0 1 1 0 0 1 1 0	102
CXE(30)=	1 1 0 0 1 1 0 0	204
CXE(31)=	0 0 0 0 0 0 1 1	3
CXE(32)=	0 0 0 0 0 1 1 0	6
CXE(33)=	0 0 0 0 1 1 0 0	12
CXE(34)=	0 0 0 1 1 0 0 0	24
CXE(35)=	0 0 1 1 0 0 0 0	48
CXE(36)=	0 1 1 0 0 0 0 0	96
CXE(37)=	1 1 0 0 0 0 0 0	192
CXE(38)=	0 0 0 1 1 0 1 1	27
CXE(39)=	0 0 1 1 0 1 1 0	54
CXE(40)=	0 1 1 0 1 1 0 0	108
CXE(41)=	1 1 0 1 1 0 0 0	216
CXE(42)=	0 0 1 0 1 0 1 1	43
CXE(43)=	0 1 0 1 0 1 1 0	86
CXE(44)=	1 0 1 0 1 1 0 0	172
CXE(45)=	1 1 0 0 0 0 1 1	195
CXE(46)=	0 0 0 1 1 1 0 1	29
CXE(47)=	0 0 1 1 1 0 1 0	58
CXE(48)=	0 1 1 1 0 1 0 0	116
CXE(49)=	1 1 1 0 1 0 0 0	232
CXE(50)=	0 1 0 0 1 0 1 1	75
CXE(51)=	1 0 0 1 0 1 1 0	150
CXE(52)=	1 0 1 1 0 1 1 1	183
CXE(53)=	1 1 1 1 0 1 0 1	245
CXE(54)=	0 1 1 1 0 0 0 1	113
CXE(55)=	1 1 1 0 0 0 1 0	226
CXE(56)=	0 1 0 1 1 1 1 1	95
CXE(57)=	1 0 1 1 1 1 1 0	190
CXE(58)=	1 1 1 0 0 1 1 1	231
CXE(59)=	0 1 0 1 0 1 0 1	85
CXE(60)=	1 0 1 0 1 0 1 0	170
CXE(61)=	1 1 0 0 1 1 1 1	207
CXE(62)=	0 0 0 0 0 1 0 1	5
CXE(63)=	0 0 0 0 1 0 1 0	10
CXE(64)=	0 0 0 1 0 1 0 0	20
CXE(65)=	0 0 1 0 1 0 0 0	40
CXE(66)=	0 1 0 1 0 0 0 0	80
CXE(67)=	1 0 1 0 0 0 0 0	160
CXE(68)=	1 1 0 1 1 0 1 1	219
CXE(69)=	0 0 1 0 1 1 0 1	45
CXE(70)=	0 1 0 1 1 0 1 0	90
CXE(71)=	1 0 1 1 0 1 0 0	180
CXE(72)=	1 1 1 1 0 0 1 1	243
CXE(73)=	0 1 1 1 1 1 0 1	125
CXE(74)=	1 1 1 1 1 0 1 0	250
CXE(75)=	0 1 1 0 1 1 1 1	111
CXE(76)=	1 1 0 1 1 1 1 0	222
CXE(77)=	0 0 1 0 0 1 1 1	39
CXE(78)=	0 1 0 0 1 1 1 0	78
CXE(79)=	1 0 0 1 1 1 0 0	156
CXE(80)=	1 0 1 0 0 0 1 1	163
CXE(81)=	1 1 0 1 1 1 0 1	221
CXE(82)=	0 0 1 0 0 0 0 1	33
CXE(83)=	0 1 0 0 0 0 1 0	66
CXE(84)=	1 0 0 0 0 1 0 0	132
CXE(85)=	1 0 0 1 0 0 1 1	147
CXE(86)=	1 0 1 1 1 1 0 1	189
CXE(87)=	1 1 1 0 0 0 0 1	225
CXE(88)=	0 1 0 1 1 0 0 1	89
CXE(89)=	1 0 1 1 0 0 1 0	178
CXE(90)=	1 1 1 1 1 1 1 1	255
CXE(91)=	0 1 1 0 0 1 0 1	101
CXE(92)=	1 1 0 0 1 0 1 0	202
CXE(93)=	0 0 0 0 1 1 1 1	15
CXE(94)=	0 0 0 1 1 1 1 0	30
CXE(95)=	0 0 1 1 1 1 0 0	60
CXE(96)=	0 1 1 1 1 0 0 0	120
CXE(97)=	1 1 1 1 0 0 0 0	240
CXE(98)=	0 1 1 1 1 0 1 1	123
CXE(99)=	1 1 1 1 0 1 1 0	246
CXE(100)=	0 1 1 1 0 1 1 1	119
CXE(101)=	1 1 1 0 1 1 1 0	238
CXE(102)=	0 1 0 0 0 1 1 1	71
CXE(103)=	1 0 0 0 1 1 1 0	142
CXE(104)=	1 0 0 0 0 1 1 1	135
CXE(105)=	1 0 0 1 0 1 0 1	149
CXE(106)=	1 0 1 1 0 0 0 1	177
CXE(107)=	1 1 1 1 1 0 0 1	249
CXE(108)=	0 1 1 0 1 0 0 1	105
CXE(109)=	1 1 0 1 0 0 1 0	210
CXE(110)=	0 0 1 1 1 1 1 1	63
CXE(111)=	0 1 1 1 1 1 1 0	126
CXE(112)=	1 1 1 1 1 1 0 0	252
CXE(113)=	0 1 1 0 0 0 1 1	99
CXE(114)=	1 1 0 0 0 1 1 0	198
CXE(115)=	0 0 0 1 0 1 1 1	23
CXE(116)=	0 0 1 0 1 1 1 0	46
CXE(117)=	0 1 0 1 1 1 0 0	92
CXE(118)=	1 0 1 1 1 0 0 0	184
CXE(119)=	1 1 1 0 1 0 1 1	235
CXE(120)=	0 1 0 0 1 1 0 1	77
CXE(121)=	1 0 0 1 1 0 1 0	154
CXE(122)=	1 0 1 1 1 1 1 1	175
CXE(123)=	1 1 0 0 0 1 0 1	197
CXE(124)=	0 0 0 1 0 0 0 1	17
CXE(125)=	0 0 1 0 0 0 1 0	34
CXE(126)=	0 1 0 0 0 1 0 0	68
CXE(127)=	0 0 0 0 0 0 0 0	0

Code C Odd Exp. look-up Table

APPENDIX C

IMPULSE INVARIANT APPROXIMATIONS TO SLICING FILTERS

C.0 IMPULSE INVARIANT APPROXIMATION TO MODIFIED AVERAGING SLICING FILTER

The analog transfer characteristic of the modified averaging slicing filter can be put in the form

$$T(s) = \frac{1}{s^2 + s\zeta\omega_n s + \omega_n^2}$$

which can be re-written as

$$T(s) = \frac{1}{-2jb}\left[\frac{1}{(s+a+jb)} - \frac{1}{(s+a-jb)}\right].$$

where $a = \zeta\omega_n$ and $b = \omega_n\sqrt{1-\zeta^2}$.

If we make the impulse invariant transformation to the digital domain [3], we get

$$T(z) = \frac{1}{-2jb}\left[\frac{1}{1-e^{-aT}e^{-jbT}z^{-1}} - \frac{1}{1-e^{-aT}e^{jbT}z^{-1}}\right]$$

$$= \frac{1}{b}\ \frac{e^{-aT}\sin(bT)z^{-1}}{1-2e^{-aT}\cos(bT)z^{-1} + e^{-2aT}z^{-2}}\ .$$

Scaling this to the proper level we obtain the following digital approximation

$$T(z) = \frac{\omega_n T}{\sqrt{1-\zeta^2}}\ \frac{e^{-aT}\sin(bT)z^{-1}}{1-2e^{-aT}\cos(bT)z^{-1} + e^{-2aT}z^{-2}}\ .$$

C.1 Impulses Invariant Approximation to Adaptive Slicing Filter

The narrower adaptive slicing filter was implemented in software using a discrete impulse invariant approximation to a two pole Butterworth filter. The analog transfer function of a two pole Butterworth filter is given by:

$$H(s) = \frac{1}{[s-e^{j\frac{3\pi}{4}}(j\Omega_c)]\,[s-e^{-j\frac{3\pi}{4}}(j\Omega_c]}$$

where Ω_c is the cutoff frequency in radians. If we split this function into two first order fractions and do the impulse invariant transformation to the digital domain shown in C.0 then we get

$$H(z)= \frac{1}{\Omega_c(e^{j\frac{3\pi}{4}}-e^{-j\frac{3\pi}{4}})}\left[\frac{1}{1-\exp[e^{-j\frac{3\pi}{4}}\Omega_c T]z^{-1}} - \frac{1}{1-\exp[e^{-j\frac{3\pi}{4}}\Omega_c T]z^{-2}}\right]$$

which when scaled by T becomes

$$H(z) = \sqrt{2}\,\Omega_c T\left[\frac{\exp(-\frac{\Omega_c T}{\sqrt{2}})\sin(\frac{\Omega_c T}{\sqrt{2}})z^{-1}}{1-2\,\exp(-\frac{\Omega_c T}{\sqrt{2}})\cos(\frac{\Omega_c T}{\sqrt{2}})z^{-1}+\exp(1\sqrt{2\Omega}_c T)z^{-2}}\right] .$$