SEMIANALYTIC COMPUTER SIMULATION
OF
DIGITAL COMMUNICATION SYSTEMS

by

Mustafa Aziz Altınkaya

B.S. in E.E. , Boğaziçi University , 1987

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science
in
Electrical and Electronic Engineering

Boğaziçi University

1990

SEMIANALYTIC COMPUTER SIMULATION
OF
DIGITAL COMMUNICATION SYSTEMS

APPROVED BY:

Prof. Dr. Bülent Sankur ....
(Thesis Supervisor)

Doç. Dr. Ali Rıza Kaylan ....

Prof. Dr. Erdal Panayırcı ....

DATE OF APPROVAL: 28.2.1990

ACKNOWLEDGEMENTS

I am very grateful to my thesis supervisor Prof. Dr. Bülent Sankur for his continuous support, encouragement and contributive suggestions during the development of this thesis without which it would never have been possible for me to finish this thesis.

I am also grateful to Prof. Dr. Erdal Panayırcı and Doç. Dr. Ali Rıza Kaylan for being the Commitee Members.

I would like to express my special thanks to my friend Cüneyt Mert for typing parts of the thesis.

I would also like to thank to my friends at TÜBİTAK and BU-Electrical Engineering Department for being kind and helpful.

ABSTRACT

Digital communication systems are subject to various realization imperfections and outer disturbances. The effect of such problems on the systems performance are difficult to solve using only analytical techniques. So, computer simulation becomes an invaluable aid.

In this thesis, for digital communication system simulations a software package SSNDC-Semianalytic Simulator of Nonlinear Digital Channels- has been developed. This software is capable of simulating coherent phase shift keying systems with linear or nonlinear channels in an additive white Gaussian noise environment. All the signals and filters are complex lowpass equivalents of their bandpass counterparts. First a noisefree simulation with a short information sequence is performed and then the effect of additive noise is analytically added. This method together with complex lowpass representation of signals results in much faster simulations than the standard Monte Carlo method.

The program has a modular structure presenting convenience for future modifications. The input data preparation is arranged in the form of menu-driven interactions.

The mathematical theory for the development of the modules of the software is included.

Finally, the results of the simulation as well as their comparison with theoretical solutions or their counterparts in literature are presented.

ÖZET

Sayısal iletişim dizgelerinin başarımı, ideal olmayan gerçekleştirilme ve dış etkenler yüzünden, kuramsal en iyi başarımdan farklılaşma gösterir. Bu farklılaşmanın niceliğinin yalnız analitik yöntemlerle hesaplanması çoğu kez zordur. Bu problemlerin bilgisayar benzetimi ile çözülmesi değerli bir seçenek oluşturmaktadır.

Bu tezde, sayısal iletişim dizgelerinin benzetimini gerçekleştirmek üzere SSNDC (Yarı-Analitik Doğrusal Olmayan Kanallar Benzetimcisi) adında bir yazılım paketi geliştirilmiştir. Bu paket evre uyumlu evre kaydırma anahtarlamalı (CPSK) dizgelerin, doğrusal ve doğrusal olmayan kanallarda, eklenen beyaz Gauss gürültüsü ortamında başarımlarını benzetim yoluyla hesaplamaktadır. Bütün imler ve süzgeçler bant geçiren tasarımlarının karmaşık alçak geçiren eşlenikleri olarak oluşturulmuştur. Önce kısa bir bilgi dizisi ile gürültüsüz ortamda benzetim gerçekleştirilmektedir. Sonra gürültünün başarım üzerindeki etkisi analitik olarak eklenmektedir. Bu yöntem, imlerin karmaşık alçak geçiren gösterimleri ile birlikte ölçün Monte Carlo yönteminden çok daha hızlı benzetimleri olanaklı kılmaktadır.

Geliştirilen yazılım, ilerideki değişikliklere olanak tanımak üzere modüler bir yapıda oluşturulmuştur. Girdiler menülerle etkileşimli olarak hazırlanmaktadır. Modüllerin oluşturulması için gereken matematiksel çıkarımlar kapsanmıştır.

Son bölümde, benzetimle elde edilen sonuçların kuramsal sonuçlarla ve/veya ilgili teknik yayınlarla karşılaştırılmaları yapılmıştır.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABREVIATIONS

CAAD	Computer Aided Analysis And Design
AWGN	Additive White Gausian Noise
TWT	Travelling Wave Tube
$X(\omega)$	Frequency Response
$x_{comp}(t)$	Complex Envelope
$x_{hilb}(t)$	Hilbert Transform
$x_{an}(t)$	Analytic Signal
f_c	Carrier Frequency
f_s	Sampling Frequency
BPSK	Binary Phase Shift Keying
QPSK	Quarternary Phase Shift Keying
OK-QPSK	Offset Quarternary Phase Shift Keying
MSK	Minimum Shift Keying
$b_i(t)$	Bit Sequence
T_b	Bit Period
LPF	Lowpass Filter
α	Phase Offset
U_i	Optimum Demodulator Output
j	Square Root Of {-1}
D	Degradation In Power
P_{be}	Bit Error Probability
ε	Amplitude Offset
E_b	Bit Energy
P_{out}	Optimum Demodulator Output Power

$Q(x)$	Cumulative Gaussian Distribution Function
A	Amplitude
φ	Carrier Static Phase Error
N_0	Noise Power
dB	decibel
ISI	Intersymbol Interference
RF	Radio Frequency
$\tau(\omega)$	Group Delay
Hz	Herz
f_N	Nyquist Frequency
LHP	Left Half Plane
$V_N(x)$	Chebychev Polynomial
$p(t)$	Amplitude Shaping Function
$\theta(t)$	Phase Shaping Function
$\psi_i(t)$	Phase Of i'th Interfering Signal
$R_i(t)$	Envelope Of i'th Interfering Signal
$E\{x\}$	Expectation
(ω_i , u_i)	Gaussian Quadrature Rules
ZMNL	Zero Memory Nonlinear Device
AM/AM	Amplitude Modulation To Amplitude Modulation
AM/PM	Amplitude Modulation To Phase Modulation
I_0	Modified Bessel Function Of 0'th Order
I_1	Modified Bessel Function Of 1'st Order
V_T	Threshold Voltage
B_N	Noise Equivalent Bandwidth

I. INTRODUCTION

1.1 General

The complexity of communication systems has grown considerably during the past few decades. While this growth increases the lead time for analysis and design, the need to insert new technologies into commercial products quickly, requires that the product design be done in a timely , cost effective and error free manner . These demands can be met only through the use of powerful computer aided analysis and design (CAAD) tools. As a result the use of digital simulation has become an important tool for the analysis and design of communication systems.

A central issue in the analysis and design process is that of performance evaluation which can be carried out using formula based mathematical techniques or simulation. In many problems neither approach by itself would be quite satisfactory because of excessive run time in (Monte Carlo) simulation or intractibility in the case of analysis. These two approaches are somewhat complementary and the traditional dichotomy between the two approaches has become blurred in recent years. New techniques have been developed which are based on the combination of these two approaches. In some cases, simulation supports analysis. In other cases, simulation can be made more efficient by judicious use of analysis which allows formulation of the problem or structuring of the simulation so as to reduce the run time.

1.2 Thesis Outline

In this study, for the evaluation of symbol error rates of a digital communication system in additive white Gaussian noise (AWGN) and interference environment, a simulation software named as SSNDC-Semianalytic Simulator of Nonlinear Digital Channels- has been developed.

In chapter 2, complex lowpass representation of bandpass signals is examined. Firstly, the need for complex lowpass representation, especially in simulation is discussed. Then the short description of complex envelopes of the bandpass modulator outputs included in SSNDC is given.

In chapter 3, the degradations resulting from the realization imperfections of digital communication systems are analyzed. These imperfections include phase and amplitude imbalance in modulators, incorrect phase reference at demodulator, synchronization error, nonideal detection filters and effects of transmitter and predetection filters.

Chapter 4 is devoted to linear distortion models of the communication channels. These models present various group delay and amplitude distortions. The sensitivities of different modulators to these distortion types are examined. IIR filters which are utilized in SSNDC are also a part of this chapter.

Chapter 5 examines methods of evaluating symbol error probability in presence of different type of interference and AWGN. For the case of the only interference being intersymbol type, methods presenting some bounds to the probability of error like Chernoff bound or moment space bounds and methods based on Taylor series expansion of characteristic functions are discussed. In this chapter a method which utilizes non-standard Gaussian quadrature rules to evaluate the symbol error probability in the presence of a combination of intersymbol, interchannel, cochannel interferences and AWGN is explained in detail.

Chapter 6 is devoted to nonlinearities introduced in communication systems. Several methods for representation of nonlinearities are discussed. Differnt nonlinearity models are given which are ideal hard limiter, soft limiter, nonlinear amplifiers used in satellite links namely traveling wave tube amplifiers (TWT). Considerable emphasis is given to Volterra series representation of nonlinearities. The method of calculating the probability of symbol error in digital satellite links with nonlinearities using Volterra series is calculated.

Chapter 7 presents techniques for estimating the bit error rate in the simulation of digital communication systems.

Chapter 8 presents the possible choices for linear and nonlinear simulations and the system configuration of SSNDC.

The results and output of the developed software are given in chapter 9. In this chapter , a discussion on the simulation results and the curves, obtained using these results, can be found.

The conclusion and recommendations for the further studies on this topic is given in chapter 10.

In appendix A , a technique for evaluating moments are presented. In appendix B , the computation of Gaussian quadrature rules are given. In appendix C , program listing of SSNDC is given.

II. COMPLEX LOWPASS REPRESENTATION OF BANDPASS SIGNALS

In most cases of interest in digital communication systems, we deal with narrowband bandpass systems. So, in simulating the channel signalling waveform, the utilization of the complex envelope representation drastically reduces the required sampling rate due to the elimination of the need for simulating a high-frequency carrier component [1],[2].

Using this technique, the choice in the working bandwidth in the simulation program depends only on the bandwidth of signals and systems and not on the frequency around which the bandwidth extends.

Narrowband signal representation is based on the concept of the analytical signal. It is well known that, given a real signal x(t) with spectrum X(w), the knowledge of X(w) for positive frequencies is sufficient to get all the information on the signal x(t) because of the Hermitian symmetry in the spectrum [3].

Let x(t) be a narrowband bandpass signal centered around the carrier f_c. Then the analytic signal $x_{an}(t)$ is defined as in equation (2.1) where $x_{hilb}(t)$ denotes the Hilbert transform of x(t). The complex envelope of $x_{an}(t)$ is given in equation (2.2). The relation between x(t) and $x_{comp}(t)$ is given in equations (2.3) and (2.4).

$$x_{an}(t) = x(t) + j x_{hilb}(t) \qquad (2.1)$$

$$x_{comp}(t) = x_{an}(t)\, e^{-j2\pi f_c t} \qquad (2.2)$$

$$x(t) = \mathrm{Re}\left\{ x_{comp}(t)\, e^{j2\pi f_c t} \right\} \qquad (2.3)$$

$$x_{comp}(t) = \left[\, x(t) + j\, x_{hilb}(t) \,\right] e^{-j2\pi f_c t} \qquad (2.4)$$

The spectrums of x(t), $x_{comp}(t)$ and $x_{an}(t)$ are shown in figure 2.1 where f_c and $2f_s$ are the carrier frequency and the the signal bandwidth respectively.

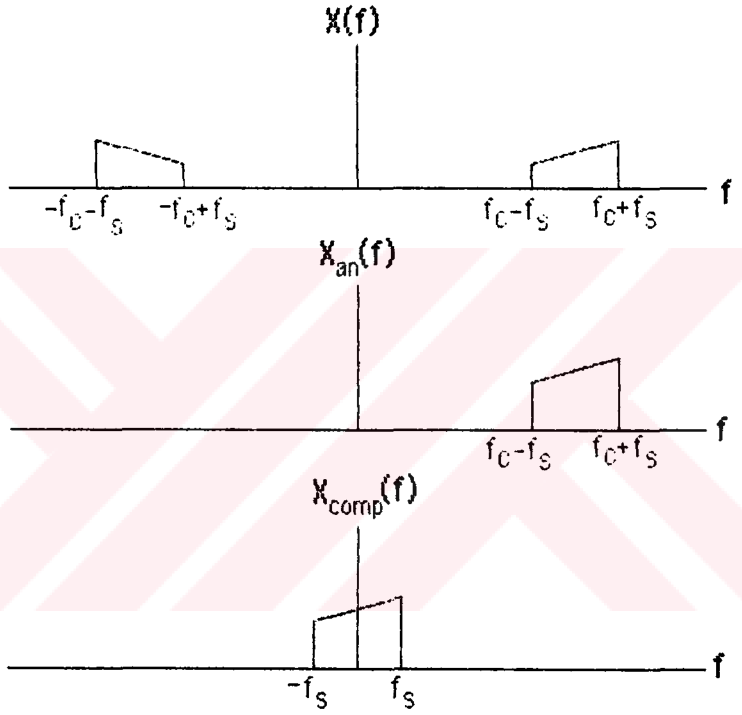

Figure 2.1 Spectrum of a bandpass signal and its analytic and complex versions[3]

Let us consider a narrowband filtering operation where x(t) is the input, h(t) the narrowband impulse response of the linear filter and y(t) the narrowband output. Some simple manipulations of the convolution integral lead to the result in equation (2.5) where the symbol (*) denotes convolution.This result can be easily visualized by examining Figure 2.1 and remembering that convolution in time-domain corresponds to multiplication in frequency-domain.

$$y_{comp}(t) = \frac{1}{2} h_{comp}(t) * x_{comp}(t) \qquad (2.5)$$

Let us represent the narrowband bandpass signal as in equation (2.6).Then $x_{hilb}(t)$, $x_{an}(t)$ and $x_{comp}(t)$ are calculated as shown in the succeeding equations.

$$x(t) = a(t) \cos(2\pi f_c t + \theta(t)) \qquad (2.6)$$

$$= a(t) \left[\cos\left(\theta(t)\right) \cos\left(2\pi f_c t\right) - \sin\left(\theta(t)\right) \sin\left(2\pi f_c t\right)\right]$$

$$x_{hilb}(t) = a(t) \left[\cos\left(\theta(t)\right) \sin\left(2\pi f_c t\right) \right. \qquad (2.7)$$

$$\left. + \sin\left(\theta(t)\right) \cos\left(2\pi f_c t\right)\right]$$

$$x_{an}(t) = a(t) \cos\left(\theta(t)\right)\left[\cos\left(2\pi f_c t\right) + j \sin\left(2\pi f_c t\right)\right] \qquad (2.8)$$

$$+ \ a(t) \sin\left(\theta(t)\right)\left[j \cos\left(2\pi f_c t\right) - \sin\left(2\pi f_c t\right)\right]$$

$$x_{an}(t) = e^{j2\pi f_c t} a(t) e^{j\theta(t)} \qquad (2.9)$$

$$x_{comp}(t) = a(t) e^{j\theta(t)} \qquad (2.10)$$

Examples:

1) $x(t) = \cos(2\pi f_c t) \Rightarrow x_{comp}(t) = e^{j0} = 1$

2) $x(t) = \sin(2\pi f_c t) \Rightarrow x_{comp}(t) = e^{-j\pi/2} = -j$

2.1 Complex Envelopes of Some Modulated Waves

The complex envelopes of bandpass modulated signals can be easily obtained by using equations (2.6) and (2.10).Now we will give the equations of the bandpass signals and their complex envelopes for some popular modulation techniques [3],[4].

For BPSK the bandpass signal $x(t)$ and its complex envelope $x_{comp}(t)$ are given in equations (2.11) and (2.12) where $b(t)=\pm 1$ and represents the bit sequence.

$$x(t) = A\, b(t) \cos(2\pi f_c t) \qquad (2.11)$$

$$x_{comp}(t) = A\, b(t) \qquad (2.12)$$

for QPSK where $b_1(t)$ and $b_2(t)$ are the bit sequences in quadrature channels

$$x(t) = A\, b_1(t) \cos(2\pi f_c t) + A\, b_2(t) \sin(2\pi f_c t) \qquad (2.13)$$

$$x_{comp}(t) = A\, [b_1(t) + j\, b_2(t)] \qquad (2.14)$$

the OK-QPSK modulator is essentially a QPSK modulator where $b_1(t)$ and $b_2(t)$ are staggered 1/2 symbols.In MSK modulators $b_1(t)$ and $b_2(t)$ are defined by equations (2.15) and (2.16) and they are staggered 1/2 symbols.

$$b_1(t) = \cos(\pi t/(2T_b)) \qquad (2.15)$$

$$b_2(t) = \sin(\pi t/(2T_b)) \qquad (2.16)$$

where T_b is the bit period.

III. PERFORMANCE DEGRADATIONS DUE TO REALIZATION IMPERFECTIONS

The theoretical performance of digital communication systems can hardly be attained in reality. Among the reasons of this fact, change in the channel characteristics in the AWGN case when various functional blocks in digital communication system are nonideally realized, are the major ones. In this part of our study we will concentrate ourselves on realization imperfections. Considerations will be limited to coherent communication systems and linear channels.

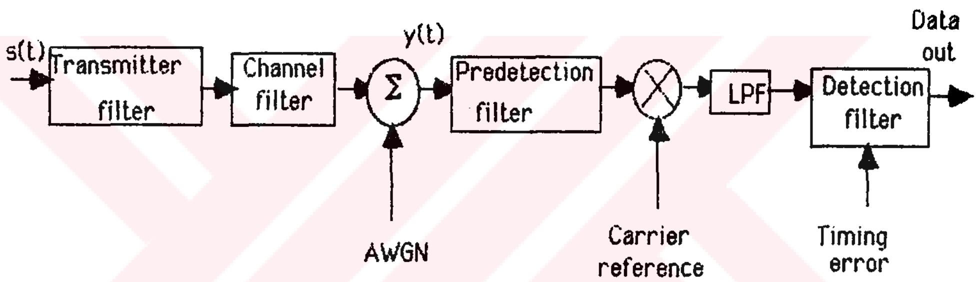

Figure 3.1 Model of nonideal communication system [4]

A general model for considering these impairments is shown in figure 3.1. The degradations to be considered are listed below [4].

1. Phase and amplitude imbalance in BPSK
2. Phase and amplitude imbalance in QPSK
3. Power loss due to transmitter filtering
4. Phase offset of the carrier recovery circuitry
5. Nonideal detection filter
6. Predetection filtering
7. Bit synchronizer timing error

3.1 Phase and Amplitude Imbalance in BPSK

In some BPSK modulators perfect switching of π-radians of the modulated output does not occur. The complex envelope of an imperfectly phase switched BPSK signal can be represented as :

$$u(t) = A\, e^{j\frac{\pi}{2}\left[-1+\left(1-\alpha\right)b(t)\right]} \tag{3.1}$$

where $b(t)=\pm 1$ is the input data stream to the demodulator and $0 \le \alpha \le 1$ is the fractional error from perfect phase swithing. The case $\alpha=0$ corresponds to ideal BPSK waveform. The output of an optimum coherent demodulator for u(t) in equation (3.1) will be given in (3.2) where r(t) is the received waveform which is not subject to attenuation or phase shift for the sake of ease in illustration [3].

$$U_i = \mathrm{Re}\left\{\int_0^T r(t)\, u_i^*(t)\, dt\right\} \qquad i=1,2 \tag{3.2}$$

$u_1(t)$ is the ideal BPSK signal with unity amplitude assuming $b(t)=+1$. So $u_1^*(t)$ and the resulting optimum demodulator output U_1 become

$$u_1^*(t) = \left[e^{j\frac{\pi}{2}[-1+1]}\right]^* = 1 \tag{3.3}$$

$$U_1 = \mathrm{Re}\left\{A\, e^{j\frac{\pi}{2}\left[-1+(1-\alpha)\,b(t)\right]}\right\} \tag{3.4}$$

$$= A\,\mathrm{Re}\left\{e^{j\frac{\pi}{2}\left[-1+(1-\alpha)\,b(t)\right]}\right\}$$

$$= A \sin\left[\frac{\pi}{2}(1-\alpha)\, b(t)\right]$$

$$= A \sin\left[\frac{\pi}{2}\, b(t)\right] \cos\left[\frac{\pi \alpha\, b(t)}{2}\right]$$

$$= A\, b(t) \cos\left[\frac{\pi \alpha}{2}\right]$$

The cosine term in the equation corresponds to a demodulation power loss of D_{ph} given in equation (3.5) which represents the required additional signal power to attain the same probability of error as in case of no imbalance is present.

$$D_{ph}(dB) = 20 \log_{10}\left[\cos\left(\frac{\alpha \pi}{2}\right)\right] \qquad (3.5)$$

Amplitude imbalance in BPSK occurs when there is an offset in the amplitude level. Complex envelope of the BPSK signal; u(t), which is subject to amplitude imbalance, the corresponding coherent demodulator output; U and the resulting bit error probability; P_{be} will be:

$$u(t) = A\left[\varepsilon + b(t)\right] e^{j\frac{\pi}{2}(-1+b(t))} \qquad (3.6)$$

$$U = A\left[\varepsilon + b(t)\right] \qquad (3.7)$$

$$P_{be} = \frac{1}{2}\left[Q\left(\sqrt{\frac{2E_b}{N_0}}\,(1+\varepsilon)\right) + Q\left(\sqrt{\frac{2E_b}{N_0}}\,(1-\varepsilon)\right)\right] \qquad (3.8)$$

The degradations due to phase and amplitude imbalances cannot be added simply because no clear power degradation term for amplitude imbalance can be given. The combined effects of these degradations on BPSK for $P_{be}=10^{-6}$; obtained by our simulation program; are shown in figure 9.3 with amplitude imbalance as a parameter.

3.2 Phase and Amplitude Imbalance in QPSK

The complex envelope of a QPSK modulator with phase imbalance is given by equation (3.9) where $b_1(t)$ and $b_2(t)$ are the bit sequences modulating two orthogonal carriers and $\beta \geq 0$ represents a phase imbalance. The outputs of the orthogonal coherent demodulators are given by equations (3.10) and (3.11) respectively.

$$u(t) = A\left[b_1(t)\, e^{j\frac{\beta}{2}} + b_2(t)\, e^{-j\left(\frac{\pi}{2}+\frac{\beta}{2}\right)} \right] \tag{3.9}$$

Defining $u_1(t)$ and $u_2(t)$ as: $u_1(t) = 1$

$$u_2(t) = -j$$

$$U_1 = \mathrm{Re}\left\{ \int_0^T u(t)\, u_1^*(t)\, dt \right\} \tag{3.10}$$

$$= A\left[b_1(t)\cos\frac{\beta}{2} - b_2(t)\sin\frac{\beta}{2} \right]$$

$$U_2 = \mathrm{Re}\left\{ \int_0^T u(t)\, u_2^*(t)\, dt \right\} \tag{3.11}$$

$$= A\left[-b_1(t)\sin\frac{\beta}{2} + b_2(t)\cos\frac{\beta}{2} \right]$$

In addition to the power loss in demodulator outputs, the term with $b_2(t)$ in U_1 and the term with $b_1(t)$ in U_2 represents the crosstalk between the quadrature channels.The effect of those two degradations on the receiver performance can be evaluated by analyzing the coherent demodulator outputs. Each of them is of the form given in (3.12). They result in two different output powers P_{out1} and P_{out2}. The corresponding probability of bit error is given by P_{be} in equation (3.14) where E_b is the energy per bit and N_0 is the single sided noise spectral density.

$$U_i = \pm A\left(\cos\frac{\beta}{2} \pm \sin\frac{\beta}{2}\right) \qquad (3.12)$$

$$P_{out1} = A^2\left(\cos\frac{\beta}{2} + \sin\frac{\beta}{2}\right)^2 \qquad (3.13)$$

$$P_{out2} = A^2\left(\cos\frac{\beta}{2} - \sin\frac{\beta}{2}\right)^2$$

$$P_{be} = \frac{1}{2}\left[Q\left(\sqrt{\frac{2E_b}{N_0}}\left(\cos\frac{\beta}{2}+\sin\frac{\beta}{2}\right)\right)+Q\left(\sqrt{\frac{2E_b}{N_0}}\left(\cos\frac{\beta}{2}-\sin\frac{\beta}{2}\right)\right)\right] \qquad (3.14)$$

Expressing the degradation as the amount of increase in bit energy E_b; to attain the same error probability, it is seen that, there is no simple expression as in the BPSK case since the error probability is the average of two Q(.) functions. So a numerical solution is required. Figure 3.2 is a plot of phase degradation in QPSK for a bit error probability of 10^{-6}.

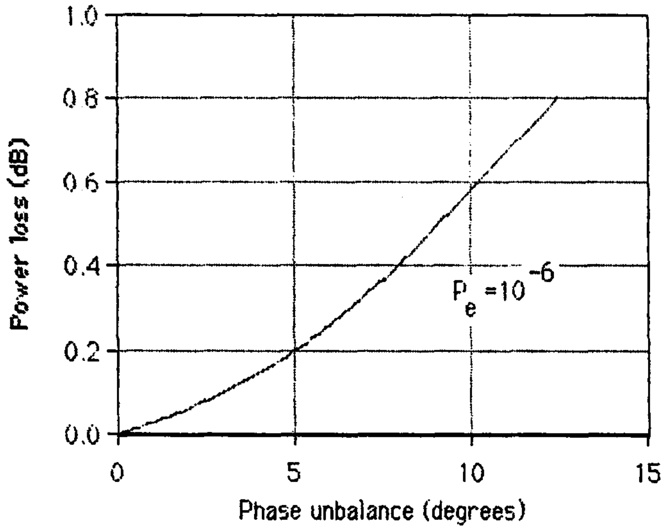

Figure 3.2 Degradation due to phase imbalance in QPSK [4]

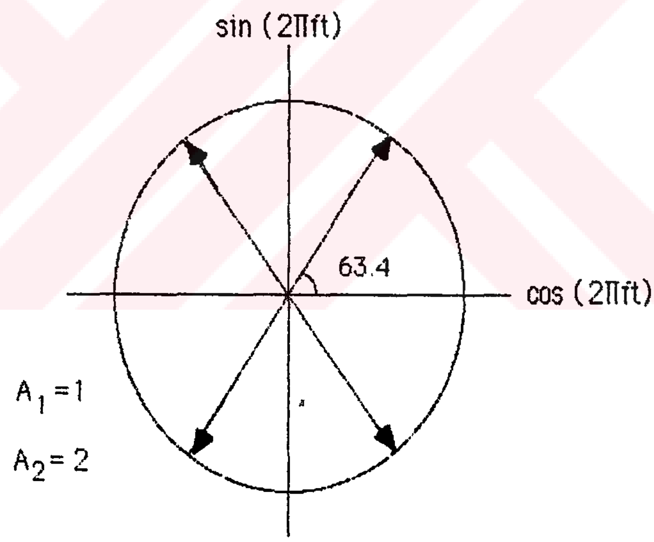

Figure 3.3 Phasor diagram of QPSK with amplitude imbalance[4]

Degradation due to amplitude imbalance is refered to as difference in quadrature carrier amplitudes. Let the complex envelope of QPSK given as in equation (3.15). It will result in the decision variables and the bit error probability of equation (3.16) and (3.17) respectively. The IQ-plot of a QPSK modulator output with $A_1=1$ and $A_2=2$ is given in figure 3.3.

$$u(t) = A_1 b_1(t) + A_2 b_2(t)\, e^{-j\frac{\pi}{2}} \tag{3.15}$$

$$U_1 = A_1 b_1(t) \tag{3.16}$$

$$U_2 = A_2 b_2'(t)$$

$$P_{be} = \frac{1}{2}\left[Q\left(\sqrt{\frac{2A_1^2T}{N_0}}\right) + Q\left(\sqrt{\frac{2A_2^2T}{N_0}}\right)\right] \tag{3.17}$$

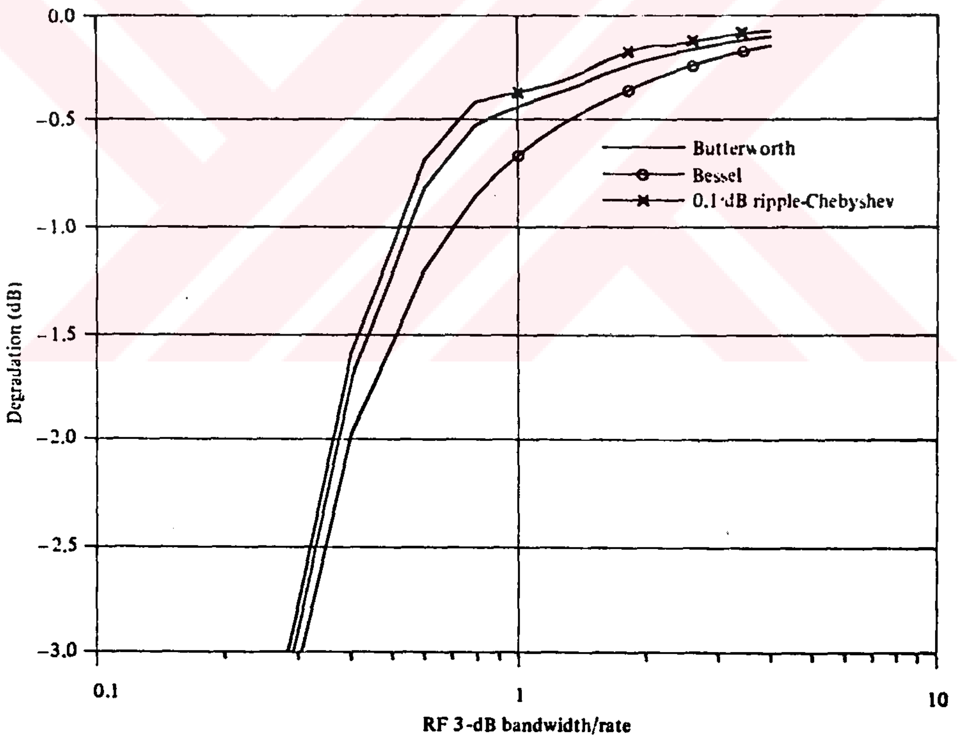

Figure 3.4 Power loss degradation due to filtering QPSK; third order filters[5]

3.3 Power Loss due to Filtering the Modulated Signal

To limit the out-of-band power of the transmitter, band-pass filters are placed after the modulators. The price of them will be the power loss and phase distortions due to nonideal filter characteristics which cause intersymbol interference. The power loss degradations due to filtering QPSK signal with 3rd order Butterworth, Bessel and 0.1 dB ripple Chebychev filters are shown in the figure 3.4.

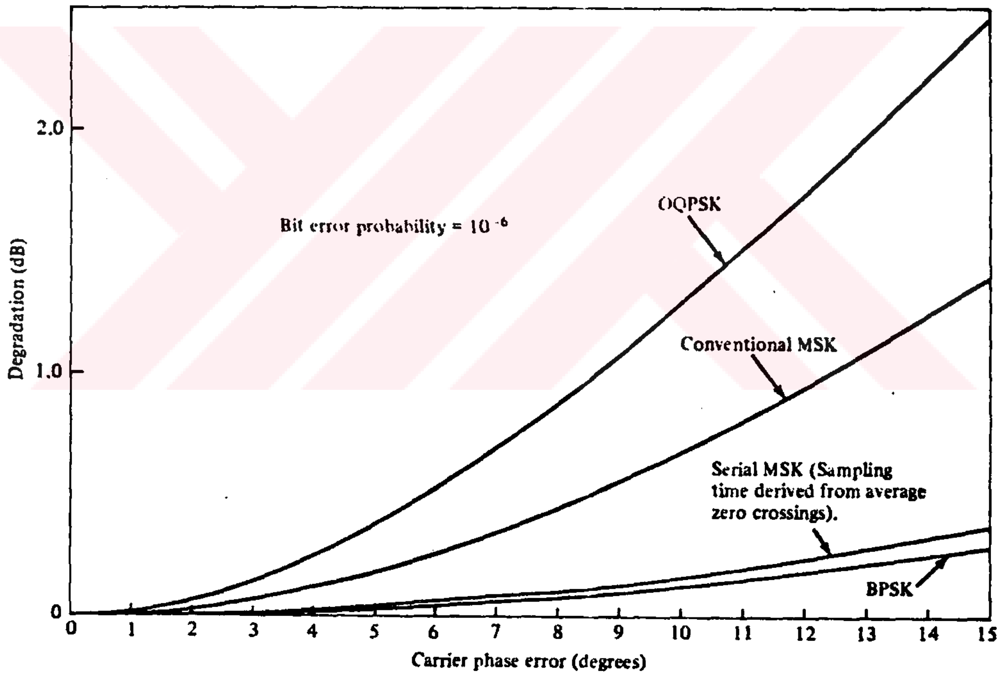

Figure 3.5 Degradation due to demodulator static phase error [4]

3.4 Phase Offset of the Carrier Recovery Circuitry

When there is a phase reference offset of φ radians at a coherent demodulator of a BPSK system, degradation effects similar to phase imbalance at modulator case are observed. Let the complex envelope of BPSK signal be given by equation (3.18) where α is a constant phase. The output of a coherent demodulator with φ radians phase offset and the corresponding power loss are given by U in equation (3.19) and by $D_{bpsk\varphi}$ in (3.20) respectively.

$$u(t) = A\, b(t)\, e^{j\alpha} \tag{3.18}$$

$$U = \mathrm{Re}\left\{ \int_0^T u(t)\, u_1^*(t)\, dt \right\} \tag{3.19}$$

$$= A\, b(t)\, \mathrm{Re}\left\{ e^{j\alpha} \left[e^{j(\alpha+\varphi)} \right]^* \right\}$$

$$= A\, b(t) \cos\varphi$$

$$D_{bpsk\,\varphi}(dB) = 20 \log_{10}\left(\cos\varphi \right) \tag{3.20}$$

For quadrature modulation systems the analysis of demodulator phase error is more complicated due to the crosstalk introduced between the quadrature channels. Similar steps, as in the phase-imbalance in modulator case, are followed. Degradations due to demodulator phase error at BPSK, QPSK and MSK systems are shown in figure 3.5. In serial MSK systems, the sampling time is derived from average zero crossings. The effect will be negligible crosstalk between quadrature channels which explains the fact that, serial MSK and BPSK are subject to essentially same amount of degradation.

3.5 Nonideal Detection Filter

The implementation of an ideal correlation type filter may be difficult. Also considering the freedom from the necessity to dump the filter, the use of an intentionally mismatched data filter can be very desirable [5].

When this is done the performance of the receiver is degraded from that of the optimum receiver for two reasons. First at the output of such a filter the ratio of peak signal to root-mean-square noise will be less than the theoretically maximum value $2E/N_0$, E being the signal energy and N_0 the noise power spectral density. Second, a suboptimum filter will introduce intersymbol interference from one symbol period to succeeding symbol periods due to its transient response.

Figure 3.6 shows average probability of error versus BT product where B is the two-sided RF-equivalent detection filter bandwidth, with E_b/N_0 as a parameter.

The integrate and dump filter is not always the best choice (even when it can be implemented conveniently) because it is not the matched filter for distorted signals. Improved performance with distorted signals may be available with simpler data filters. This fact can be seen in the figure 3.7 where 0.1 dB ripple Chebychev filters distort the signal.

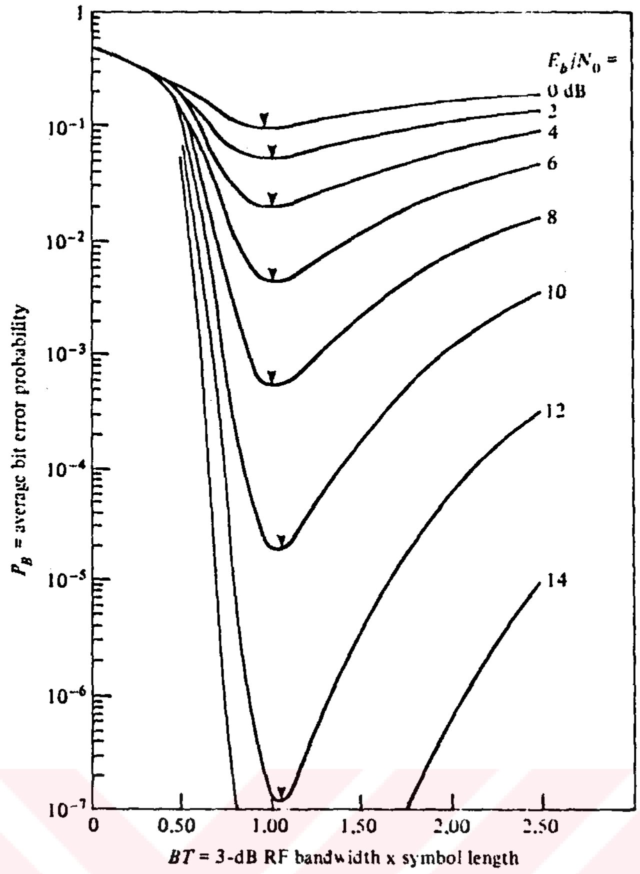

Figure 3.6 Two pole Butterworth data filter detection of coherent QPSK and BPSK [4]

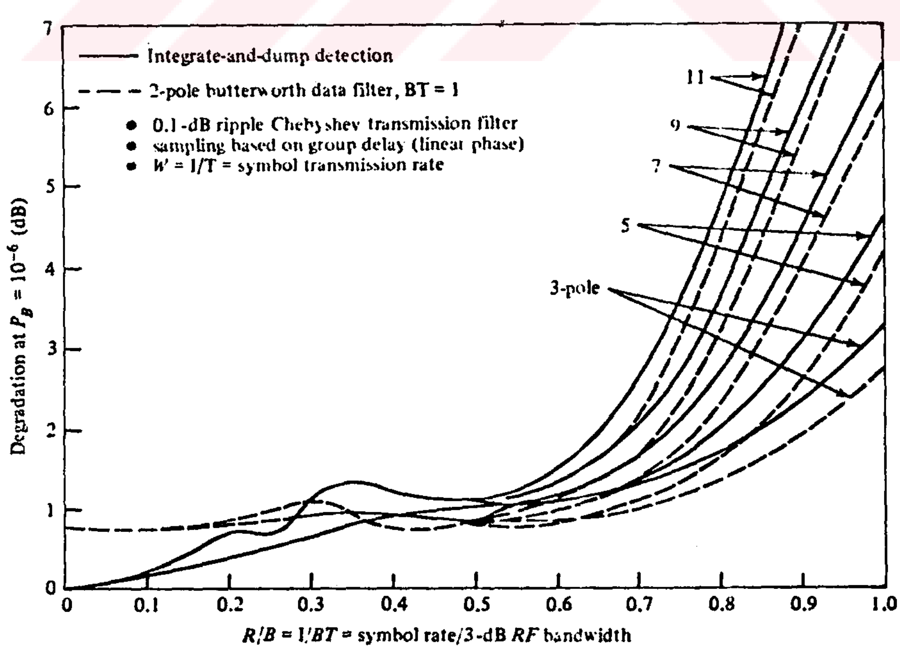

Figure 3.7 Comparison of Matched and Butterworth data filters [5]

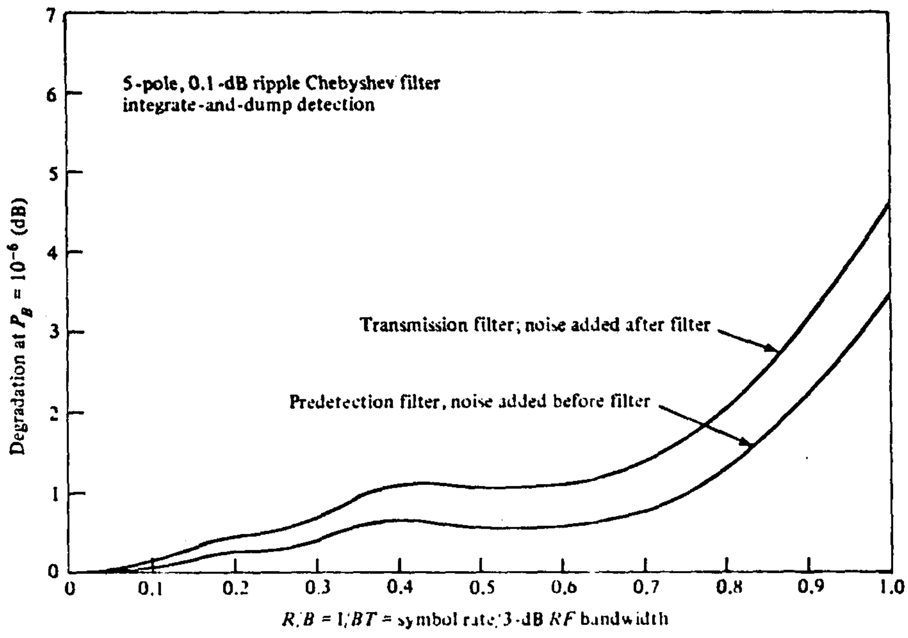

Figure 3.8 Comparison of transmission and predetection filtering with QPSK signals [5]

3.6 Predetection Filtering

Since all practical receivers involve filtering prior to the detection operation, which may be for example due to an intermediate frequency amplifier, the effects of filters placed prior to an ideal matched filter receiver are now considered. In figure 3.8 the degradation in the case of antipodal base band signalling or, equivalently BPSK signalling which includes QPSK and OK-QPSK when viewed on a per quadrature channel basis are shown. The two curves in the figure show the degradation as a function of symbol rate normalized by the 3-dB RF-bandwidth of the filter. The only difference is in the placement of the Chebychev filters. In the case of predetection filtering, the noise is added before filtering and in the case of transmitter filtering the noise is added after filtering. The predetection filtering case introduces less degradation due to the noise rejection effect of the filter. So the effect of signal power loss and intersymbol interference caused by filtering is less compared to transmitter filtering case.

3.7 Bit Synchronizer Timing Error

The degradation due to bit synchronization error in baseband systems are investigated by several authors; [6],[7],[8]; for a number of pulseshapes including ideal bandlimited, Gaussian, Chebychev pulses.

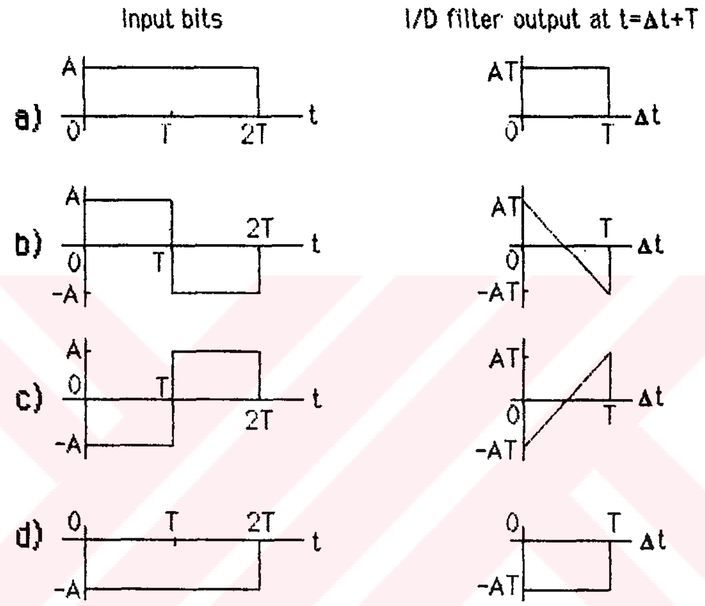

Figure 3.9 I/D filter outputs for bit combinations as function of sampling time[4]

The I/D filters output for PSK systems is similar to the case of antipodal baseband signalling with rectangular pulses. For analysis two adjacent bits must be considered. The I/D filter outputs for four different bit combinations at times $\Delta t+T$ is shown in the figure 3.9

The degradation in SNR is proportional to the square of matched filters output as a function of $\Delta t/T$. For each case the relative degradation is given in equation (7.1). The resulting average bit error probability for BPSK assuming equally likely sequences is given in (7.2).

$$D_a = D_d = 1 \qquad 0 \le t \le T \tag{3.21}$$

$$D_b = D_c = \left(1 - \frac{2\Delta t}{T} \right)^2 \qquad 0 \le t \le T$$

$$P_{be} = \frac{1}{2}\left[Q\left(\sqrt{\frac{2D_a E_b}{N_0}} \right) + Q\left(\sqrt{\frac{2D_b E_b}{N_0}} \right) \right] \tag{3.22}$$

IV. IMPERFECT LINEAR CHANNELS

To obtain optimum performance in digital communication systems we try to satisfy Nyquist's ISI-free transmission criterion. In many applications a virtually ideal received pulse spectrum can be obtained with available hardware. The variation from the ideal arises from the non-ideal phase and amplitude characteristics.This phase distortion or the corresponding group delay distortion together with any amplitude distortion, introduce ISI. In this case additional signal power is required to achieve the same performance as in the distortionless AWGN case.

The quadratic and linear delay distortion models, used in our simulation program, are shown in figures 4.9 and 4.10, respectively. When the double sided RF-bandwidth is $2f_{max}$, the ideal group delay in the $f_c \pm f_{max}$ range should be constant which corresponds to a linear phase,remembering that group delay is defined as in equation (4.1) where τ is the group delay spectrum and β is the phase spectrum.

$$\tau(\omega) = -\frac{d\beta(\omega)}{d\omega} \tag{4.1}$$

Quadratic delay distortion is in theory approached near midband of a flat bandpass channel with sharp cut-offs such as a carrier system voice channel and approximates the type of delay distortion often encountered in channels without phase equalization [10]. The corresponding phase distortion will be cubic. Minimum eye diagram opening at the best sampling instant with raised cosine pulse spectrum and quadratic delay distortion is given in figure 4.1 for coherent BPSK and QPSK, for DPSK (BPSK with differential phase detection) and DQPSK (QPSK with differential phase detection) [11].

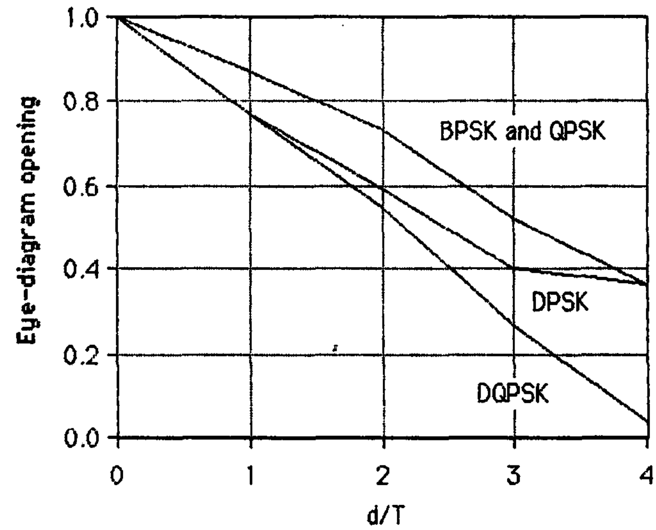

Figure 4.1 Maximum transmission impairments with raised-cosine pulse spectrum and quadratic delay distortion (d: maximum delay distortion over band f_{max})[11]

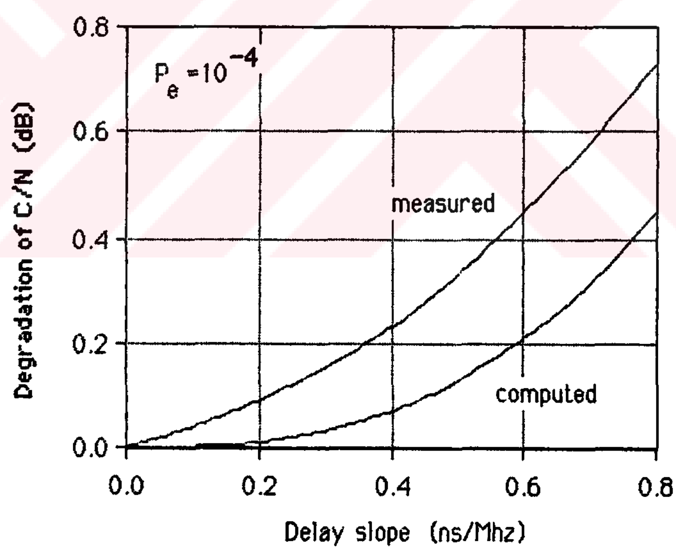

Figure 4.2 Measured and computed C/N degradation due to delay slope on a 45-Mbit/s offset QPSK, 50% raised-cosine system[12]

After phase equalization of quadratic delay distortion a linear delay distortion component may remain, owing to inexact equalization. Delay distortion encountered in troposcatter channels as a result of frequency selective fading can also be represented as linear distortion type [10]. The corresponding phase characteristics will be quadratic. Computer calculated and measured carrier to noise power; (C/N); degradations due to delay slope distortion are shown in figure 4.2. For the measurements a 45Mbit/s bit rate OK-QPSK modem was employed [12].

Figure 4.3 illustrates the degradation of QPSK and BPSK signals resulting from quadratic or cubic phase distortions [5]. The degradation in power is plotted as a function of the phase deviation in degrees measured at a frequency displaced by the symbol rate from the carrier. Matched filter detection is utilized. A parabolic phase distortion causes larger degradation with QPSK signals than with BPSK. This is because of the crosstalk introduced between the quadrature channels of the QPSK system. In order not to introduce crosstalk, the distortion filter must have a real impulse response. If the impulse response of a filter is to be real, it must have antisymetrical phase characteristics. So a parabolic phase distortion which is a symetrical phase function, introduces crosstalk between the quadrature

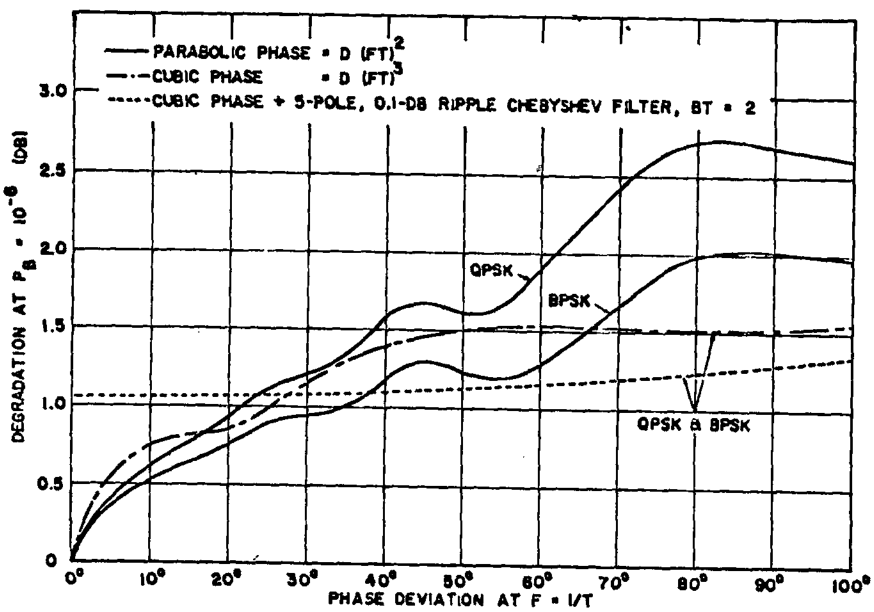

Figure 4.3 Effect of pure phase distortion with integrate-and-dump detection [4]

channels of a QPSK system. With cubic phase distortion which is an antisymetrical phase function; the impulse response of the distortion filter is real, resulting at the same amount of degradation in QPSK and BPSK systems. The figure also shows the result where a cubic phase distortion is cascaded with a five pole 0.1dB ripple Chebychev filter (with BT=2). The Chebychev filter intoduces an initial degradation in the absence of phase distortion. Notably however, the introduction of a bandlimiting transmission filter reduces the sensitivity to phase distortion.

The major amplitude distortions are linear and parabolic. They are encountered in fading multipath channels. Computer calculated and measured carrier to noise power; (C/N); degradations due to amplitude slope distortion are shown in figure 4.4 [12] . Other possible amplitude and delay distortions are ripple type. The detailed description of these filters are given in the following section.

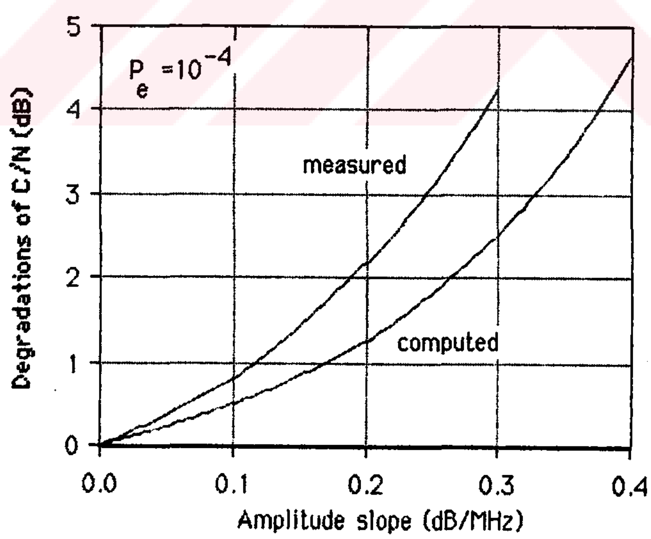

Figure 4.4 Measured and computed C/N degradation due to amplitude slope on a 45-Mb/s offset QPSK, 50% raised-cosine system [12]

4.1 Linear Distortion Models of SSNDC

All of the following filters are complex lowpass equivalents of their bandpass versions. So the frequencies in the figures and equations are shifted by the carrier frequency and normalized to the baud rate.

4.1.1) Linear Amplitude Distortion:

The linear amplitude distortion can be defined as (see figure 4.5) in equation (4.2) where b is the amplitude slope defined in terms of dB/Hz and f is the frequency in Hz [9]. The transfer function of the distortion filter; $H_D(f)$; is obtained as in equation (4.3).

$$D = bf \tag{4.2}$$

$$H_D(f) = 10^{\,bf/20} \tag{4.3}$$

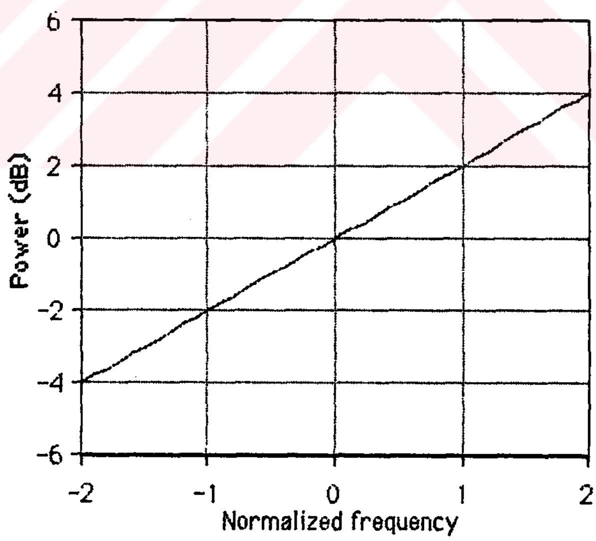

Figure 4.5 Power spectrum of linear amplitude distortion[10]

4.1.2) Parabolic Amplitude Distortion:

The parabolic amplitude distortion can be defined as (see figure 4.6) in equation (4.4) where b is the amplitude variation defined in terms of

$dB/(Hz)^2$ and f is the frequency in Hz. The transfer function of the distortion filter is obtained as in equation (4.5).

$$D = bf^2 \tag{4.4}$$

$$H_D(f) = 10^{bf^2/20} \tag{4.5}$$

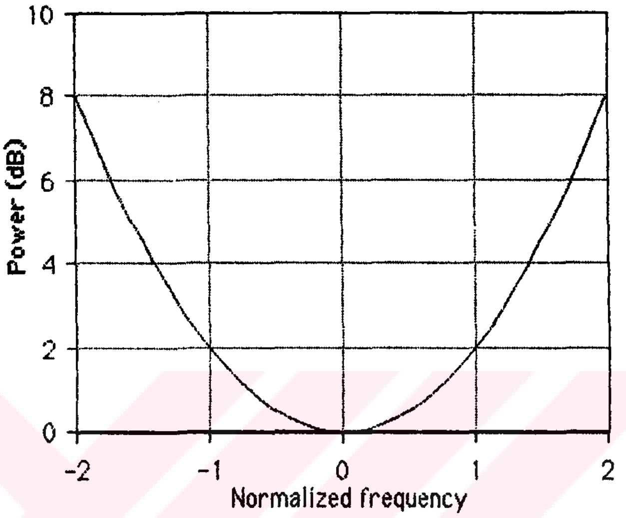

Figure 4.6 Power spectrum of quadratic amplitude distortion[10]

4.1.3) Ripple Amplitude Distortion:

There are two types of ripple amplitude distortion. Type I ripple amplitude distortion is defined as (see figure 4.7) in equation (4.6) where b is the maximum amplitude distortion in the passband in (dB), c the number of ripples in the passband and f_N is the Nyquist frequency in Hz. The corresponding transfer function of the distortion filter is given as in equation (4.7).

$$D_I(dB) = b \sin\left(\pi fc/f_N\right) \tag{4.6}$$

$$H_{D_I}(f) = 10^{\frac{b}{20}\sin\left(\pi fc/f_N\right)} \tag{4.7}$$

Type II. ripple amplitude distortion differs from type I. in that, it has a peak at the center of the passband. The distortion in logarithmic scale and the transfer fuction of the distortion filter are given as (see figure 4.8) in equations (4.8) and (4.9).

$$D_{II}(dB) = b \cos\left(\pi fc/f_N\right) \qquad (4.8)$$

$$H_{D_{II}}(f) = 10^{\frac{b}{20}\cos\left(\pi fc/f_N\right)} \qquad (4.9)$$

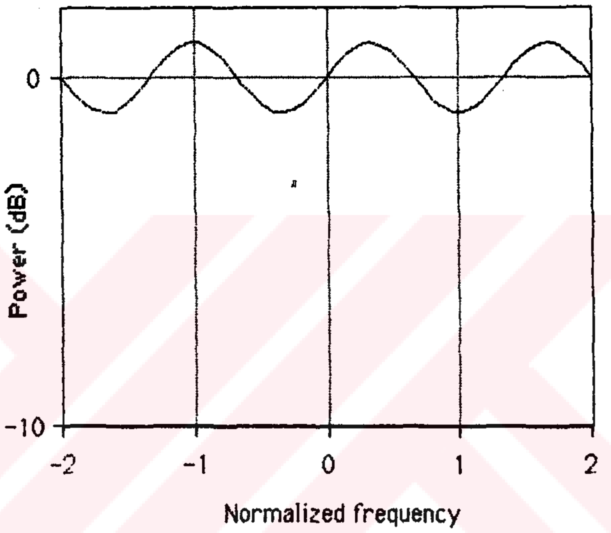

Figure 4.7 Ripple amplitude distortion type I [9]

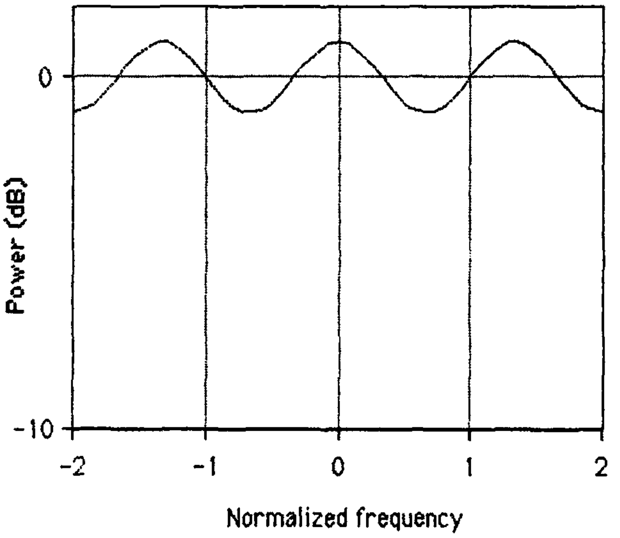

Figure 4.8 Ripple amplitude distortion type II [9]

4.1.4) Linear Group Delay:

By definition the linear group delay is given by (see figure 4.9) equation (4.10) where b is the delay slope and given in terms of sec/Hz and f is given in Hz. Refering to equation (4.1) the phase distortion can be found as in equation (4.11). If b is given in ns/MHz and f is given in MHz, the phase distortion is found as in equation (4.12). The factor 10^{-3} comes from the change in the units.

$$\tau = bf \tag{4.10}$$

$$\phi = -\pi bf^2 \tag{4.11}$$

$$\phi = -\pi bf^2 \, 10^{-3} \tag{4.12}$$

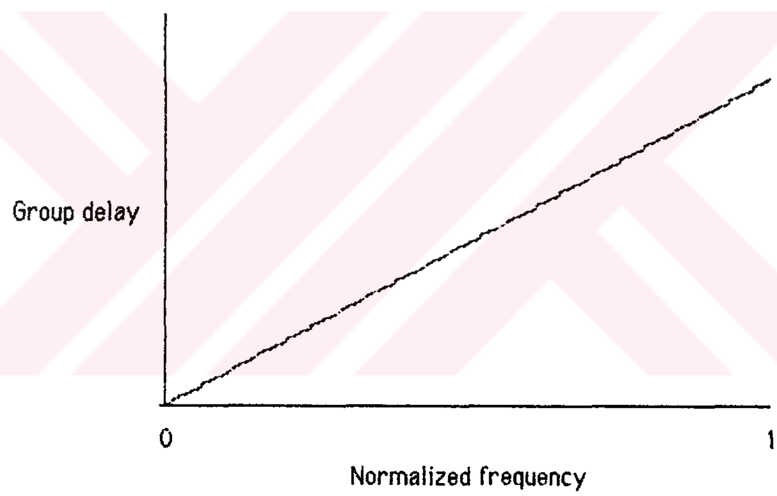

Figure 4.9 Linear group delay [11]

4.1.5) Parabolic Group Delay:

By definition the parabolic group delay is given by (see figure 4.10) equation (4.13) where b is given in sec/Hz and f is given in Hz. The corresponding phase distortion is found as in equations (4.14); or (4.15), where b is given in ns/(MHz)^2 and f is given in MHz.

$$\tau = bf^2 \tag{4.13}$$

$$\phi = (-2\pi bf^3)/3 \tag{4.14}$$

$$\phi = \frac{-2\pi}{3} bf^3 10^{-3} \qquad (4.15)$$

Figure 4.10 Parabolic group delay [11]

4.1.6) Ripple Group Delay:

There are two types of ripple group delay. Type I. ripple group delay is defined as (see figure 4.11) in equation (4.16) where b is the maximum group delay in the passband in sec, c the number of ripples in the passband and f_N the Nyquist frequency. The phase distortion is obtained as in equation (4.17) where ϕ is in radians. Equivalently, when b is in ns and f and f_N is in MHz equation (4.18) is obtained.

$$\tau_I = b \cos\left[\pi fc/f_N\right] \qquad (4.16)$$

$$\phi_I = -2\pi b \left(\frac{f_N}{c}\right) \sin\left(\pi fc/f_N\right) \qquad (4.17)$$

$$\phi_I = -2\pi b \left(\frac{f_N}{c}\right) 10^{-6} \sin\left(\pi fc/f_N\right) \qquad (4.18)$$

The corresponding equations for ripple group delay type II (figure 4.12) are as follows.

$$\tau_{ll} = b \sin\left[\pi f c/f_N\right] \tag{4.19}$$

$$\phi_{ll} = -2\pi b \left(\frac{f_N}{c}\right) \cos\left(\pi f c/f_N\right) \tag{4.20}$$

$$\phi_{ll} = -2\pi b \left(\frac{f_N}{c}\right) 10^{-6} \cos\left(\pi f c/f_N\right) \tag{4.21}$$

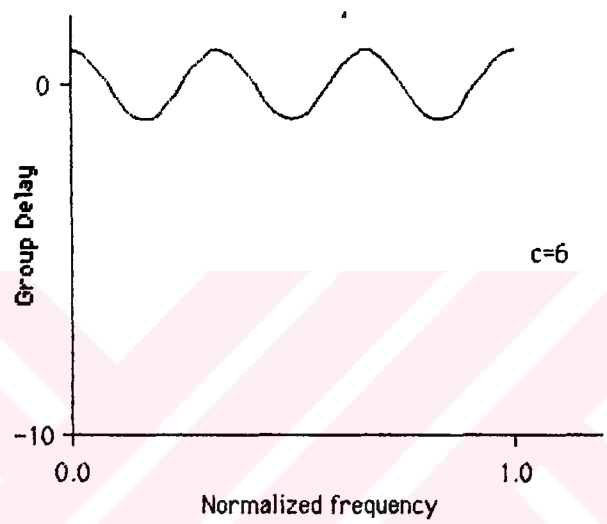

Figure 4.11 Ripple group delay type I [9]

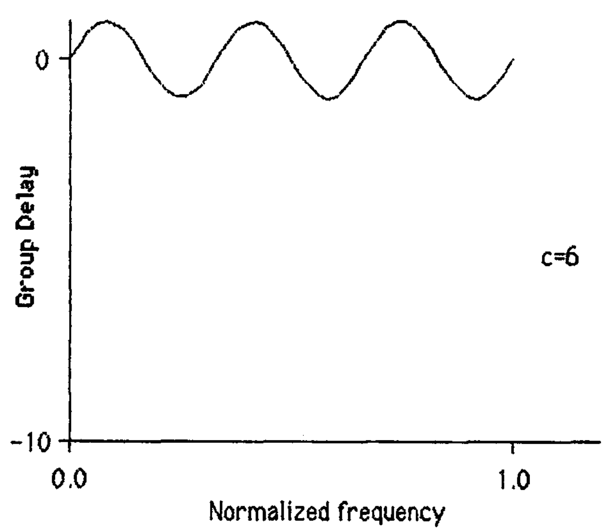

Figure 4.11 Ripple group delay type II [9]

4.2 Linear Filters of SSNDC

4.2.1) Butterworth Filters:

Butterworth filters, also called maximally flat amplitude response filters, are commonly used types of linear filters in communication systems. An n'th order normalized Butterworth filter has a magnitude function given by:

$$|H(j\omega)|^2 = \frac{1}{1+\omega^{2n}} \tag{4.22}$$

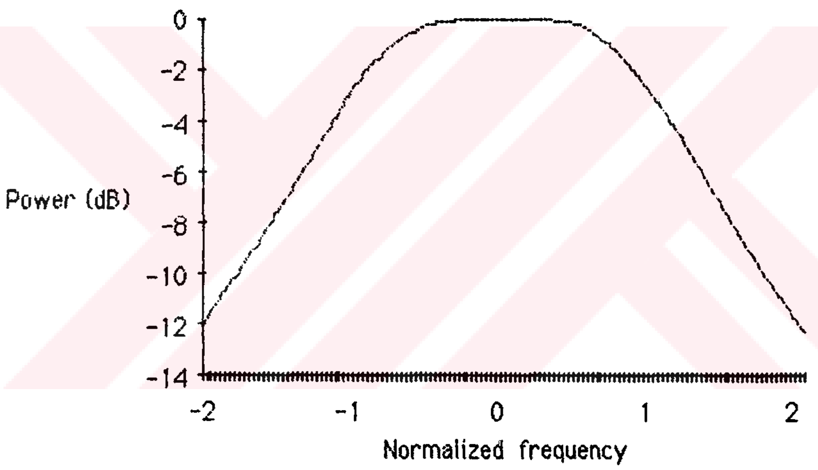

Figure 4.13 Power spectrum of 2nd order Butterworth filter [13]

The magnitude response of a 2nd order Butterworth filter is shown in figure 4.13. The gain at the center frequency is unity and the 3-(dB) cut-off frequency is at $\omega=1$. The high frequency roll-off of an n'th order Butterworth filter magnitude square is 20n dB/decade. The poles of Butterworth filters are located equally spaced on a circle in the s-plane [13]. In the figure 4.14 the poles of a 3rd order Butterworth filter are shown.

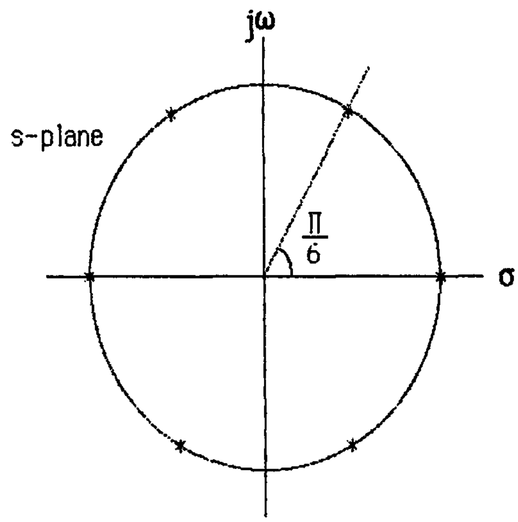

Figure 4.14 Pole locations of a 3rd order Butterworth filter [13]

The transfer function of these filters can be obtained by using the LHP poles; namely for an M'th order Butterworth filter the transfer function is given as in equation (4.23) where σ_i and ω_i define the pole location on s-plane.The angles of the LHP poles are given by $(2i+M-1)\pi/(2M)$ where $i=1,2,3,\ldots,M$

$$H(j\omega) = \prod_{i=1}^{M} \frac{1}{j\omega - (\sigma_i + j\omega_i)} \qquad (4.23)$$

4.2.1) Chebychev Filters:

These are IIR filters which have ripples in the passband of their spectrums and show monotonically decreasing behaviour in the transition- and stopband. The squared magnitude response of a 3rd order Chebychev filter with 3dB ripple amplitude is given in figure 4.15.

A Chebychev filter is defined by three parameters; the critical frequency, ω_c; the order N; and the passband ripple amplitude A_{max}.The number of ripples in the passband is equal to the filter order. When A_{max} is the peak-to-peak passband ripple given in dB, the ripple parameter ε is obtained as:

$$\varepsilon = \sqrt{10^{(A_{max}/10)-1}} \tag{4.24}$$

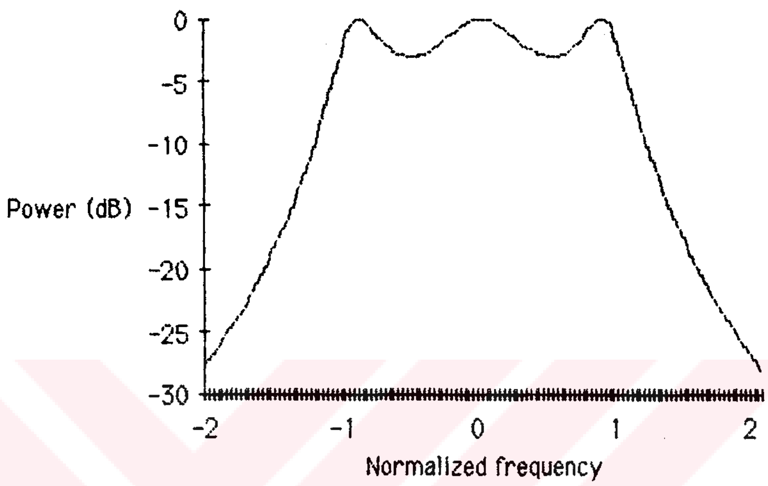

Figure 4.15 Power spectrum of 3rd order Chebychev filter with 3dB ripple [13]

The poles of a Chebychev filter lie on an ellipse in s-plane. Refering to figure 4.16 the ellipse is defined by two circles corresponding to minor axis and major axis of the ellipse [3]. The radius of the minor axis is $a\omega_c$ where :

$$a = \frac{1}{2}\left[\alpha^{1/N} - \alpha^{-1/N}\right] \tag{4.25}$$

with :

$$\alpha = \varepsilon^{-1} + \sqrt{1+\varepsilon^{-2}} \tag{4.26}$$

The radius of the major axis is $b\omega_c$ where :

$$b = \frac{1}{2}\left[\alpha^{1/N} + \alpha^{-1/N}\right] \tag{4.27}$$

To locate the poles of the Chebychev filter on ellipse we identify N angles as in the Butterworth case $(2i+N-1)\pi/(2N)$ where $i=1,2,..N$

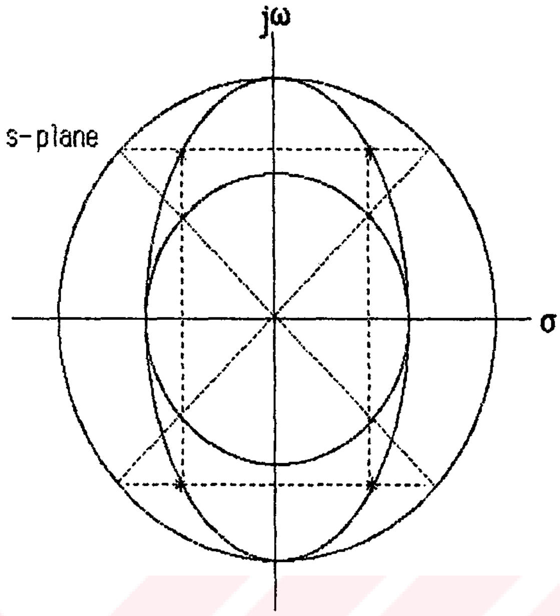

Figure 4.16 Pole locations of a 2nd order Chebychev filter [13]

The poles of a Chebychev filter fall on the ellipse with the ordinate specified by the points identified on the outer circle and the abscisca specified by the inner circle. The resulting square magnitude function will be :

$$|H(j\omega)|^2 = \frac{1}{1+\varepsilon^2 V_N^2(\omega/\omega_c)} \qquad (4.28)$$

where $V_N(x)$ is the N'th order Chebychev polynomial defined as :

$$V_N(x) = \cos(N\cos^{-1}(x)) \qquad (4.29)$$

V. INTERFERENCE INTO DIGITAL SIGNALS

The effects of interfering sources on digital communication systems are of great interest since interference is among the major causes of performance degradations. Because of the inherent nonlinear nature of digital systems, there is no formal solution to this problem. The performance of the digital system depends upon every detail of its design on environment, interference being only one aspect.

Thus effect of interference cannot be explicitly defined as can be done (in most instances) for analog systems. For an existing system design which produces a certain degradation D_1 (at a given BER) without interference and a degradation D_2 with interference, it is fair to say that D_2-D_1 is degradation that can be attributed to the presence of interference with the given set of conditions. If many sources of impairment are present, it becomes more difficult to extract one effect from another. In this case it is perhaps possible to show a generalized method to obtain numerical methods. Here we will consider some numerical approaches including exact and bounding techniques, with their application to coherent phase shift keying (CPSK) systems.

5.1 General Formulation, CPSK :

An M-ary CPSK system which is subject to intersymbol, interchannel and cochannel interferences and AWGN can be modelled as in figure 5.1. The complex envelope of the desired signal at the receiver input is given by equation (5.1) [14].

$$e_1(t)=\sum_{k=-\infty}^{\infty} p(t-kT)\exp\left[ja_k\theta(t-kT)\right] \qquad (5.1)$$

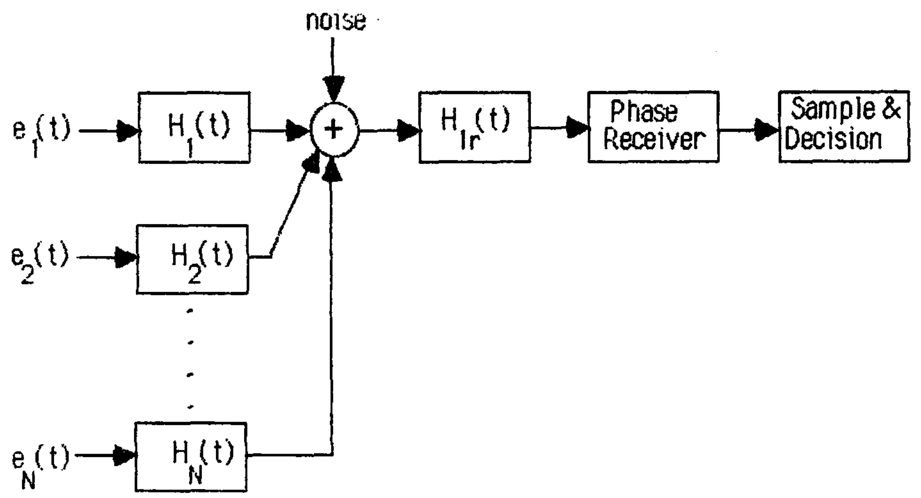

Figure 5.1 Block diagram of linear interference problem [14]

where

p(t) : possible amplitude shaping function

θ(t) : possible phase pulse shaping function

a_k : k'th symbol phase

For an M-ary system, a_k is usually chosen in the set [2πn/M] for n=1,2, . . . , M. The i'th interfering complex LP-equivalent signal can be represented by equation

$$e_i(t)= R_i(t) \exp\left[j\left[2\pi f_i^{'} t + \psi_i(t) + \mu_i\right]\right] \qquad i=2, 3, \ldots ,N \qquad (5.2)$$

where

$R_i(t)$: envelope of the i'th interfering signal

$\psi_i(t)$: phase of the i'th interfering signal

$f_i^{'}$: the frequency difference of the interfering carrier from the desired signal

μ_i : phase angle of the i'th interfering signal assumed to be independent of one another and uniformly distributed on (0,2π)

When the interfering signals are digital modulation signals, their form will be :

$$e_i(t)=\sum_{k_i=-\infty}^{\infty} r_i\, p_i(t-k_iT_i-\tau_i)\, \exp\left[j\left[2\pi f_i't + a_{ik_i}\psi_i(t-k_iT_i-\tau_i)+\mu_i\right]\right] \quad (5.3)$$

where

T_i : symbol duration

τ_i : relative time origin

r_i : relative interference level

a_{ik_i} : possible symbol phases of interfering signals

Each signal is passed through a filter with LP-equivalent transfer function $H_i(f)$. The receiver is assumed to be an ideal phase receiver that, once per symbol, samples the instantaneous phase β and decides that $a_k=(2\pi k/M)$ was sent if:

$$\frac{(2k-1)\pi}{M} \le \beta \le \frac{(2k+1)\pi}{M}$$

The complex envelope of the input to the phase detector is :

$$e(t)= n_c(t) + jn_s(t) + \sum_{i=1}^{N} e_i(t)*h_i(t) \quad (5.4)$$

$n_c(t)$ and $n_s(t)$ are the in-phase and quadrature components of the noise. Suppose the zero'th symbol a_0 is to be detected. The decision variable will be :

$$\beta_0 \equiv \beta(t_0) = \tan^{-1}\left(\frac{e_s(t_0)}{e_c(t_0)}\right)$$

where $e_c(t)$ and $e_s(t)$ are the real and imaginary parts of e(t), given as follows:

$$e_c(t_0) = s_{c0}+ n_{c0}+ x_{c0}+ y_{c0}$$

$$e_s(t_0) = s_{s0} + n_{s0} + x_{s0} + y_{s0}$$

where

- s : useful signal
- n : noise
- x : intersymbol interference
- y : interchannel interference

Let

$$C_k(t) \equiv p(t-kT)\cos(a_k\theta(t-kT))$$

$$S_k(t) \equiv p(t-kT)\sin(a_k\theta(t-kT))$$

$$A_i(t) \equiv R_i(t)\cos(w_i't + \psi_i(t) + \mu_i)$$

$$B_i(t) \equiv R_i(t)\sin(w_i't + \psi_i(t) + \mu_i)$$

$$h_i(t) \equiv h_{ic}(t) + jh_{is}(t)$$

Then :

$$s_c(t) = C_0(t)*h_{1c}(t) - S_0(t)*h_{1s}(t) \quad (5.5a)$$

$$s_s(t) = C_0(t)*h_{1s}(t) + S_0(t)*h_{1c}(t) \quad (5.5b)$$

$$x_c(t) = \sum_{k\neq 0} C_k(t)*h_{1c}(t) - S_k(t)*h_{1s}(t) \quad (5.5c)$$

$$x_s(t) = \sum_{k\neq 0} C_k(t)*h_{1s}(t) + S_k(t)*h_{1c}(t) \quad (5.5d)$$

$$y_c(t) = \sum_{i=2}^{N} A_i(t)*h_{1c}(t) - B_i(t)*h_{1s}(t) \quad (5.5e)$$

$$y_s(t) = \sum_{i=2}^{N} A_i(t)*h_{1s}(t) + B_i(t)*h_{1c}(t) \quad (5.5f)$$

If the interfering signals are digital modulation signals, equations (5.5e) and (5.5f) become similar to (5.5c) and (5.5d). Thus, the external interference problem becomes formally similar to ISI, making techniques developed for the latter possible to use for the former.

In order not to bother with the heavy notation of M-ary system, we will deal with binary systems and note that for the M-ary system, the symbol error probability is bounded to within a factor of 2 by that of the binary system [15]. For a binary system the error probability, with the noise being a white Gaussian process of variance σ^2, is given by equation (5.6) :

$$P_2 = \frac{1}{2} E\left[Q\left(\frac{s_1 + x_{c0} + y_{c0}}{\sigma}\right) + Q\left(\frac{-s_2 - x_{c0} - y_{c0}}{\sigma}\right)\right] \qquad (5.6)$$

where

s_1 : s_{c0} given $a_0=0$

s_2 : s_{c0} given $a_0=\pi$

Q : cumulative Gaussian distribution function

E : expectation over x_{c0} , y_{c0}

Thus the computational problem reduces to a conditional expectation of Q(·). The different methods attacking to this problem constitute the difference in various approaches.

5.2 Numerical Methods For Interference Calculations

The problem can be expressed as to evaluate a term

$$I = E\left[Q\left(\frac{s+u}{\sigma}\right)\right] \qquad (5.7a)$$

or equivalently

$$I = \int_{-\infty}^{\infty} Q\left(\frac{s+u}{\sigma}\right) f(u)\, du \qquad (5.7b)$$

u represents the probabilistic interference terms and f(u) its pdf. Still another equivalent form, using the characteristic function can be obtained, which is :

$$I = \frac{1}{2\pi}\int_{-\infty}^{0}\int_{-\infty}^{\infty} \exp\left(-\frac{1}{2}\sigma^2 v^2\right)\exp\left[-jv(x-s)\right] dv\, dx \qquad (5.7c)$$

where x and s represent the interference and the useful signal respectively.

5.2.1) Series Methods:

These approaches either start from (5.7a) and are based on a power series expansion of Q(·) function or start from (5.7c) and consist of expanding the characteristic function. When the characteristic function expansion method is used, some fraction of interference power (Δ) is assigned to noise power and a reduced interference source is obtained, to speed up the convergence of the series [16]. But no constructive method to find Δ is available.

5.2.2) Gaussian Quadrature Rules:

This approach approximates the integral in (5.7b) as:

$$I \approx \sum_{i=1}^{L} w_i\, Q\left(\frac{s+u_i}{\sigma}\right) \qquad (5.8)$$

The set of pairs (w_i,u_i) is called a quadrature rule and can be derived from the first 2L+1 moments of u [15], [17]. The fact that,it has the best convergence specifications among the methods of this type [17], and it is easily applicable to many situations, makes this approach more advantageous than the other ones. A method to obtain the moments of interference is presented in appendix A. Appendix B gives the description of a method to find quadrature rules.

5.2.3) Direct Averaging Method:

This approach is based on a per letter evaluation of equation (5.7b).This requires an explicit representation of pdf of interference. In general this is practically impossible. In special cases, f(u) is available in particular when the interfering signals are all angle modulated.

5.3 Bounding Approaches

Because of the numerical complexity of the "exact" formulation, bounds which are perhaps less accurate but definitely easier to compute are proposed. Two of these bounding techniques are given below.

5.3.1) Chernoff Bound:

The point of departure of Chernoff bound is the inequality $P_e \leq e^{g(\lambda)}$ where $g(\lambda)=\ln E(e^{\lambda V})$. λ is any positive integer and V is the decision variable. Although this upper bound is tighter than the worst case bound, its tightness decreases with increasing interference power [7].

5.3.2) Moment Space Bounds:

These are bounds obtained via an isomorphism theorem, from the theory of moment spaces. These upper and lower bounds are seen to be equivalent to upper and lower envelopes of some compact convex body generated from a set of kernel functions. The proposed method of the original paper [18] which takes only ISI into account, can be extended to include other interference effects by evaluating the moments of those interfering symbols using the technique which is given in appendix A. The tightness of the obtained bounds and the rapid convergence specifications make this method an interesting research subject.

VI. NONLINEAR CHANNELS

6.1 Modeling of Nonlinearities in Simulations

Practical communication systems include nonlinear elements. Typical nonlinear elements are amplifiers. Their nonlinear behaviour becomes dominant when they are operated so as to extract the maximum power they are capable of delivering. This is just the case with satellite communications. The travelling wave tube (TWT) amplifiers at satellite transponders are operated at their maximum power output operating points. The nonlinearities can be modelled as in figure 6.1.

Figure 6.1 Modelling of a nonlinearity with memory [19]

The zero memory nonlinear device (ZMNL) is sandwiched between two narrowband filters $H_1(f)$ and $H_2(f)$. The ZMNL can exhibit two kinds of nonlinear distortion effects on its input signal:

1) A nonlinear output-input power characteristic (amplitude modulation to amplitude modulation or AM/AM conversion)

2) A nonlinear output phase-input power characteristic (amplitude modulation to phase modulation or AM/PM conversion)

Those effects can be seen, considering the input-output relation of a ZMNL device. Suppose for the time being that $h_{1comp}(t)= h_{2comp}(t)= \delta(t)$ where $\delta(t)$ is the dirac delta function, then the relation between input and output is of the system in figure 6.1 can be expressed as in equation (6.1):

$$y_{comp}(t) = g(|x_{comp}(t)|) \exp[\, j(f(x_{comp}(t))+\arg(x_{comp}(t))\,] \qquad (6.1)$$

Let

$$x_{comp}(t) = A \exp(j\theta) \tag{6.2}$$

then equation 6.1 can be rewritten as:

$$y_{comp}(t) = g(A) \exp[\, j(f(A)+\theta)\,] \tag{6.3}$$

In equation (6.3) g(A) represents the AM/AM conversion and f(A) the AM/PM conversion.

The inphase and quadrature channel representation of this nonlinearity is shown in figure 6.2 [20].

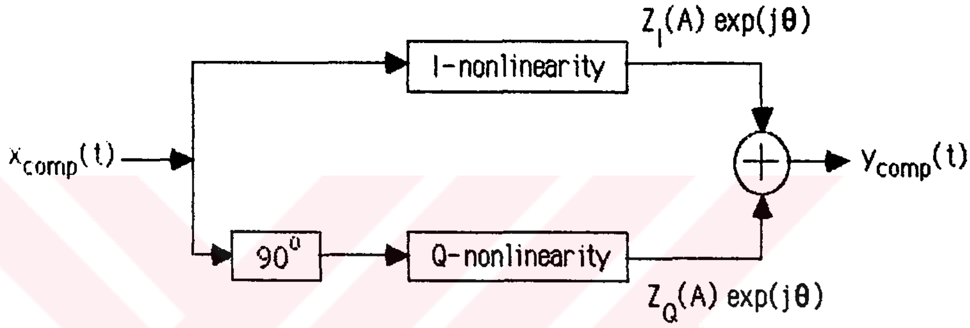

Figure 6.2 IQ-channel representation of nonlinearity [20]

Referring to figure 6.2 the input-output relation can be written as:

$$y_{comp}(t) = [\, Z_I(A) + jZ_Q(A)\,] \exp(j\theta) \tag{6.4}$$

The relation of $Z_I(A)$ and $Z_Q(A)$ with g(A) and f(A) are found to be:

$$g(A) = \sqrt{Z_I^2(A) + Z_Q^2(A)} \tag{6.5}$$

$$f(A) = \tan^{-1}\left[\frac{Z_Q(A)}{Z_I(A)}\right] \tag{6.6}$$

When $h_{1comp}(t)$ and $h_{2comp}(t)$ have colored Fourier spectrum the nonlinearity will attain memory. We used Butterworth filters for $h_{1comp}(t)$

and $h_{2comp}(t)$ in our simulation program. Usually $h_{2comp}(t)$ has smaller bandwidth than $h_{1comp}(t)$.

6.2 Nonlinearity Models

6.2.1) Bandpass Limiters:

The bandpass limiters introduce AM/AM conversion effects and have the input output characteristics shown in figure 6.3 [20].

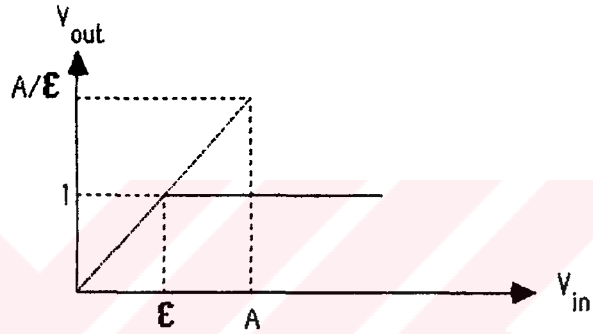

Figure 6.3 Limiter characteristics [20]

ε, in the figure is the clipping point. When ε=0 all of the amplitude information in the input signal is lost. Then the limiter is called to be the hard limiter. If a predetection bandpass limiter is introduced preceding the detection filter which is otherwise matched to the received signal, the performance of the digital signal will decrease. Such a situation could arise in a communication satellite link where the limiting takes place in a repeater that detects the signal before transmitting it.

6.2.2) TWT Amplifiers

The high frequency, large output TWT amplifiers, used in satellite communications, exhibit both AM/AM and AM/PM conversion effects. The single carrier AM/AM and AM/PM conversion effects of an INTELSAT IV TWT, which is also the TWT nonlinearity model of SSNDC, are shown in figures 6.4 and 6.5. For low input levels the output power is essentially a linear function of the input power. As the input power increases, the output power increases nonlinearly until a point is reached where any additional input level increase results in a decreassing output power. This point is called the saturation point. The operation point of a TWT is given as the input or output power relative to saturation or back-off. One definition of saturation is that 11 dB decrease in input power will result in 7 dB decrease in output power. To maximize the available power out of a TWT, it is operated near saturation.

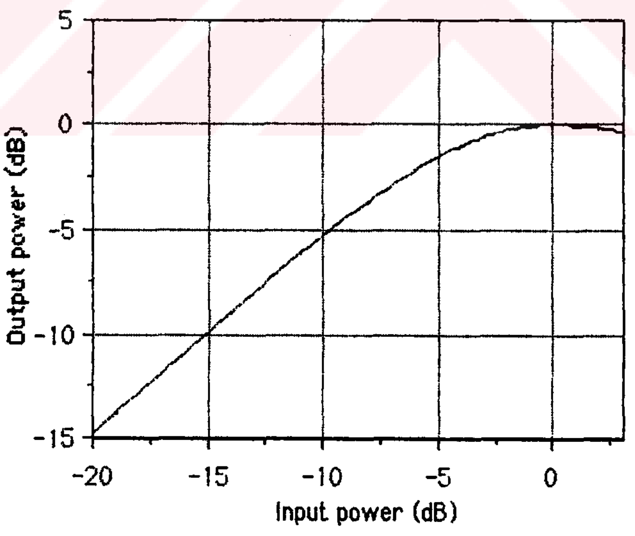

Figure 6.4 AM/AM characteristics of INTELSAT-IV TWT amplifier[21]

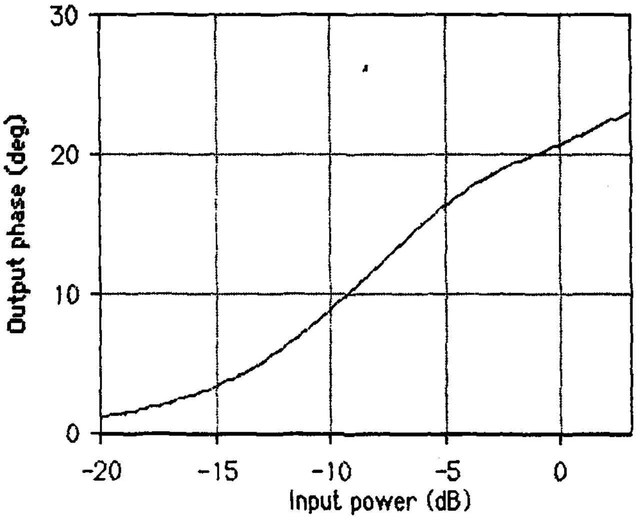

Figure 6.5 AM/PM characteristics of INTELSAT-IV TWT amplifier [21]

To simulate a TWT nonlinearity, samples of the quadrature curves are stored and used for specific input power, the outputs are obtained by interpolation.

Alternatively, approximations to these quadrature curves can be used on a "best fit" basis. For the INTELSAT IV TWT (Hughes Corp. 261H tube) these two envelope nonlinearities are given by a least square fit.

$$Z_I(A) = C_1 A\, e^{-C_2 A^2}\, I_0(C_2 A^2) \qquad (6.7)$$

$$Z_Q(A) = S_1 A\, e^{-S_2 A^2}\, I_1(S_2 A^2) \qquad (6.8)$$

where

I_0 : modified Bessel function of zero'th order

I_1 : modified Bessel function of first order

$C_1=1.61245$

$S_1=1.71850$

$C_2=0.53557$

$S_2=0.242218$

Polynomial approximations could be used as well but more coefficients are required.

6.3 Volterra Series Representation of Nonlinearities

For nonlinearities that have memory, the Volterra series approach is appealing due to its generality and its clear relationship to a linear system impulse response. A Volterra series is a Taylor series with memory described by [19]:

$$Y(t) = \sum_{n=1}^{\infty} Y_n(t) \qquad (6.9)$$

Here the system is assumed to have no constant (d.c.) response. Each term of order n is described by an n-fold convolution as:

$$Y_n(t) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_n(T_1, T_2, \ldots, T_n)\, x(t-T_1) \ldots x(t-T_n)\, dT_1 \ldots dT_n \qquad (6.10)$$

The first order term is the usual linear system response. An immediate problem with Volterra series is the justification of truncating the series at some order. The other problem is the complexity of finding the Volterra kernels denoted by $h_n(T_1, T_2, \ldots, T_n)$ in equation (6.10). These problems make a Volterra series model impractical for all but a very few applications in which the complexity and cost can be justified.

As an application of the Volterra model, consider the simplified block diagram of a digital satellite link with complex lowpass representation, shown in figure 6.6 [22] where (a_n) is the sequence of discrete independently identically distributed generally complex random variables. x(t) is the modulated signal:

$$x(t) = \sum_n a_n \, \delta(t-nT) \qquad (6.11)$$

s(t) is the overall impulse response of the linear filters preceding the nonlinearity. c(.) is a ZMNL device with:

$$c(.) = g(.) \exp(jf(.)) \qquad (6.12)$$

g(.) and f(.) are defined as in equation (6.3). u(t) is the impulse response of the filters following the nonlinearity. $n_1(t)$ and $n_2(t)$ are generally complex baseband Gaussian processes with zero mean and variances σ_1^2 and σ_2^2 (representing uplink and downlink noises respectively). First we will assume that $n_1(t) = 0$. Following a few straightforward steps, the relation between y(t) and x(t) can be expressed as in equation (6.13) (a complete description is given in [22] and [23]).

$$y(t) = \sum_{m=0}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_{2m+1}(T_1, \ldots, T_{2m+1}) \prod_{i=1}^{m+1} x(t-T_i) \prod_{l=m+2}^{2m+1} x^*(t-T_l) \, dT_1 \ldots dT_{2m+1} \qquad (6.13)$$

Assume that the ZMNL can be represented by a Taylor series expansion:

$$c(A) = \sum_{m=0}^{\infty} \gamma_{2m+1} A^{2m+1} \qquad (6.14)$$

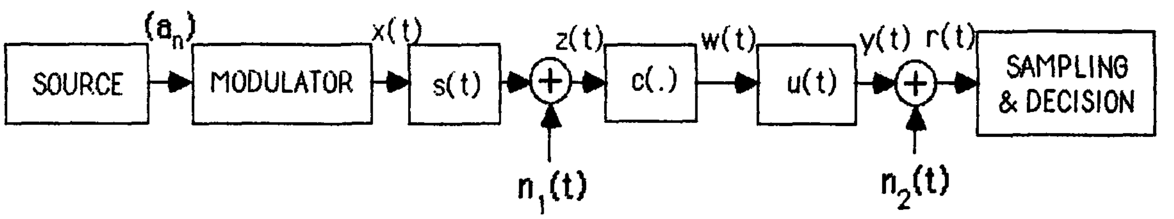

Figure 6.6 Equivalent block diagram of a digital satellite link [22]

Only odd terms are present in equation (6.14) due to the bandpass nature of the nonlinearities. Using (6.14) the output of ZMNL can be represented as:

$$w(t)=\sum_{m=0}^{\infty}\gamma_{2m+1}z^{m+1}(t)\,z^{*m}(t) \tag{6.15}$$

Since:

$$z(t)=\int_{-\infty}^{\infty}s(T)\,x(t-T)\,dT \tag{6.16}$$

and:

$$y(t)=\int_{-\infty}^{\infty}u(T)\,w(t-T)\,dT \tag{6.17}$$

After some straight forward steps, the following expression is obtained:

$$y(t)=\sum_{m=0}^{\infty}\gamma_{2m+1}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}u(T)\prod_{r=1}^{m+1}s(T_r-T)\prod_{s=m+2}^{2m+1}s^*(T_s-T)$$
$$\prod_{i=1}^{m+1}x(t-T_i)\prod_{l=m+2}^{2m+1}x^*(t-T_l)\,dT\,dT_1\ldots dT_{2m+1} \tag{6.18}$$

Comparing (6.18) with (6.13) one gets the low pass equivalent kernels:

$$h_{2m+1}(T_1,\ldots,T_{2m+1})=\gamma_{2m+1}\int_{-\infty}^{\infty}u(T)\prod_{r=1}^{m+1}s(T_r-T)\prod_{s=m+2}^{2m+1}s^*(T_s-T)\,dT \tag{6.19}$$

The received signal r(t) is given by:

$$r(t)=y(t)+n_2(t) \tag{6.20}$$

Let y(t) be sampled at time $t=t_0$ and defining:

$$H_{2m+1}(n_1,\ldots,n_{2m+1})=h_{2m+1}(t_0-n_1T_{symbol},\ldots,t_0-n_{2m+1}T_{symbol}) \tag{6.21}$$

$$N_2 = n_2(t_0) \tag{6.22}$$

$$R = r(t_0) \tag{6.23}$$

Remembering that x(t) is given in equation (6.11) one can write R as follows:

$$R = \sum_{m=0}^{\infty} \sum_{n_1=-\infty}^{\infty} \cdots \sum_{n_{2m+1}=-\infty}^{\infty} a_{n_1} \cdots a_{n_{m+1}} a^*_{n_{m+2}} \cdots a^*_{n_{2m+1}} H_{2m+1}(n_1, \ldots, n_{2m+1}) + N_2$$

(6.24)

The decision device operates on samples of the in-phase and quadrature components of R. From (6.24) we can extract all the terms containing only the transmitted symbol a_0 which contributes to form the useful sample R_0:

$$R_0 = R_{0P} + jR_{0Q} = a_0 \sum_{m=0}^{m_M} |a_0|^{2m} H_{2m+1}(0,0,\ldots,0) \tag{6.25}$$

m_M in (6.25) is asuitable number to stop the summation which is found to be 3 for TWT nonlinearity. Letting $P=R-R_0$ equation (6.24) becomes:

$$R=(R_{0P}+P_P+N_{2P}) + j\,(R_{0Q}+P_Q+N_{2Q}) \tag{6.26}$$

The error probability can be evaluated in a similar manner as in the case with linear interference problem discussed in chapter 6. The complete description of this method is given in [23].

For nonlinear digital communication systems which contain one nonlinear element, the results of this section can be utilized to obtain symbol error probability curves in a shorter time compared to a simulation.

VII. TECHNIQUES OF BER ESTIMATION

The definition of digital links performance commonly used is the bit error rate (BER), or the bit error probability. To arrive at an estimate of BER basically two different approaches exist. The first one, which we might refer as analytical, is strictly based upon manipulation of equations. It is still computer-aided, however as closed-form solutions are not available. The advantage of these approaches is their speed and their disadvantage is the analytical intractability when the system under examination gets more complex. Even for the nonlinear systems analytical methods exist, [22], [23] but they seem to be limited with very particular cases.

The second class of approches are simulation-based which may be further divided into the following groups [24]:

a) Monte Carlo (M.C.) simulation

b) modified M.C. simulation also referred to as importance sampling

c) extreme value theory (classical and generalized)

d) tail extrapolation

e) hybrid simulation/analysis (quasi-analytical)

Before discussing these methods, it is useful to mention the decision process shortly. The decision process can be described in terms of the probability density functions (pdf), $f_0(Q)$ and $f_1(Q)$, of the input voltage at the sampling instant, given that a "zero" or a "one" is sent respectively. These densities are sketched in figure 7.1.

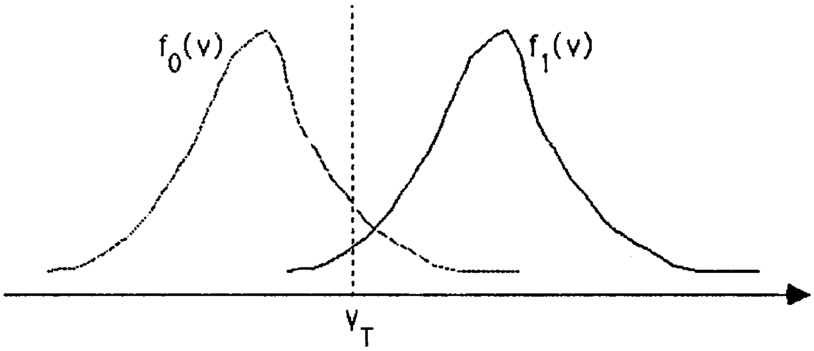

Figure 7.1. Hypothetical probability density functions [24]

For a simple threshold-sensing decision device an error will occur when a "zero" is sent and the voltage at the input of the decision device exceeds threshhold voltage, V_T or a "one" is sent and disturbances cause the voltage to drop below V_T. These probabilities are given as follows:

$$\text{Prob [error/one]} \equiv p_1 = \int_{-\infty}^{V_T} f_1(v)\,dv = F_1(V_T) \qquad (7.1a)$$

$$\text{Prob [error/zero]} \equiv p_0 = \int_{V_T}^{\infty} f_0(v)\,dv = 1 - F_0(V_T) \qquad (7.1b)$$

The average probability is then

$P = \text{Prob[one]}\, p_1 + \text{Prob[zero]}\, p_0$

The functions $F_0(.)$ and $F_1(.)$ are evidently the cumulative distribution functions (CDF). Generally one may make assumptions or deal directly with the CDF or pdf depending upon the estimation technique. In either case, it is only a small region of these functions, namely "the tails", that we are interested in.

The M.C. method makes no a priori assumptions and in that sense, it is the most general of the techniques. It supplies and empirical determination of distribution functions evaluated at a single point. Because it is the most general, it is the computationally most costly of these methods. This cost is related to the number of observations for a reliable estimate of BER that we are interested in.

Some of the techniques listed before are applicable to the simulation, while others are limited to monitoring cases. A basic distinction between these two cases is that monitoring implies lack of knowledge of the actual transmitted sequence while reverse is true for simulation. Hence, methods c and d are applicable to monitoring. The MC method, of course emulates the conventional laboratory BER measurement method, using a known transmitted sequence. It is not suitable for monitoring unless, the operational environment provides for periodic sequences. Methods b and e are not suitable for a physical counterpart.

Since we are primarily concerned with simulation, we will discuss the methods a, b, and e.

7.1 Monte Carlo Method

Let us assume that a "zero" is sent, so that 1 b) applies. Then:

$$p_0 = \int_{-\infty}^{\infty} h_0(v)\, f_0(v)\, dv \qquad (7.2)$$

Where

$$h_0(v) = \begin{cases} 1 & v \geq v_T \\ 0 & v < v_T \end{cases}$$

A natural estimator p_0 is the sample mean

$$\hat{p}_0 = \frac{1}{N} \sum_{i=1}^{N} h_0(v_i) \qquad (7.3)$$

If N bits are processed through the system, out of which n are observed to be in error, a simple unbiased estimator of the BER is the sample mean

$$\hat{p} = \frac{n}{N} \tag{7.4}$$

In the limit when $N \Rightarrow \infty$, p will tend to true value p. For finite N, we quantify the reliability of the estimator in terms of confidence intervals. Two numbers h_1 and h_2 are searched, functions of p, such that for given high error probability, $h_2 \le p \le h_1$ and the confidence interval $h_1 - h_2$ be as small as possible. The confidence level, $1-\alpha$ is defined through the relation

$$\text{Prob}\,[\, h_2 \le p \le h_1] = 1-\alpha \tag{7.5}$$

The confidence levels for M.C. methods are shown in figure 7.2.

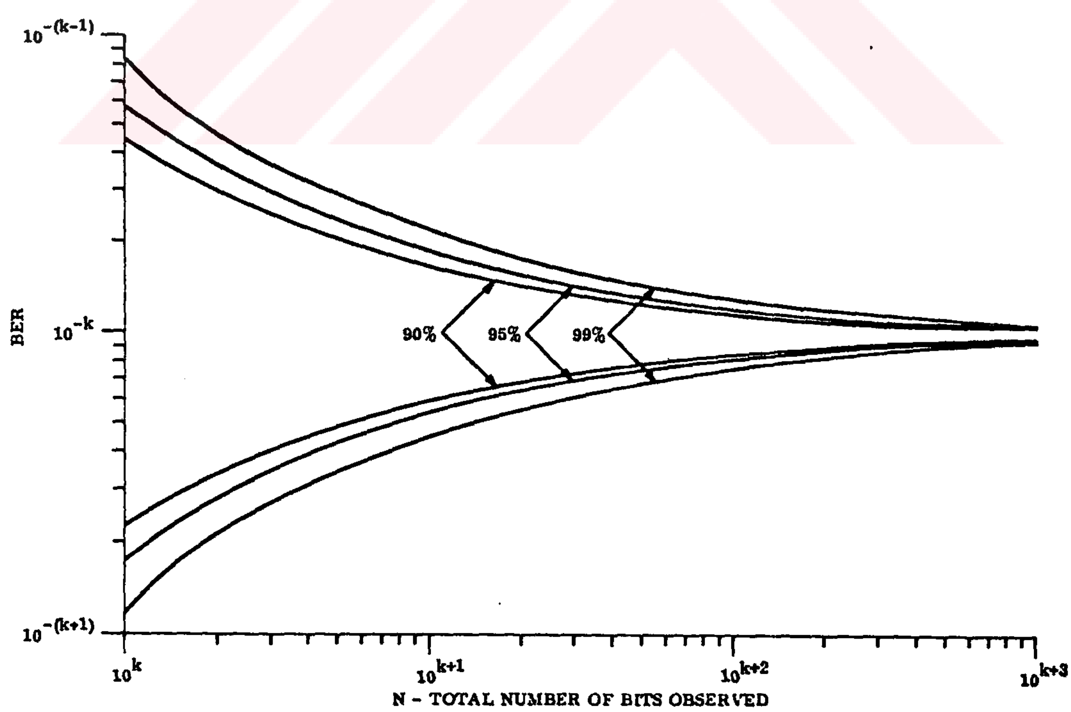

Figure 7.2 Confidence bands on BER when observed value is 10^{-k}

(Monte Carlo technique) [24]

7.2 Importance Sampling

The important events, namely errors are caused very rarely by the underlying noise processes. The simulation efficiency could be enhanced if errors could be made artificially to occur more often in an invertible way, such that the true BER could be obtained from the inflated one. This is the idea behind importance sampling. Let us modify the equation (7.2) as follows [13], [24] :

$$p_0 = \int_{-\infty}^{\infty} \left[h_0(v) \frac{f_0(0)}{f_0^*(v)} \right] f_0^*(v)\, dv \qquad (7.6)$$

f_0^* is another probability density function of the same type with $f_0(0)$, but with a higher variance. Denoting the term within brackets as $h_0^*(v)$, the new estimator is given by:

$$\hat{p}_0^* = \frac{1}{N^*} \sum_{i=1}^{N^*} h_0^*(v_i) \qquad (7.7)$$

7.3 Hybrid Simulation/Analysis

In this method the thermal noise is omitted and simulation is used only to obtain the statistics of all other sources of distortion and interference, The effect of thermal noise is then added analytically and the average error rate is calculated.

When the demodulation and detection process is nonlinear, direct simulation (Monte Carlo Method) is the only choice. Such cases include envelope detection of FSK signals. However with coherent phase shift keying

(CPSK) systems the hybrid simulation/analysis, also referred to as quasi-analytic method is applicable [25], [26].

The distinction between these two approaches is shown in figure 7.3 [25].

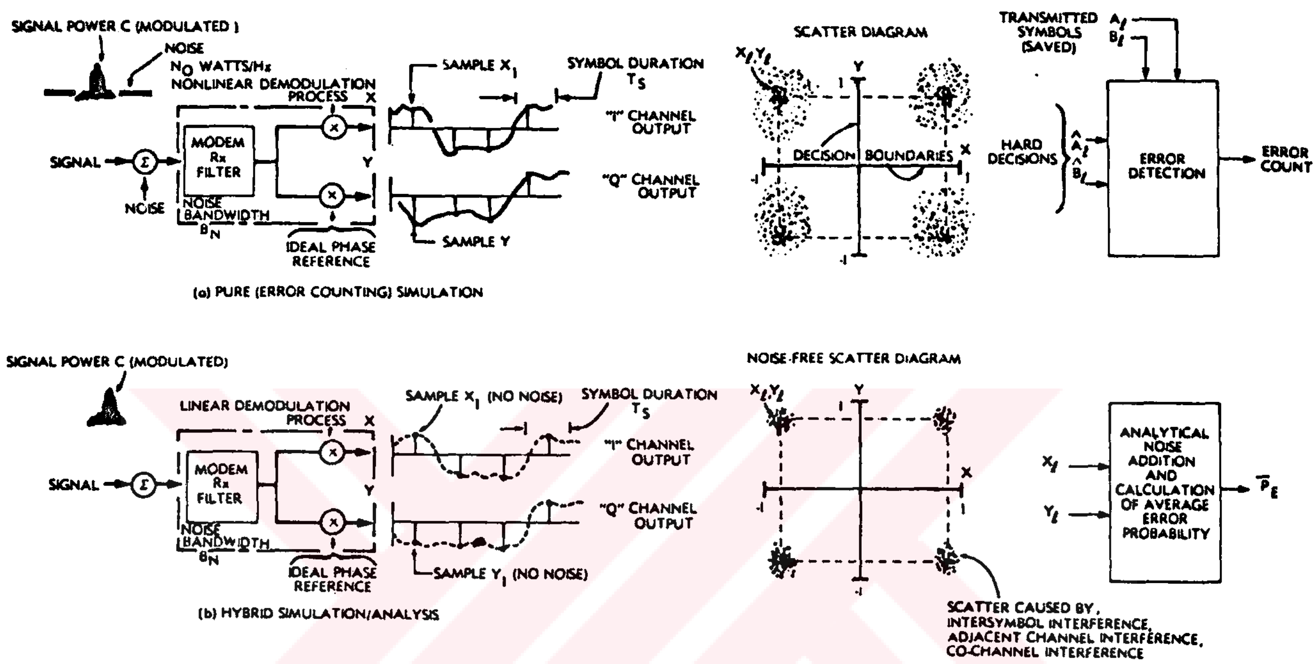

Figure 7.3 Pure Monte Carlo and quasi-analytic simulations [24]

The upper and lower models illustrate direct simulation and quasi-analytical methods, respectively. With quasi-analytical methods the average P_e is calculated as:

$$\overline{P(\mathcal{E})} = \frac{1}{N} \sum_{l=1}^{N} P(\mathcal{E}|X_l, Y_l) \qquad (7.8)$$

where N is the number of symbols in the simulation run and $P(\mathcal{E}|X_l,Y_l)$ is the error probability given a particular value of decision metric (X_l,Y_l) on the l'th symbol.

Referring to the QPSK model shown in the lower part of figure 7.3, the samples (X_l,Y_l) are plotted as a noise free scatter diagram. For a perfect channel, exhibiting no interference and other distortions, the points should lie on top of each other. Such a diagram should give ideal (theoretical) performance. For a single scatter point the conditional probability of bit error is given as:

$$P(\varepsilon|X_l,Y_l) = \frac{1}{2}\left[Q\left(\sqrt{\frac{2E_s X_l^2}{\bar{P}N_0 B_N T_S}}\right) + Q\left(\sqrt{\frac{2E_s Y_l^2}{\bar{P}N_0 B_N T_S}}\right)\right] \quad (7.9)$$

where:

N_0 : noise density at the input to the receiver

E_s : energy per QPSK symbol (E_s=$2E_b$)

$B_N T_s$: product of receive modem noise bandwidth and the symbol duration

$\bar{P}$ is the mean square value of detected samples given by:

$$\bar{P} = \frac{1}{N}\sum_{l=1}^{N}\left(X_l^2 + Y_l^2\right) \quad (7.10)$$

VIII. SOFTWARE SSNDC

8.1 Linear System Simulations

The software SSNDC is capable of performing linear transmission system simulations in a few seconds, relying on the hybrid simulation/analysis method discussed in chapter 6. Figure 8.1 shows the possible blocks in a linear system simulation.

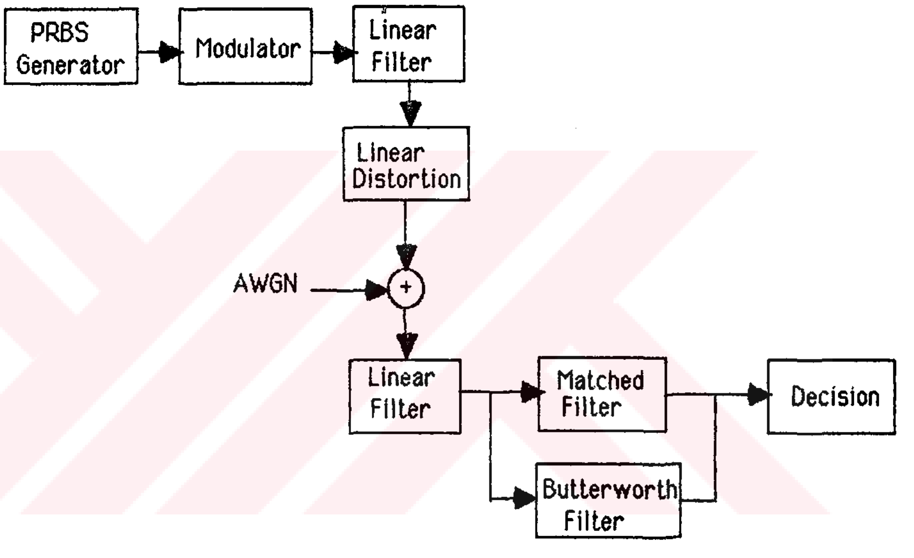

Figure 8.1 Linear transmission system of SSNDC

When the linear transmission system is being configured, the user is let to choose for each module from the available set of model library.

First the length of the symbol sequence and number of samples per a symbol is entered.

The available modulation types are:

a) BPSK

b) QPSK

c) OK-QPSK

d) MSK

The modulators are subject to the following realization imperfections:

a)amplitude imbalance

b)phase imbalance

The available linear transmitter (TX)-filters are:

a) Phase equalized Butterworth

b) Butterworth

c) Chebychev

These filters are reentrant. The linear TX-filter module in figure 8.1 (the linear filter succeeding the modulator) can be configured by a cascade combination of these filters. A cascade combination of two filters of the same type is also possible.

The available linear distortion models are:

a) Linear amplitude distortion

b) Parabolic amplitude distortion

c) Ripple amplitude distortion type I

d) Ripple amplitude distortion type II

e) Linear group delay

f) Parabolic group delay

g) Ripple group delay type I

h) Ripple group delay type II

The linear distortion module in figure 8.1 can be configured from a cascade combination of these filters, each one of them being included only once,since they are not reentrant.

The possible linear receiver (RX)-filters are same as linear TX-filters and the linear RX-filter module in figure 8.1 can be also configured by a cascade combination of these filters . The only difference is, that the noise power after passing this filter must be calculated.

There are four different demodulators matched to the signals at the corresponding modulator outputs. These demodulators are subject to the following realization imperfections:

a)static phase error

b)sampling time error

For BPSK systems a Butterworth detection filter can be utilized instead of matched filter.

8.2 Nonlinear System Simulations

The model for nonlinear system simulations by SSNDC is shown in figure 8.2.

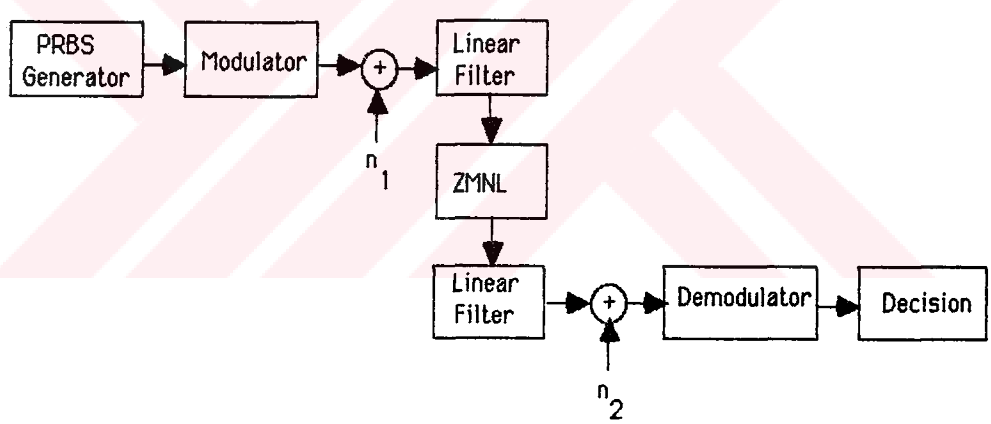

Figure 8.2 Nonlinear transmission system of SSNDC

Until the AWGN n_1 is added to the signal, this model follows the same steps as in the linear case. n_1 is generated by the Gaussnoise generating routine. Without changing the symbol sequence the simulation is repeated until the confidence level attains a satisfactory value.

The linear filters before and after ZMNL are chosen from the set in the linear system simulation. The possible ZMNL devices are:

a) Hard Limiter

b) Clipper

c) TWT

These nonlinear devices are generated according to the principles which are presented in chapter 6. The demodulators are matched to the corresponding modulator outputs.

8.3 System Configurator of SSNDC

The flow diagram of a system congiguration session with SSNDC is given in the following figures.

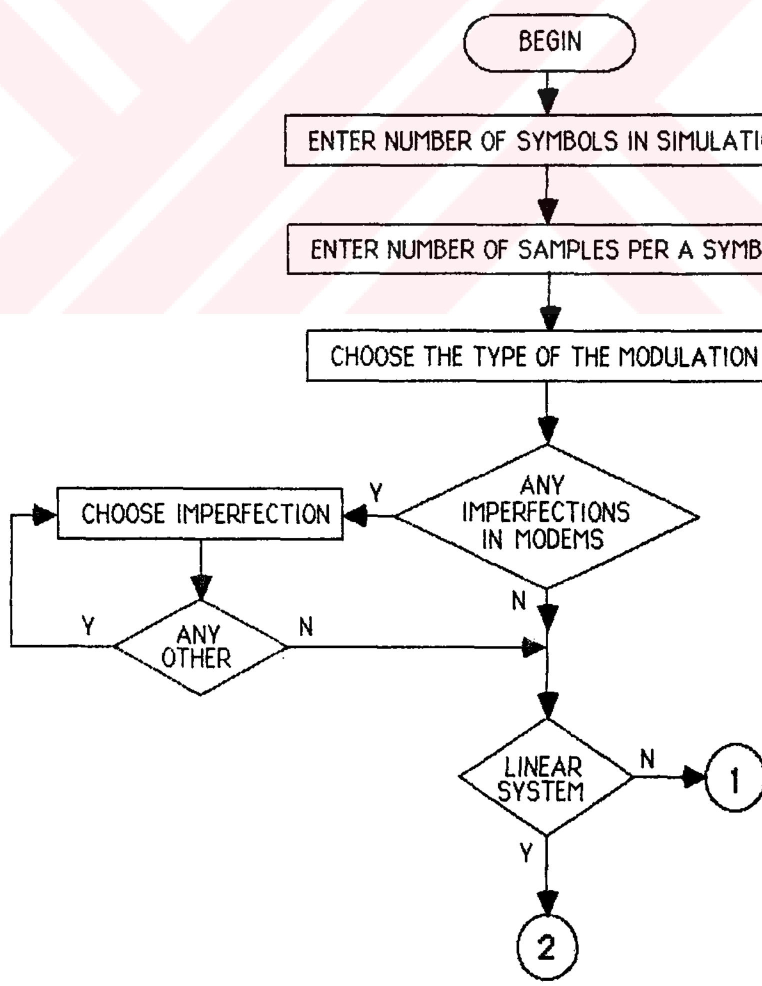

Figure 8.3 a) First part of system configuration session with SSNDC

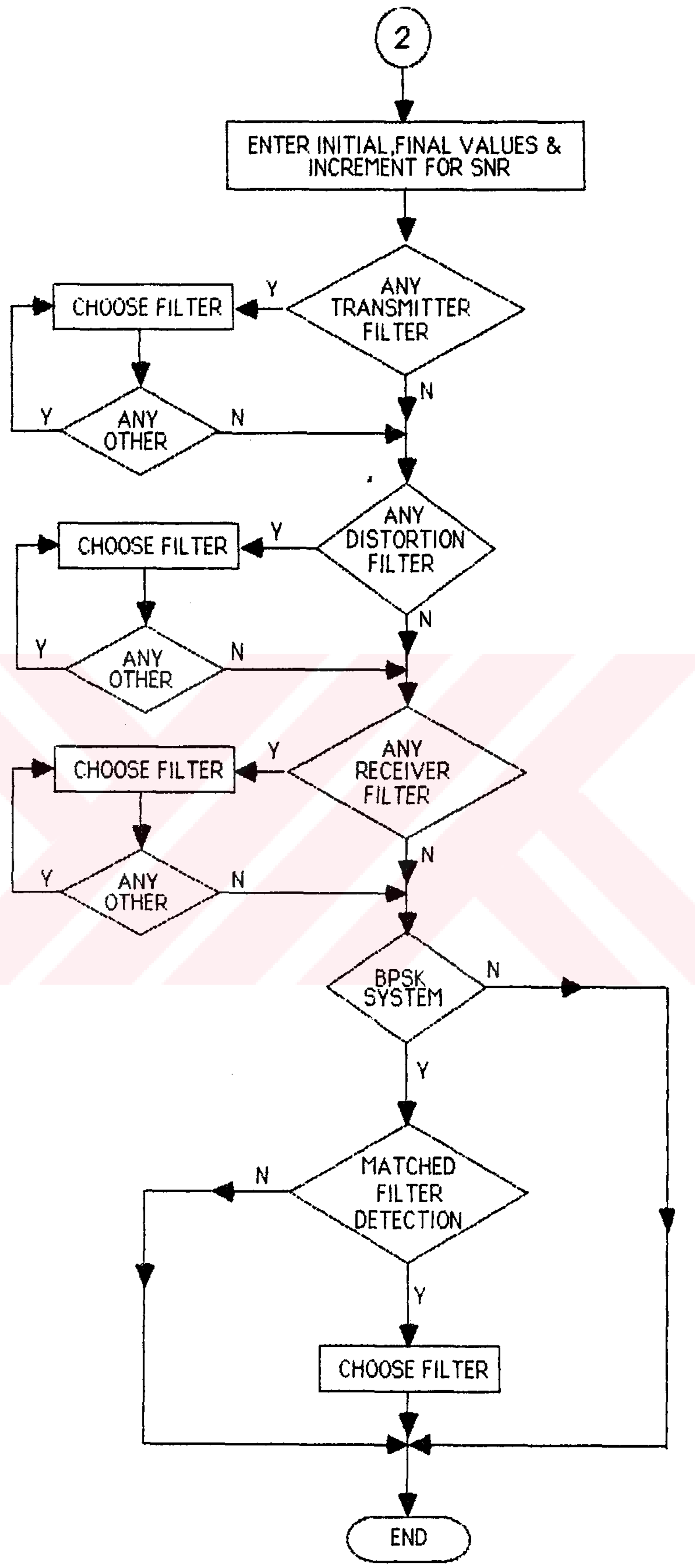

Figure 8.3 b) Second part of system configuration session with SSNDC (linear system)

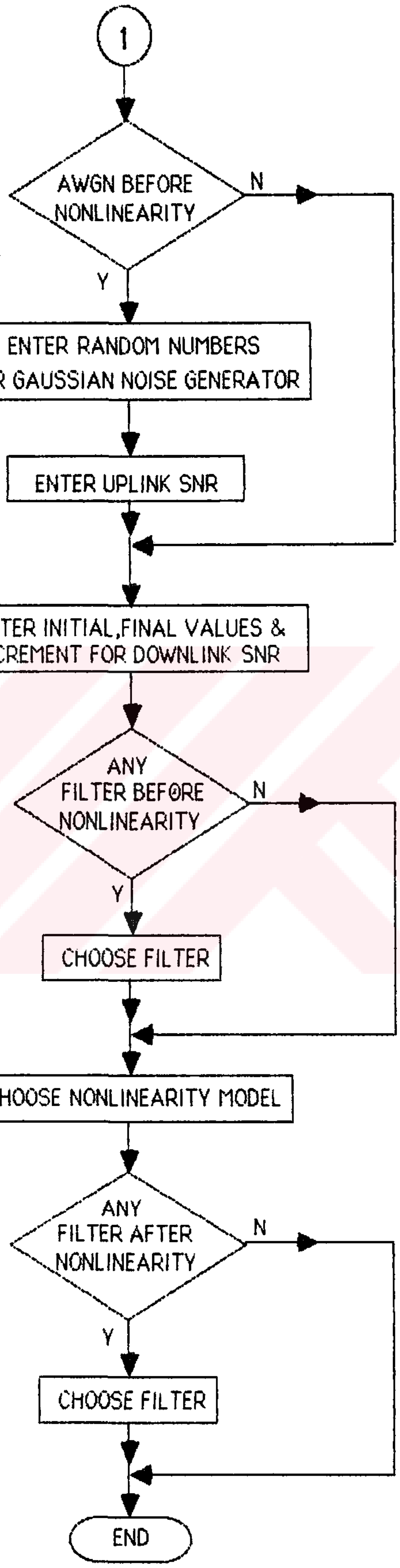

Figure 8.3 c) Second part of system configuration session with SSNDC (nonlinear system)

IX. RESULTS OBTAINED BY SSNDC

9.1 General

The results we have obtained by our simulation program, are in general similar to the ones given in [5] and [4]. For phase imbalances in BPSK and QPSK our results given in figures 9.1 and 9.4 are just the same as their counterparts in [4]. For amplitude imbalances we have found a different expression for degradation and our result matches to this expression given in equation (3.8). The effect of these two realization imperfections are found to be additive which can be justified, examining figure 9.3. The effect of amplitude imbalance in BPSK and QPSK are given in figures 9.2 and 9.5 respectively.

For linear delay distortions, the crosstalk introduced between the inphase and quadrature channels caused larger degradation in QPSK than in BPSK as discussed in chapter 4. , which is shown in figure 9.6. For quadratic delay distortions in BPSK and QPSK same amount of degradation is observed. But, introduction of a 0.1 dB ripple 3rd order Chebychev filter worsened the situation (figure 9.7), in contradiction to the results of [5] given in figure 4.2.

The effect of parabolic amplitude distortion in BPSK and QPSK are shown in figure 9.8.

The degradation caused by 0.1 dB ripple 5th order Chebychev filter is shown in figure 9.10. Our result show's a great amount of degradation. A relatively smaller degradation is observed when the signalling rate equals to the critical frequency of the filter (i.e. BT=1). This was not a surprise considering the increased matching between the signals and the filters frequency responses.

For imperfect demodulator structures, figures 9.9 and 9.11 are obtained which show the degradations due to demodulator static phase error and delayed sampling times for various modulation schemes. Our results are similar to the theoretical results. Observe that, in figure 9.9 for demodulator static phase error of 45 degrees (50 per cent), 3 dB degradation is resulted in agreement with the expression in equation (3.20). Figure 9.12 which shows the effect of sampling time error, is also in close agreement with the corresponding curves in figure 3.5 although only small sampling time shifts are considered in the later one.

Detection of BPSK with 2nd order Butterworth filters is considered in figure 9.11. Our result was quite similar to the results formerly obtained in [5] given in figure 3.6.

For nonlinear system simulations the OK-QPSK signals performance with TWT amplifier nonlinearity sandwiched between two Butterworth filters are considered. The probability of bit error curves for different values of uplink SNR are given in figure 9.13. For increased values of downlink SNR, the dominance of uplink AWGN and the resulting bottoming effect is observed. For TWT amplifier at saturation and with removed nonlinearity the systems performance did not change practically although with increased backoff the performance decreases.

9.2 Figures

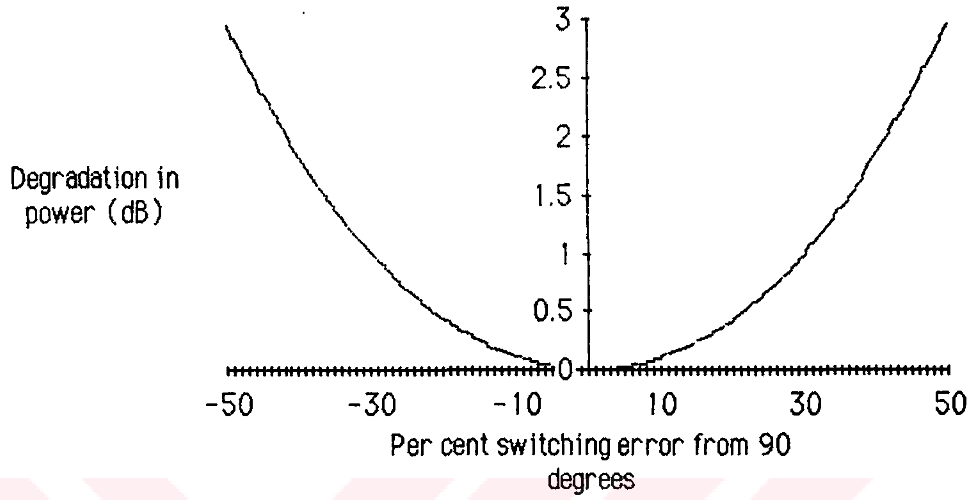

Figure 9.1 Degradation due to phase imbalance in BPSK

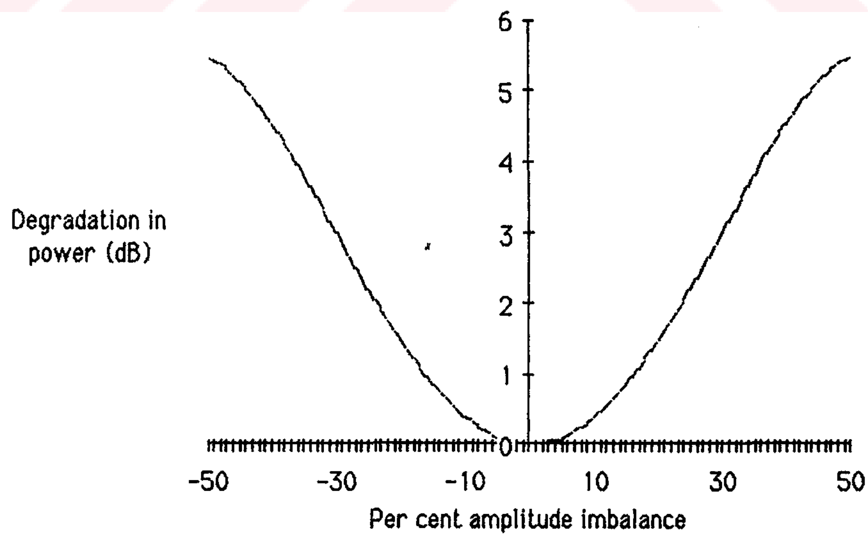

Figure 9.2 Degradation due to amplitude imbalance in BPSK

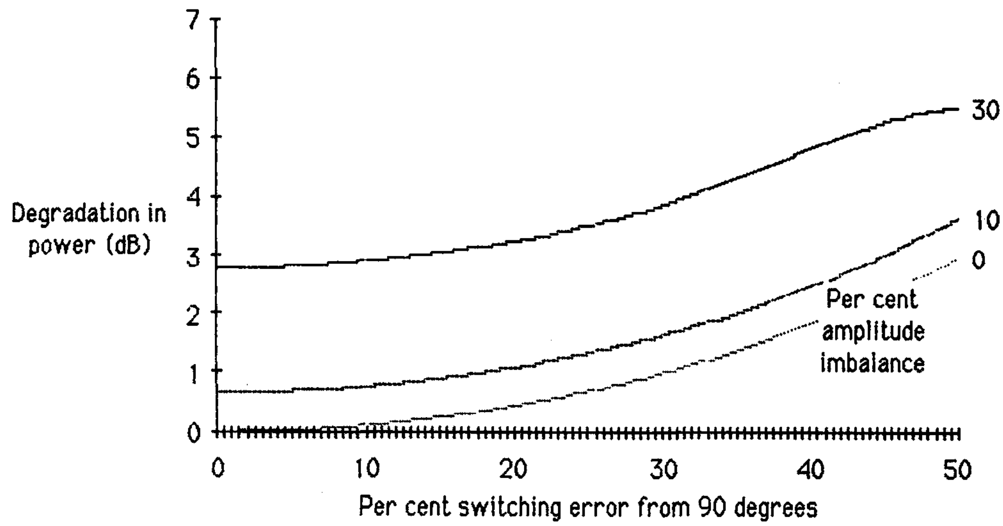

Figure 9.3 Degradation due to combined effects of phase and amplitude imbalance in BPSK

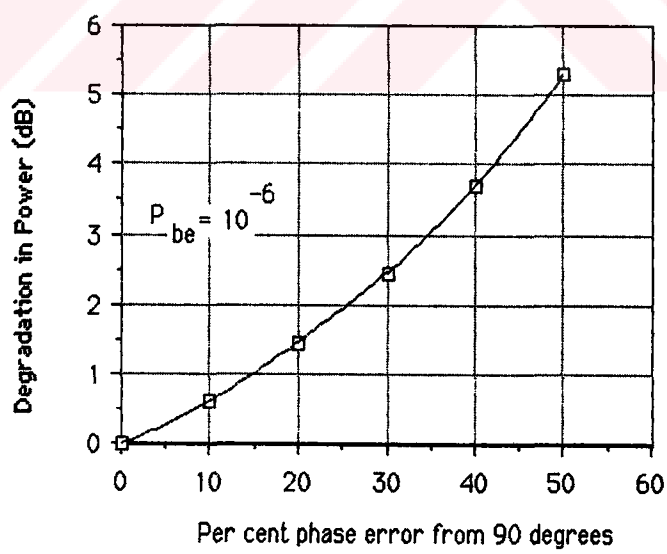

Figure 9.4 Degradation due to phase imbalance in QPSK

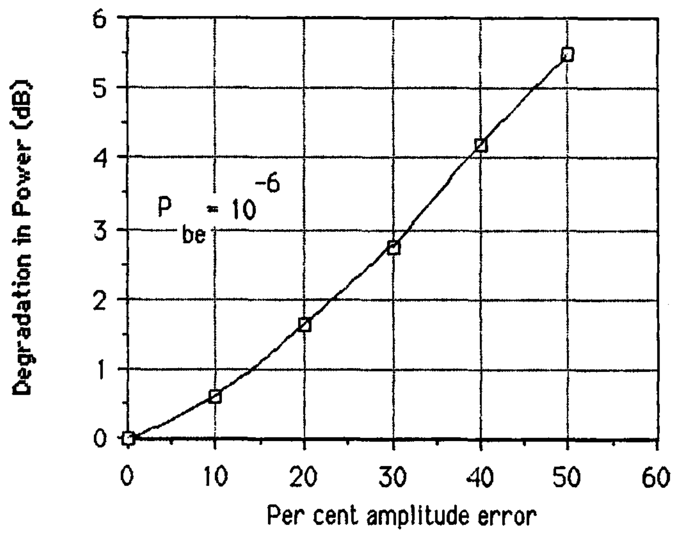

Figure 9.5 Degradation due to amplitude imbalance in QPSK

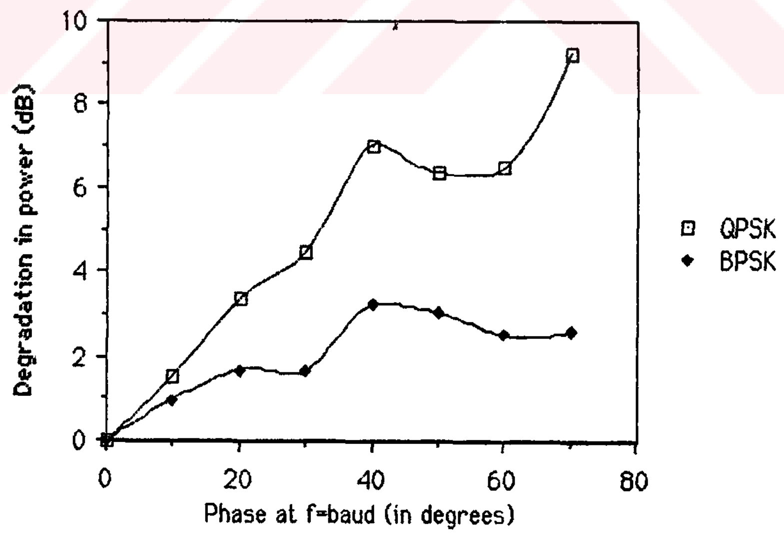

Figure 9.6 Effects of linear delay distortion on the system performance with QPSK and BPSK modulations at $P_{be}=10^{-6}$

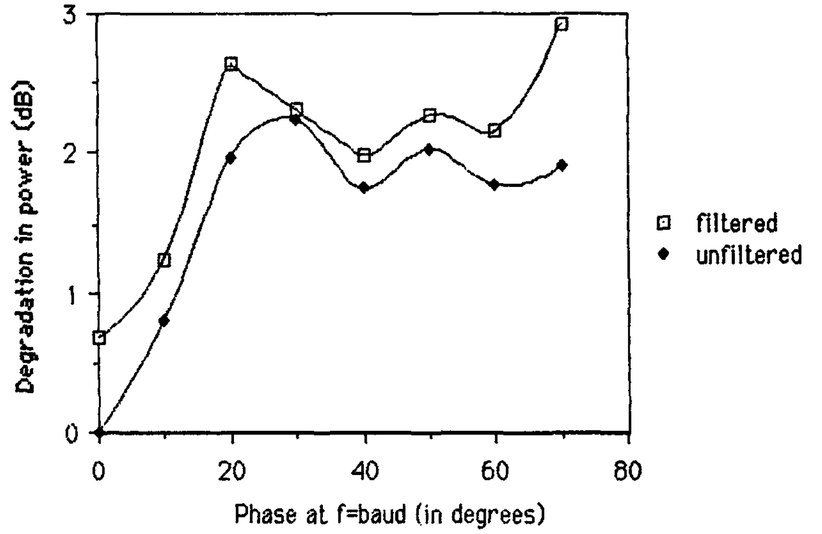

Figure 9.7 Effects of prefiltering with 0.1 dB ripple 3rd order Chebychev filter on the quadratic delay distortion imposed on QPSK

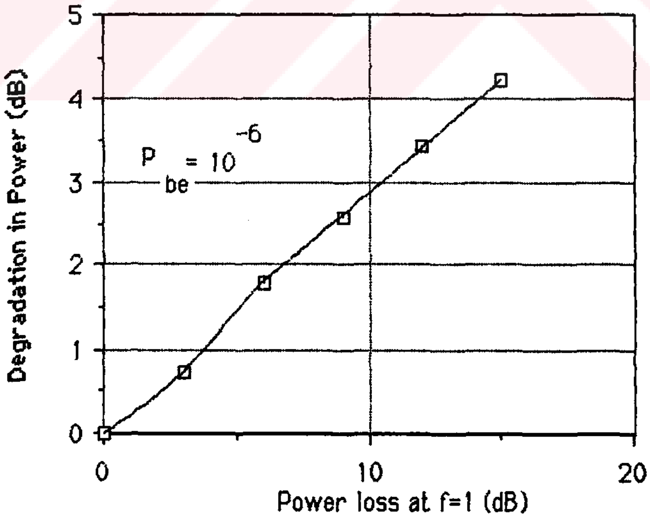

Figure 9.8 Parabolic amplitude distortion

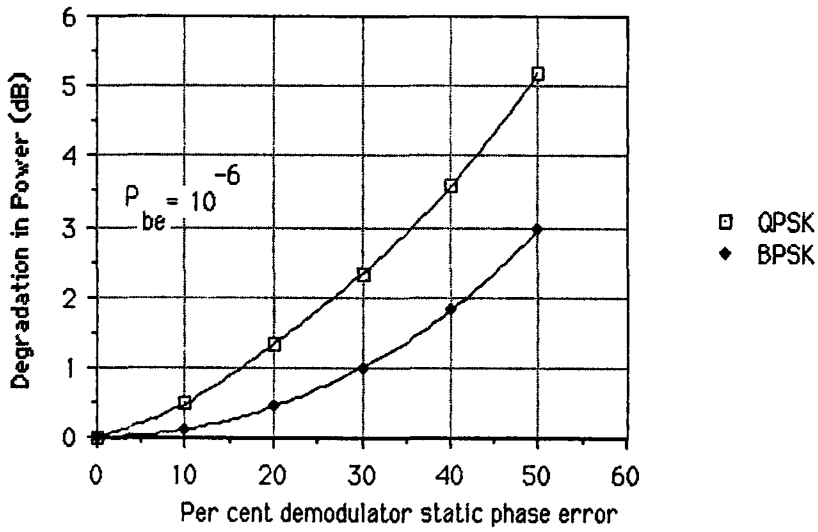

Figure 9.9 Demodulator Static Phase Error

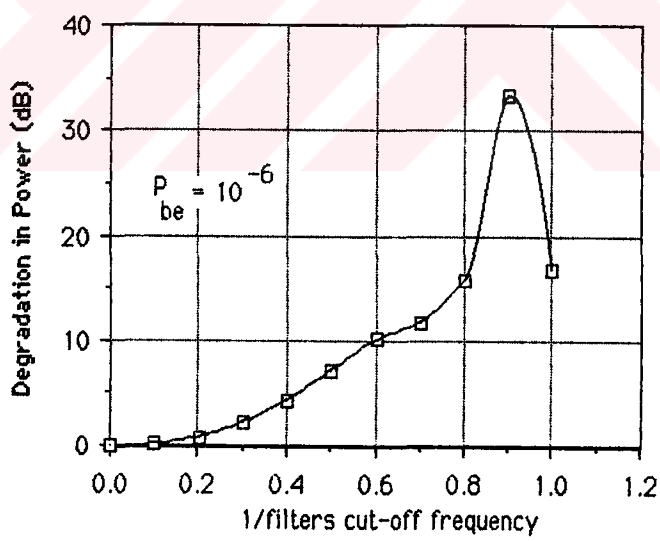

Figure 9.10 Effect of .1 dB ripple 5th order Chebychev filter on probability of error

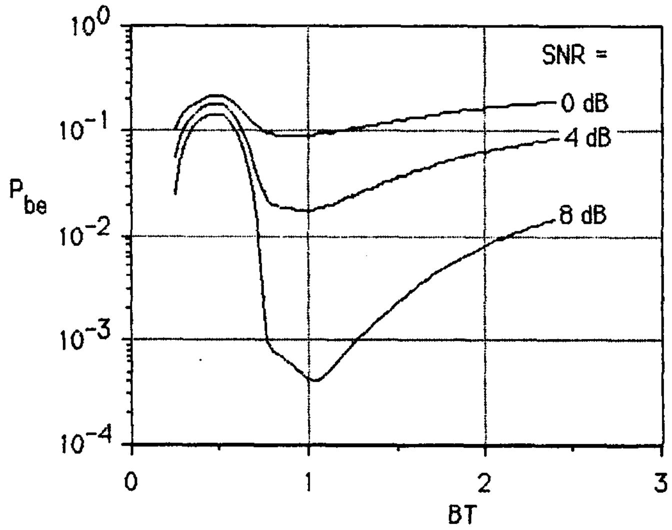

Figure 9.11 Detection of BPSK with 2nd order Butterworth filter

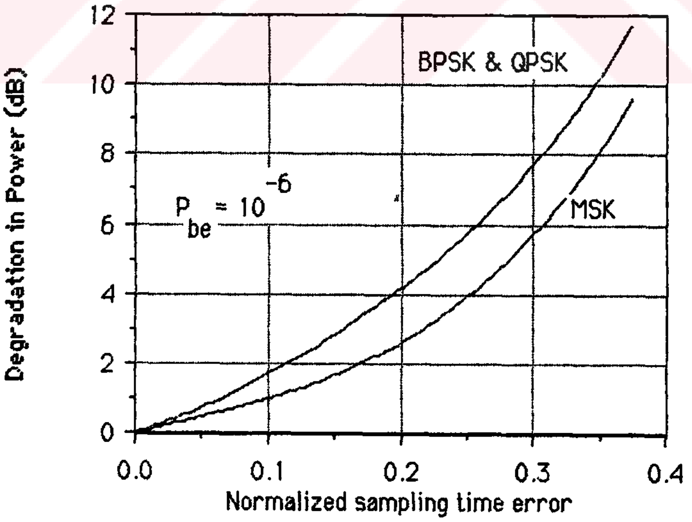

Figure 9.12 : Degradation due to Sampling Time Error

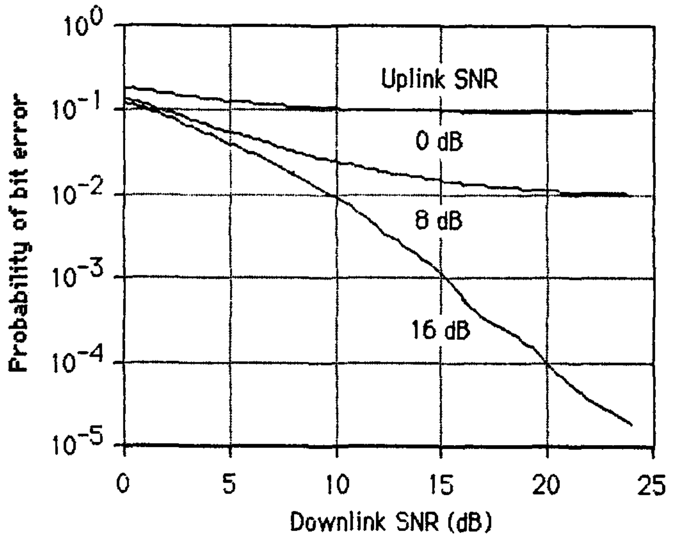

Figure 9.13 Nonlinear system response (TWT) at saturation

X. CONCLUSION

In this thesis various channel models for digital communication systems are analized and in order to investigate the performance of digital communication systems under influence of such disturbing sources, a simulation software package called' SSNDC-Semianalytic Simulator of Nonlinear Digital Channels- is implemented on Microvax II under VMS operating system, using standard FORTRAN.

Various CPSK (Coherent Phase Shift Keying) modulation types can be simulated by SSNDC. To shorted the CPU time, all of the bandpass signals and filters are generated as complex lowpass equivalents of their bandpass counterparts. Further the effect of AWGN (additive white Gaussian noise) is included analytically which makes a simulation with a short information sequence possible for evaluation of symbol error probability curves under influence of several disturbing sources.

The modulators and demodulators of SSNDC are subject to realization imperfections which are considered in chapter 3. Both linear and nonlinear channels can be simulated. Linear channel models of SSNDC include several possible linear distortion filters which introduce amplitude and phase distortions into the system and some commonly used linear filters, such as Butterworth and Chebychev filters. Nonlinear channel models of SSNDC are ZMNL (zero memory nonlinear) devices, introducing one or both of the possible nonlinear distortions which are AM/AM and AM/PM conversions. Linear and nonlinear filtering operations are performed in frequency and time domains respectively.

Relying on the quasi-analytic simulation method, SSNDC can be used to obtain probability of error curves for CPSK systems operating over linear or nonlinear channels in a few seconds.

10.1 Suggestions for Future Work

The progress in integrated circuit design and computer manufacturing technology made it possible, to implement simulation programs which required a large amount of CPU time formerly, on personel computers and to get results at reasonable time periods. The availability and economical convenience of personel computers make using them very desirable. Also using a medium level computer language like " C " the speed and portability of the simulation software may be increased.

In this thesis the Volterra series representation of nonlinearities is presented. Using this representation analytical solutions to limited amount of nonlinear systems can be obtained.

We have utilized quasi-analitical simulation technique to get the simulation results in a short time. Another possible technique to achieve this, is importance sampling which is also an active research field.

The moment space bounding technique, mentioned in chapter 5., is another interesting subject. When the interchannel interference effects are reduced to the same level as intersymbol interference effects, by the method described in appendix A, the performance of digital communication systems under the influence of all possible interferences can be calculated. The bounds obtained by this method are very tight and their convergence is rapid.

In this thesis linear and nonlinear channels in AWGN environment are considered. Other channel models which are to be implemented, are stochastic channels such as troposcatter channels and noise sources with other disributions such as impulsive noise and noise with a Ricean density function.

Currently not implemented linear filters should also be included in the model library according to the requirements.

Other features of a digital communications simulation package include various source and channel encoding capabilities and other modulation scemes [11], [26].

APPENDIX A

Obtaining The Moments Of Interference

Let us rank from 1 to M, the M statistically independent interfering samples that are significantly different from zero, so that the interference X is given by [28]

$$X = \sum_{h=1}^{M} X_h \qquad \text{(A1)}$$

let us define the partial sum

$$Y_n = \sum_{h=1}^{n} X_h \qquad \text{(A2)}$$

note that

$$Y_M = X \qquad \text{(A3)}$$

The j'th moment of X is given as

$$E\left[X^j\right] = E\left[Y_M^j\right] \qquad \text{(A4)}$$

because of the statistical independence of the interference terms

$$E\left[Y_{n+1}^j\right] = E\left[\left(Y_n + X_{n+1}\right)^j\right] \qquad \text{(A5)}$$

$$= \sum_{h=0}^{j} \begin{bmatrix} j \\ h \end{bmatrix} E\left[Y_n^h\right] E\left[X_{n+1}^{j-h}\right] \qquad \text{(A6)}$$

often X_h are even random variables, in this case

$$E\left[X^{2j+1}\right] = 0 \qquad j \geq 0 \qquad \text{(A7)}$$

$$E\left[X^{2j}\right] = \sum_{h=0}^{j} \begin{bmatrix} 2j \\ 2h \end{bmatrix} E\left[Y_{M-1}^{2h}\right] E\left[X_M^{2j-2h}\right] \qquad \text{(A8)}$$

Let X_h be a function of α_i and β_i, where α_i and β_i are also random variables. The samples X_h are not in general independent due to the same α_i and β_i, but they become statistically independent for constant values of α_i and β_i. So, the above procedure can be used in the computation of the conditional moments

$$E\left[\left(\sum_{h=1}^{M} X_h\right)^j \mid \alpha_i , \beta_i\right] \tag{A9}$$

which results in the equation

$$E\left[\left(\sum_{h=1}^{M} X_h\right)^j\right] = \iint E\left[\left(\sum_{h=1}^{M} X_h\right)^j \mid \alpha_i , \beta_i\right] dF(\alpha_i)\, dF(\beta_i) \tag{A10}$$

Evaluating the double integral in equation (A10), the moments of X can be calculated.

APPENDIX B

Obtaining The Quadrature Rules From The Moments

Let

$$\int_a^b f(x)\,\omega(x)\,dx \approx \sum_{i=1}^{m} \omega_i f(x_i) \qquad \text{(B1)}$$

the k'th moment of x is given as [29]:

$$\mu^k = \int_a^b x^k\,\omega(x)\,dx \qquad k=0,1,\ldots\ldots\ldots 2N \qquad \text{(B2)}$$

are known.

Then the Gram matrix M of the moments is formed [23], [29], [30], whose entries are given as $(M)_{ij} = \mu^{i+j}$

$i,j=0,1,\ldots\ldots\ldots\ldots N$ (B3)

The Cholesky decomposition is performed on M, such that

$$M = R^T R \qquad \text{(B4)}$$

where R is a upper triangular matrix with positive entries found as

$$r_{ii} = \left(m_{ii} - \sum_{k=1}^{i-1} r_{ki}^2 \right)^{\frac{1}{2}} \qquad \text{(B5)}$$

$$r_{ij} = \frac{\left(m_{ij} - \sum_{k=1}^{i-1} r_{ki}\, r_{kj} \right)}{r_{ii}} \qquad i<j \qquad \text{(B6)}$$

Using the entries of matrix R, new variables are obtained, given as

$$\alpha_j = \frac{r_{j,j+1}}{r_{j,j}} - \frac{r_{j-1,j}}{r_{j-1,j-1}} \qquad j=1,2,\ldots,N$$

$$\beta_j = \frac{r_{j+1,j+1}}{r_{j,j}} \qquad j=1,2,\ldots,N-1 \qquad \text{(B7)}$$

with $r_{0,0}=1$ and $r_{0,1}=0$.

The sets of $\{ \alpha_j \}$ and $\{ \beta_j \}$ are used to form the tridiagonal matrix [J] which is defined as

$$[J] = \begin{bmatrix} \alpha_1 & \beta_1 & & & & 0 \\ \beta_1 & \alpha_2 & \cdot & & & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & 0 & & \cdot & \cdot & \beta_{N-1} \\ & & & & \beta_{N-1} & \alpha_N \end{bmatrix} \qquad (B8)$$

The relation in equation (A10) is true for any set of orthogonal polynomials $\{p_j(x)\}_{j=1}^{N}$

$$p_j(x) = \left(a_j x + b_j\right) p_{j-1}(x) - c_j p_{j-2}(x) \qquad j=1,2,....,N \qquad (B9)$$

where $p_{-1}(x)=0$ and $p_0(x)=1$.Then in matrix notation:

$$x\vec{p}(x) = [J]\,\vec{p}(x) + (1/a_N)\, p_N(x)\, \vec{e}_N \qquad (B10)$$

where $e_N = [\,0,0,...,1]^T$ .It is easily seen that

$$\alpha_j = -\frac{b_j}{a_j}$$
$$\beta_j = \left(\frac{c_{j+1}}{a_j a_{j+1}}\right)^{\frac{1}{2}} \qquad (B11)$$

The quadrature rule $\{ \omega_i \,,\, x_i \}_{i=1,N}$ are obtained from the eigenvalues and eigenvectors of [J]. If

$$[J]\,\vec{q}_j = t_j \vec{q}_j \qquad (B12)$$

then

$$x_j = t_j$$
$$\omega_j = \frac{q_{1,j}^2}{p_0^2(t_j)} \qquad (B13)$$

APPENDIX C

Program Listing of SSNDC

```
PROGRAM SSNDC
COMMON /C11/ MODTYPE
COMMON /C66/ ISYSTEMTYPE
OPEN(UNIT=1,FILE='RESULT.DAT',STATUS='UNKNOWN')
OPEN(UNIT=2,FILE='PLOT.DAT',STATUS='UNKNOWN')
CALL CONST
CALL SYSTEM_CONFIGURATOR
CALL INITIALIZE
PRINT*,CHAR(27)//'[2J'
IF(MODTYPE.EQ.11)THEN
  WRITE(1,*)'        BPSK SIMULATION'
  WRITE(6,*)'        BPSK SIMULATION'
ELSE IF(MODTYPE.EQ.12)THEN
  WRITE(1,*)'        QPSK SIMULATION'
  WRITE(6,*)'        QPSK SIMULATION'
ELSE IF(MODTYPE.EQ.13)THEN
  WRITE(1,*)'        OK-QPSK SIMULATION'
  WRITE(6,*)'        OK-QPSK SIMULATION'
ELSE IF(MODTYPE.EQ.14)THEN
  WRITE(1,*)'        MSK SIMULATION'
  WRITE(6,*)'        MSK SIMULATION'
ENDIF

IF(ISYSTEMTYPE.EQ.1)CALL LINEAR_SYSTEM_SIMULATOR
IF(ISYSTEMTYPE.EQ.2)CALL NONLINEAR_SYSTEM_SIMULATOR
END
```

```
C=====================================================
        SUBROUTINE SYSTEM_CONFIGURATOR
C this subroutine inputs the parameters from the keyboard

        CHARACTER*1 CH
        CHARACTER*2 CHA

        INTEGER INPUTDATA(4096)
        COMPLEX SIGNAL(16384)
        COMPLEX GAUSSNOISE(16384)
        INTEGER OUTPUTDATA(4096)
        INTEGER*4 RANDOMNUMBER(2)
        CHARACTER*1 PLOTCONTROL
        CHARACTER*1 AWGNBEFORE
        INTEGER REGISTER(24)
        CHARACTER*1 ERRORTYPE
        DIMENSION TRANSFERFUNC(16384)
        COMPLEX FFTOUT(16384)
        COMPLEX NOISE(16384)
        REAL COLOREDNOISE3(128)
        DIMENSION LFILTERORDER(10)
        DIMENSION CRITICALFREQ(10)
        DIMENSION LFILTERNUMBER(10)
        DIMENSION NLFILTERNUMBER(10)
        DIMENSION PASSBANDRIPPLE(10)

        COMMON /C1/ PI,PI2
        COMMON /C2/ INPUTDATA
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C7/ SIGNAL
        COMMON /C9/ GAUSSNOISE
        COMMON /C10/ RANDOMNUMBER
        COMMON /C11/ MODTYPE
        COMMON /C12/ OUTPUTDATA
        COMMON /C14/ REGISTER
        COMMON /C15/ NREG
        COMMON /C16/ DELTASNR
        COMMON /C17/ NERROR
        COMMON /C18/ ERRORRATIO
        COMMON /C19/ ERRORTYPE
        COMMON /C20/ SAMINT
        COMMON /C21/ BITLENGTH
        COMMON /C23/ SAMFREQ
        COMMON /C24/ TRANSFERFUNC
        COMMON /C29/ FFTOUT
        COMMON /C30/ PLOTCONTROL
        COMMON /C33/ PHASEOFFSET
        COMMON /C34/ AMPLITUDEOFFSET
        COMMON /C35/ LFILTERNUMBER
        COMMON /C36/ CRITICALFREQ
        COMMON /C37/ LFILTERORDER
        COMMON /C38/ ALIN
        COMMON /C39/ APAR
        COMMON /C40/ ARS,NSRIP
        COMMON /C41/ ARC,NCRIP
        COMMON /C42/ GLIN
        COMMON /C43/ GPAR
        COMMON /C44/ GRS,NGSRIP
        COMMON /C45/ GRC,NGCRIP
        COMMON /C51/ PASSBANDRIPPLE
        COMMON /C55/ SIGNALPOWER
```

```
COMMON /C56/ NLFILTERNUMBER
COMMON /C57/ DEMOD_PHASEERROR
COMMON /C58/ NOISE
COMMON /C60/ LSHIFT
COMMON /C61/ COLOREDNOISE3
COMMON /C62/ COLOREDNOISEPOWER
COMMON /C63/ IDETECTIONTYPE
COMMON /C64/ EQUIVNOISEBANDW
COMMON /C65/ BACKOFF
COMMON /C66/ ISYSTEMTYPE
COMMON /C67/ AWGNBEFORE
COMMON /C75/ UPSNR
COMMON /C76/ ISNRINITIAL
COMMON /C77/ ISNREND
COMMON /C78/ ISNR

PRINT*,CHAR(27)//'[2J'
PRINT*,'############################################################'
PRINT*,'     ENTER NUMBER OF SYMBOLS GENERATED (power of 2)?'
PRINT*
PRINT*,'############################################################'
PRINT*
READ(5,*)NBITS

PRINT*,CHAR(27)//'[2J'
PRINT*,'############################################################'
PRINT*,'      ENTER NUMBER OF SAMPLES/SYMBOL (power of 2)?'
PRINT*
PRINT*,'############################################################'
PRINT*
READ(5,*)NUMSAM
BITLENGTH=1.
SAMFREQ=REAL(NUMSAM)
SAMINT=BITLENGTH/REAL(NUMSAM)

PRINT*,CHAR(27)//'[2J'
PRINT*,'###############################################'
PRINT*,' CHOOSE THE TYPE OF THE MODULATION '
PRINT*
PRINT*,' 11 ..... BPSK'
PRINT*,' 12 ..... QPSK'
PRINT*,' 13 ..... OK-QPSK'
PRINT*,' 14 ..... MSK'
PRINT*,'###############################################'
PRINT*
READ(5,*)MODTYPE

PRINT*,CHAR(27)//'[2J'
PRINT*,'############################################################'
PRINT*,' C H O O S E   THE   S Y S T E M   T Y P E'
PRINT*
PRINT*,'  1 ..... LINEAR'
PRINT*,'  2 ..... NONLINEAR'
PRINT*,'############################################################'
PRINT*
READ(5,*) ISYSTEMTYPE

PRINT*,CHAR(27)//'[2J'
PRINT*,'###########################################################'
PRINT*
PRINT*,'          ENTER THE FOLLOWING VALUES FOR '
IF(ISYSTEMTYPE.EQ.2)THEN
```

```
          PRINT*,'             DOWNLINK NOISE in (dB)'
        ELSE
          PRINT*,'                  AWGN in (dB)'
        ENDIF
        PRINT*,'*****************************************************'
        WRITE(6,*)'INITIAL SNR = ?'
        READ(5,*)ISNRINITIAL
        WRITE(6,*)'FINAL SNR = ?'
        READ(5,*)ISNREND
        WRITE(6,*)'SNR STEP = ?'
        READ(5,*)ISNR

600     IF(MODTYPE.GE.11.AND.MODTYPE.LE.14)THEN
        PRINT*,CHAR(27)//'[2J'
501       PRINT*,'*****************************************************'
          PRINT*,' ANY REALIZATION IMPERFECTIONS IN MODEMS (Y/N) ?'
          PRINT*,'     '
          PRINT*,'*****************************************************'
502       READ(5,110)CH
          IF(CH.EQ.'Y')THEN
500         PRINT*,'ENTER YOUR CHOICE'
            PRINT*
            PRINT*,' 1.....PHASE UNBALANCE'
            PRINT*,' 2.....AMPLITUDE UNBALANCE'
            PRINT*,' 3.....DEMODULATOR PHASE ERROR'
            PRINT*,' 4.....SAMPLING TIME ERROR'
            READ(5,110)CH
            IF(CH.EQ.'1')THEN
510           PRINT*,'ENTER NORMALIZED PHASE UNBALANCE'
              PRINT*,'  MINIMUM = 0.0'
              PRINT*,'  MAXIMUM = 1.0'
              READ(5,*)PHASEOFFSET
              IF(PHASEOFFSET.LT.0..OR.PHASEOFFSET.GT.1.)GOTO 510
            ELSE IF(CH.EQ.'2')THEN
520             PRINT*,'ENTER NORMALIZED AMPLITUDE UNBALANCE'
                PRINT*,'  MINIMUM = 0.0'
                PRINT*,'  MAXIMUM = 1.0'
                READ(5,*)AMPLITUDEOFFSET
                IF(AMPLITUDEOFFSET.LT.0..OR.AMPLITUDEOFFSET.GT.1.)
     $GOTO 520
            ELSE IF(CH.EQ.'3')THEN
525             PRINT*,'ENTER NORMALIZED DEMODULATOR PHASE ERROR'
                PRINT*,'  MINIMUM = 0.0'
                PRINT*,'  MAXIMUM = 1.0'
                READ(5,*)DEMOD_PHASEERROR
                IF(DEMOD_PHASEERROR.LT.0..OR.DEMOD_PHASEERROR.GT.1.)
     $GOTO 525
            ELSE IF(CH.EQ.'4')THEN
527             PRINT*,'ENTER SAMPLING TIME ERROR IN NUMBER OF SAMPLES'
                PRINT*,'  MINIMUM = 0'
                PRINT*,'  MAXIMUM = ',NUMSAM
                READ(5,*)LSHIFT
                IF(LSHIFT.LT.0.OR.LSHIFT.GT.NUMSAM)
     $GOTO 527
            ELSE
              GOTO 500
            ENDIF
            PRINT*,'  PHASE IMBALANCE          =',PHASEOFFSET
            PRINT*,'  AMPLITUDE IMBALANCE      =',AMPLITUDEOFFSET
            PRINT*,'  DEMODULATOR PHASE ERROR  =',DEMOD_PHASEERROR
            PRINT*,'  SAMPLING TIME SHIFT      =',LSHIFT,'  SAMPLES'
            PRINT*
```

```
          PRINT*,'  ANY CHANGE IN THESE VALUES (Y/N) ?'
          GOTO 502
530       IF(CH.NE.'N')GOTO 501
        ENDIF
        PRINT*,CHAR(27)//'[2J'
      ENDIF

      IF(ISYSTEMTYPE.EQ.2)THEN
        PRINT*,CHAR(27)//'[2J'
        PRINT*,'##########################################################'
        PRINT*
        PRINT*,' AWGN BEFORE THE NONLINEARITY (Y/N) ?'
        PRINT*,'##########################################################'
        READ(5,110)CH
        IF(CH.EQ.'Y') THEN
          AWGNBEFORE=CH
          DO K=1,2
      PRINT*,CHAR(27)//'[2J'
      PRINT*,'##############################################'
      PRINT*
      PRINT*,' ENTER A 5-DIGIT ODD NUMBER'
      PRINT*
      PRINT*,'##############################################'
      READ(5,*)RANDOMNUMBER(K)
          END DO
          PRINT*,CHAR(27)//'[2J'
          PRINT*,'##############################################'
          PRINT*
          PRINT*,'  ENTER UPLINK SNR in (dB)'
          PRINT*
          PRINT*,'##############################################'
          PRINT*
          READ(*,*)UPSNR
          WRITE(1,*)'  UPLINK SNR =',UPSNR,' (dB)'
        ENDIF
      ENDIF

      PRINT*,CHAR(27)//'[2J'
      PRINT*,'##########################################################'
      PRINT*
561   IF(ISYSTEMTYPE.EQ.2)THEN
        PRINT*,' ANY LINEAR FILTERING BEFORE NONLINEARITY (Y/N) ?'
      ELSE
        PRINT*,' ANY LINEAR TRANSMITTER FILTERING (Y/N) ?'
      ENDIF
      PRINT*,'##########################################################'
      KK=0
560   READ(5,110)CH
      IF(CH.EQ.'Y')THEN
        KK=KK+1
        PRINT*,'ENTER YOUR CHOICE'
        PRINT*
        PRINT*,'  1.....EQUALIZED BUTTERWORTH FILTER'
        PRINT*,'  2.....BUTTERWORTH FILTER'
        PRINT*,'  3.....CHEBYCHEV FILTER'

        READ(5,*)ICHOICE
        IF(ICHOICE.EQ.1)THEN
          LFILTERNUMBER(KK)=1
        ELSE IF(ICHOICE.EQ.2)THEN
          LFILTERNUMBER(KK)=2
        ELSE IF(ICHOICE.EQ.3)THEN
```

```
        LFILTERNUMBER(KK)=3
        PRINT*,' ENTER PEAK VALUE OF RIPPLE in  (dB)'
        READ(5,*)PASSBANDRIPPLE(KK)
      ENDIF


        PRINT*,' ENTER ORDER OF FILTER'
        READ(5,*)LFILTERORDER(KK)
        WRITE(1,*)'LFILTERORDER=',LFILTERORDER(KK)
        PRINT*,' ENTER NORMALIZED CRITICAL FREQUENCY'
        READ(5,*)CRITICALFREQ(KK)
        WRITE(1,*)'CRITICALFREQ=',CRITICALFREQ(KK)

    IF(ISYSTEMTYPE.EQ.2)THEN
      GOTO 680
    ELSE
      PRINT*,' ANY OTHER LINEAR TRANSMITTER FILTERING (Y/N) ?'
      GOTO 560
    ENDIF

    ELSE IF(CH.NE.'N')THEN
      GOTO 560
    ENDIF

    IF(ISYSTEMTYPE.NE.2)GOTO 700

680  NKK=0
     NKK=NKK+1
     PRINT*,CHAR(27)//'[2J'
     PRINT*,'#########################################################'
     PRINT*
     PRINT*,' ENTER  NONLINEARITY MODEL'
     PRINT*,'#########################################################'
      PRINT*
      PRINT*,'   1.....CLIPPER'
      PRINT*,'   2.....HARD LIMITER'
      PRINT*,'   3.....TWT'

      READ(5,*)ICHOICE
      IF(ICHOICE.EQ.1)THEN
        NLFILTERNUMBER(NKK)=1
      ELSE IF(ICHOICE.EQ.2)THEN
        NLFILTERNUMBER(NKK)=2
      ELSE IF(ICHOICE.EQ.3)THEN
        NLFILTERNUMBER(NKK)=3
        PRINT*,' ENTER INPUTPOWER BACKOFF in (dB)'
         READ(5,*)BACKOFF
      ENDIF

    PRINT*,CHAR(27)//'[2J'
    PRINT*,'#########################################################'
    PRINT*
    PRINT*,' ANY LINEAR FILTERING AFTER NONLINEARITY (Y/N) ?'
    PRINT*,'#########################################################'
860 READ(5,110)CH
    IF(CH.EQ.'Y')THEN
      KK=2
      PRINT*,'ENTER YOUR CHOICE'
      PRINT*
      PRINT*,'   1.....EQUALIZED BUTTERWORTH FILTER'
      PRINT*,'   2.....BUTTERWORTH FILTER'
      PRINT*,'   3.....CHEBYCHEV FILTER'
```

```
      READ(5,*)ICHOICE
      IF(ICHOICE.EQ.1)THEN
        LFILTERNUMBER(KK)=1
      ELSE IF(ICHOICE.EQ.2)THEN
        LFILTERNUMBER(KK)=2
      ELSE IF(ICHOICE.EQ.3)THEN
        LFILTERNUMBER(KK)=3
        PRINT*,' ENTER PEAK VALUE OF RIPPLE in  (dB)'
        READ(5,*)PASSBANDRIPPLE(KK)
      ENDIF

        PRINT*,' ENTER ORDER OF FILTER'
        READ(5,*)LFILTERORDER(KK)
        WRITE(1,*)'LFILTERORDER=',LFILTERORDER(KK)
        PRINT*,' ENTER NORMALIZED CRITICAL FREQUENCY'
        READ(5,*)CRITICALFREQ(KK)
        WRITE(1,*)'CRITICALFREQ=',CRITICALFREQ(KK)

    ELSE IF(CH.NE.'N')THEN
      GOTO 860
    ENDIF
     GOTO 1000

700 PRINT*,CHAR(27)//'[2J'
    PRINT*,'##########################################################'
    PRINT*
661 PRINT*,' ANY LINEAR DISTORTION (Y/N) ?'
    PRINT*,'##########################################################'
660 READ(5,110)CH
    IF(CH.EQ.'Y')THEN
      KK=KK+1
      PRINT*,'ENTER YOUR CHOICE'
      PRINT*
      PRINT*,'  1.....LINEAR AMPLITUDE DISTORTION'
      PRINT*,'  2.....PARABOLIC AMPLITUDE DISTORTION'
      PRINT*,'  3.....RIPPLE AMPLITUDE DISTORTION (Type I)'
      PRINT*,'  4.....RIPPLE AMPLITUDE DISTORTION (Type II)'
      PRINT*,'  5.....LINEAR GROUP DELAY'
      PRINT*,'  6.....PARABOLIC GROUP DELAY'
      PRINT*,'  7.....RIPPLE GROUP DELAY (Type I)'
      PRINT*,'  8.....RIPPLE GROUP DELAY (Type II)'
      READ(5,*)ICHOICE2
      IF(ICHOICE2.EQ.1)THEN
        LFILTERNUMBER(KK)=11
        PRINT*,' ENTER SLOPE OF THE LINEAR AMPLITUDE
   $ DISTORTION in  (dB/MHz)'
        READ(5,*)ALIN
        WRITE(1,*)' ALIN = ',ALIN,' (dB/MHz)'
      ELSE IF(ICHOICE2.EQ.2)THEN
        LFILTERNUMBER(KK)=12
        PRINT*,' ENTER PARABOLIC AMPLITUDE DISTORTION
   $ COEFFICIENT in  (dB/(MHz)2)'
        READ(5,*)APAR
      ELSE IF(ICHOICE2.EQ.3)THEN
        LFILTERNUMBER(KK)=13
        PRINT*,' ENTER PEAK VALUE OF RIPPLE in  (dB)'
        READ(5,*)ARS
        WRITE(1,*)' ARS = ',ARS,' (dB)'
        PRINT*,' ENTER NUMBER OF RIPPLES'
        READ(5,*)NSRIP
        WRITE(1,*)' NSRIP = ',NSRIP
```

```
        ELSE IF(ICHOICE2.EQ.4)THEN
          LFILTERNUMBER(KK)=14
          PRINT*,' ENTER PEAK VALUE OF RIPPLE in  (dB)'
          READ(5,*)ARC
          WRITE(1,*)' ARC = ',ARC,' (dB)'
          PRINT*,' ENTER NUMBER OF RIPPLES'
          READ(5,*)NCRIP
          WRITE(1,*)' NCRIP = ',NCRIP
        ELSE IF(ICHOICE2.EQ.5)THEN
          LFILTERNUMBER(KK)=15
          PRINT*,' ENTER LINEAR GROUP DELAY in  (ns/MHz)'
          READ(5,*)GLIN
        ELSE IF(ICHOICE2.EQ.6)THEN
          LFILTERNUMBER(KK)=16
          PRINT*,' ENTER PARABOLIC GROUP DELAY in  (ns/MHz2)'
          READ(5,*)GPAR
        ELSE IF(ICHOICE2.EQ.7)THEN
          LFILTERNUMBER(KK)=17
          PRINT*,' ENTER MAXIMUM DELAY in  (ns)'
          READ(5,*)GRS
          WRITE(1,*)' GRS = ',GRS,' (ns)'
          PRINT*,' ENTER NUMBER OF RIPPLES'
          READ(5,*)NGSRIP
          WRITE(1,*)' NGSRIP = ',NGSRIP
        ELSE IF(ICHOICE2.EQ.8)THEN
          LFILTERNUMBER(KK)=18
          PRINT*,' ENTER MAXIMUM DELAY in  (ns)'
          READ(5,*)GRC
          WRITE(1,*)' GRC = ',GRC,' (ns)'
          PRINT*,' ENTER NUMBER OF RIPPLES'
          READ(5,*)NGCRIP
          WRITE(1,*)' NGCRIP = ',NGCRIP
        ELSE
          GOTO 661
        ENDIF

        PRINT*,' ANY OTHER LINEAR DISTORTION (Y/N) ?'
        GOTO 660
      ELSE IF(CH.NE.'N')THEN
        GOTO 660
      ENDIF


      PRINT*,CHAR(27)//'[2J'
      PRINT*,'######################################################'
      PRINT*
761   PRINT*,' ANY LINEAR RECEIVER FILTERING (Y/N) ?'
      PRINT*,'######################################################'
760   READ(5,110)CH
      IF(CH.EQ.'Y')THEN
        KK=KK+2
        PRINT*,'ENTER YOUR CHOICE'
        PRINT*
        PRINT*,'  1.....EQUALIZED BUTTERWORTH FILTER'
        PRINT*,'  2.....BUTTERWORTH FILTER'
        PRINT*,'  3.....CHEBYCHEV FILTER'

        READ(5,*)ICHOICE
        IF(ICHOICE.EQ.1)THEN
          LFILTERNUMBER(KK)=1
        ELSE IF(ICHOICE.EQ.2)THEN
          LFILTERNUMBER(KK)=2
```

```
      ELSE IF(ICHOICE.EQ.3)THEN
        LFILTERNUMBER(KK)=3
        PRINT*,' ENTER PEAK VALUE OF RIPPLE in  (dB)'
        READ(5,*)PASSBANDRIPPLE(KK)
      ENDIF

        PRINT*,' ENTER ORDER OF FILTER'
        READ(5,*)LFILTERORDER(KK)
        WRITE(1,*)'LFILTERORDER=',LFILTERORDER(KK)
        PRINT*,' ENTER NORMALIZED CRITICAL FREQUENCY'
        READ(5,*)CRITICALFREQ(KK)
        WRITE(1,*)'CRITICALFREQ=',CRITICALFREQ(KK)

    ELSE IF(CH.NE.'N')THEN
      GOTO 760
    ENDIF

    IF(MODTYPE.EQ.11)THEN
      PRINT*,CHAR(27)//'[2J'
      PRINT*,'###########################################################'
      PRINT*,'        ENTER YOUR CHOICE'
      PRINT*
      PRINT*,'  1.....MATCHED FILTER DETECTION'
      PRINT*,'  2.....BUTTERWORTH FILTER DETECTION'
      PRINT*,'###########################################################'
      READ(5,*)IDETECTIONTYPE
      IF(IDETECTIONTYPE.EQ.2)THEN
        KK=KK+3
        LFILTERNUMBER(KK)=2
        PRINT*,' ENTER ORDER OF FILTER'
        READ(5,*)LFILTERORDER(KK)
        WRITE(1,*)'BUTTERWORTH DETECTION FILTER'
        WRITE(1,*)'LFILTERORDER=',LFILTERORDER(KK)
        PRINT*,' ENTER NORMALIZED CRITICAL FREQUENCY'
        READ(5,*)CRITICALFREQ(KK)
        WRITE(1,*)'CRITICALFREQ=',CRITICALFREQ(KK)
        DUM=PI/REAL(2*LFILTERORDER(KK))
        EQUIVNOISEBANDW=(DUM/SIN(DUM))*CRITICALFREQ(KK)
      ENDIF
    ENDIF

110 FORMAT(A1)

1000 RETURN
    END
```

```
C==========================================================
        SUBROUTINE NONLINEAR_SYSTEM_SIMULATOR

         CHARACTER*1 AWGNBEFORE
        COMPLEX FFTOUT(16384)
        COMPLEX SIGNAL(16384)
        COMPLEX NOISYSIGNAL(16384)
        COMPLEX NOISEFREESIGNAL(16384)
        COMPLEX GAUSSNOISE(16384)
        INTEGER INPUTDATA(4096)
        DIMENSION LFILTERNUMBER(10)
         DIMENSION NLFILTERNUMBER(10)

        COMMON /C2/ INPUTDATA
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C7/ SIGNAL
        COMMON /C9/ GAUSSNOISE
        COMMON /C11/ MODTYPE
        COMMON /C15/ NREG
        COMMON /C17/ NERROR
        COMMON /C18/ ERRORRATIO
        COMMON /C29/ FFTOUT
        COMMON /C34/ AMPLITUDEOFFSET
        COMMON /C35/ LFILTERNUMBER
        COMMON /C40/KSNR
        COMMON /C50/ NOISYSIGNAL
        COMMON /C51/ PASSBANDRIPPLE
        COMMON /C56/ NLFILTERNUMBER
         COMMON /C65/ BACKOFF
         COMMON /C67/ AWGNBEFORE
         COMMON /C68/ ERROR
         COMMON /C75/ UPSNR
         COMMON /C76/ ISNRINITIAL
         COMMON /C77/ ISNREND
         COMMON /C78/ ISNR

        REAL NOISEPOWER,NOISEPOWERNEW
         REAL DOWNSNR(10)
         REAL AVERAGEERROR(10)

         NODOWNSNR=((ISNREND-ISNRINITIAL)/ISNR)+1
         DO I=1,NODOWNSNR
           DOWNSNR(I)=REAL(ISNRINITIAL+(I-1)*ISNR)
         END DO

        CALL PRBSGENERATOR
         IF (MODTYPE.NE.11)THEN
           CALL MLCS
           NUMSAM=NUMSAM/2
         ENDIF

        IF ( MODTYPE.EQ.11 )CALL BPSK_MODULATOR
        IF ( MODTYPE.EQ.12 )CALL QPSK_MODULATOR
        IF ( MODTYPE.EQ.13 )CALL OKQPSK_MODULATOR
        IF ( MODTYPE.EQ.14 )CALL MSK_MODULATOR

C signalpower and uplink noise power correction factor calculation
        SIGNALPOWER=0.0
        DO  I=1,NUMSAM*NBITS
            SIGNALPOWER=SIGNALPOWER+REAL(SIGNAL(I)*CONJG(SIGNAL(I)))
        END DO
```

```
      SIGNALPOWER=SIGNALPOWER/FLOAT(2*NUMSAM*NBITS)
      CORRECTIONFACTOR=(REAL(NUMSAM)*SIGNALPOWER)/(10.0**(UPSNR/10.0))

      DO I=1,NUMSAM*NBITS
        NOISEFREESIGNAL(I)=SIGNAL(I)
      END DO

      DO I=1,NODOWNSNR
        AVERAGEERROR(I)=0
      END DO
      NRUNS=0
      IRUNCONTROL=0
      CALL TIME_BEGIN

600   CALL GAUSSNOISEGENERATOR

C addition of uplink noise
      DO 200 I=1,NUMSAM*NBITS
        SIGNAL(I)=NOISEFREESIGNAL(I)+GAUSSNOISE(I)*
     $SQRT(CORRECTIONFACTOR)
200   CONTINUE

C zero padding ,to ensure linear convolution equivalence
      LDIM=NUMSAM*NBITS
      DO I=LDIM+1,2*LDIM
        SIGNAL(I)=CMPLX(0.)
      END DO
      NBITS=NBITS*2

C linear filtering in frequency domain before memoryless nonlinearity
        KK=1
        IF(LFILTERNUMBER(1).NE.0)THEN
          CALL FFT(SIGNAL,NUMSAM*NBITS,0)
          IF(LFILTERNUMBER(KK).EQ.1)CALL EQUALIZED_BUTTERWORTH(KK)
          IF(LFILTERNUMBER(KK).EQ.2)CALL BUTTERWORTH(KK)
          IF(LFILTERNUMBER(KK).EQ.3)CALL CHEBYCHEV(KK)
            DO I=1,NUMSAM*NBITS
              SIGNAL(I)=CONJG(FFTOUT(I))
            END DO
            CALL FFT(SIGNAL,NUMSAM*NBITS,0)
            DO I=1,NUMSAM*NBITS
              SIGNAL(I)=CONJG(FFTOUT(I))/CMPLX(FLOAT(NUMSAM*NBITS))
            END DO
        ENDIF
        NBITS=NBITS/2


C nonlinear filtering in time domain
        IF(NLFILTERNUMBER(1).EQ.1)CALL CLIPPER
        IF(NLFILTERNUMBER(1).EQ.2)CALL HARD_LIMITER
        IF(NLFILTERNUMBER(1).EQ.3)THEN
          CALL TWT(BACKOFF,PSHIFT)
          CALL PHASECOMPENSATOR(PSHIFT)
        ENDIF

C zero padding ,to ensure linear convolution equivalence
      DO I=LDIM+1,2*LDIM
        SIGNAL(I)=CMPLX(0.)
      END DO
      NBITS=NBITS*2

C linear filtering in frequency domain after memoryless nonlinearity
```

```
      IF(LFILTERNUMBER(2).NE.0)THEN
        KK=2
        CALL FFT(SIGNAL,NUMSAM*NBITS,0)
        IF(LFILTERNUMBER(KK).EQ.1)CALL EQUALIZED_BUTTERWORTH(KK)
        IF(LFILTERNUMBER(KK).EQ.2)CALL BUTTERWORTH(KK)
        IF(LFILTERNUMBER(KK).EQ.3)CALL CHEBYCHEV(KK)
          DO I=1,NUMSAM*NBITS
            SIGNAL(I)=CONJG(FFTOUT(I))
          END DO
          CALL FFT(SIGNAL,NUMSAM*NBITS,0)
          DO I=1,NUMSAM*NBITS
            SIGNAL(I)=CONJG(FFTOUT(I))/CMPLX(FLOAT(NUMSAM*NBITS))
          END DO
        ENDIF
        NBITS=NBITS/2

      ISNRNUMBER=0
      WRITE(6,*)'NUMBER OF BITS CONSIDERED =',(NRUNS+1)*NBITS

C calculation of BER for a number of downlink SNR's
      DO KSNR=ISNRINITIAL,ISNREND,ISNR
         ISNRNUMBER=ISNRNUMBER+1

        IF ( MODTYPE.EQ.11 )CALL BPSK_C_DEMODULATOR
        IF ( MODTYPE.EQ.12 )CALL QPSK_C_DEMODULATOR
        IF ( MODTYPE.EQ.13 )CALL OKQPSK_C_DEMODULATOR
        IF ( MODTYPE.EQ.14 )CALL MSK_C_DEMODULATOR

C decision mechanism to use another additional set of random uplink
C noise samples to increase confidence level
        AVERAGEOLD=AVERAGEERROR(ISNRNUMBER)
        AVERAGEERROR(ISNRNUMBER)=AVERAGEERROR(ISNRNUMBER)*NRUNS+ERROR
        AVERAGEERROR(ISNRNUMBER)=AVERAGEERROR(ISNRNUMBER)
    $/REAL(NRUNS+1)

        WRITE(*,*)KSNR,ISNRNUMBER,AVERAGEERROR(ISNRNUMBER)
        IF(ISNRNUMBER.EQ.NODOWNSNR.AND.NRUNS.GT.1)THEN
          IF(ABS((AVERAGEERROR(ISNRNUMBER)-AVERAGEOLD)/
    $AVERAGEERROR(ISNRNUMBER)).LT.0.05)THEN
            IRUNCONTROL=IRUNCONTROL+1
          ELSE
            IRUNCONTROL=0
          ENDIF
        ENDIF

        IF(IRUNCONTROL.EQ.3)GOTO 1000
      END DO

      WRITE(*,*)' '
      NRUNS=NRUNS+1
      GOTO 600

1000  DO I=1,NODOWNSNR
       WRITE(1,177)DOWNSNR(I),AVERAGEERROR(I)
       WRITE(6,177)DOWNSNR(I),AVERAGEERROR(I)
      END DO

      WRITE(1,*)'NUMBER OF BITS CONSIDERED =',(NRUNS+1)*NBITS
      WRITE(6,*)'NUMBER OF BITS CONSIDERED =',(NRUNS+1)*NBITS
      PRINT*,CHAR(27)//'[2J'
      PRINT*,'_____________________________________________________'
      PRINT*,' SEE FILE : RESULT.DAT FOR THE RESULTS OF THIS RUN...'
```

```
      PRINT*,'____________________________________________________________'
177    FORMAT (2X,F4.1,4X,E11.3,2X,I4)
      CALL TIME_END

      RETURN
      END
```

```
C==============================================================
        SUBROUTINE GAUSSNOISEGENERATOR
C   This subroutine generates in-phase and quadrature Gaussian
C   random variables with
C      * mean = 0.0
C      * variance = 1.0

        CHARACTER*1 CH

        COMPLEX GAUSSNOISE(16384)
        INTEGER*4 RANDOMNUMBER(2)
        CHARACTER*1 PRINTCONTROL
        CHARACTER*1 REPORTCONTROL
        CHARACTER*1 PLOTCONTROL

        COMMON /C1/ PI,PI2
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C9/ GAUSSNOISE
        COMMON /C10/ RANDOMNUMBER
        COMMON /C20/ SAMINT
        COMMON /C30/ PLOTCONTROL

        REAL MEAN

        DO K=1,2

        MEAN=0.0
        VARIANCE=0.0
        DO 60 I=1,NUMSAM*NBITS
          U1=RAN(RANDOMNUMBER(K))
          U2=RAN(RANDOMNUMBER(K))
          GG=-2.0*ALOG(U1+0.000001)
          IF ( GG.LT.0.0 ) GG=0.0
        IF(K.EQ.1)THEN
          GAUSSNOISE(I)=CMPLX(SQRT(GG)*SIN(2.0*PI*(U2-0.5)))
          MEAN=MEAN+REAL(GAUSSNOISE(I))
          VARIANCE=VARIANCE+REAL(GAUSSNOISE(I)*GAUSSNOISE(I))
        ELSE
          GAUSSNOISE(I)=GAUSSNOISE(I)+CMPLX(0,-1)*CMPLX(SQRT(.5))
     $*CMPLX(SQRT(GG)*SIN(2.0*PI*(U2-0.5))),
          MEAN=MEAN-AIMAG(GAUSSNOISE(I))
          VARIANCE=VARIANCE+AIMAG(GAUSSNOISE(I))*AIMAG(GAUSSNOISE(I))
        ENDIF
60      CONTINUE
        MEAN=MEAN/FLOAT(NUMSAM*NBITS)
        VARIANCE=VARIANCE/FLOAT(NUMSAM*NBITS)
        END DO

        RETURN
        END
```

```
C=============================================================
        SUBROUTINE LINEAR_SYSTEM_SIMULATOR

        COMPLEX FFTOUT(16384)
        COMPLEX SIGNAL(16384)
        COMPLEX NOISE(16384)
        REAL COLOREDNOISE3(128)
        DIMENSION LFILTERNUMBER(10)
        DIMENSION NLFILTERNUMBER(10)

        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C7/ SIGNAL
        COMMON /C11/ MODTYPE
        COMMON /C23/ SAMFREQ
        COMMON /C29/ FFTOUT
        COMMON /C33/ PHASEOFFSET
        COMMON /C34/ AMPLITUDEOFFSET
        COMMON /C35/ LFILTERNUMBER
        COMMON /C36/ CRITICALFREQ
        COMMON /C37/ LFILTERORDER
        COMMON /C40/KSNR
        COMMON /C51/ PASSBANDRIPPLE
        COMMON /C56/ NLFILTERNUMBER
        COMMON /C58/ NOISE
        COMMON /C60/ LSHIFT
        COMMON /C61/ COLOREDNOISE3
        COMMON /C62/ COLOREDNOISEPOWER
        COMMON /C63/ IDETECTIONTYPE
        COMMON /C65/ BACKOFF
        COMMON /C76/ ISNRINITIAL
        COMMON /C77/ ISNREND
        COMMON /C78/ ISNR

        REAL NOISEPOWER,NOISEPOWERNEW

        CALL TIME_BEGIN
        CALL PRBSGENERATOR
        IF(MODTYPE.NE.11)THEN
          CALL MLCS
          NUMSAM=NUMSAM/2
        ENDIF

        IF(MODTYPE.EQ.11)CALL BPSK_MODULATOR
        IF(MODTYPE.EQ.12)CALL QPSK_MODULATOR
        IF(MODTYPE.EQ.13)CALL OKQPSK_MODULATOR
        IF(MODTYPE.EQ.14)CALL MSK_MODULATOR'

C zero padding ,to ensure linear convolution equivalence
        LDIM=NUMSAM*NBITS
        DO I=LDIM+1,2*LDIM
          SIGNAL(I)=CMPLX(0.)
        END DO
        NBITS=NBITS*2

C linear filtering in frequency domain before noise addition
          KK=0
          IF(LFILTERNUMBER(1).NE.0)THEN
            CALL FFT(SIGNAL,NUMSAM*NBITS,0)
564         KK=KK+1
            IF(LFILTERNUMBER(KK).EQ.1)CALL EQUALIZED_BUTTERWORTH(KK)
            IF(LFILTERNUMBER(KK).EQ.2)CALL BUTTERWORTH(KK)
```

```
          IF(LFILTERNUMBER(KK).EQ.3)CALL CHEBYCHEV(KK)
          IF(LFILTERNUMBER(KK).EQ.11)CALL LINEAR_AMP_DIST
          IF(LFILTERNUMBER(KK).EQ.12)CALL PARABOLIC_AMP_DIST
          IF(LFILTERNUMBER(KK).EQ.13)CALL RIPPLE_AMP_DIST_SIN
          IF(LFILTERNUMBER(KK).EQ.14)CALL RIPPLE_AMP_DIST_COS
          IF(LFILTERNUMBER(KK).EQ.15)CALL LINEAR_GROUP_DELAY
          IF(LFILTERNUMBER(KK).EQ.16)CALL PARABOLIC_GROUP_DELAY
          IF(LFILTERNUMBER(KK).EQ.17)CALL RIPPLE_GROUP_DELAY_SIN
          IF(LFILTERNUMBER(KK).EQ.18)CALL RIPPLE_GROUP_DELAY_COS
          IF(LFILTERNUMBER(KK).EQ.0)THEN
            KK=KK-1
            DO I=1,NUMSAM*NBITS
              SIGNAL(I)=CONJG(FFTOUT(I))
            END DO
            CALL FFT(SIGNAL,NUMSAM*NBITS,0)
            DO I=1,NUMSAM*NBITS
              SIGNAL(I)=CONJG(FFTOUT(I))/CMPLX(FLOAT(NUMSAM*NBITS))
            END DO
            GOTO 601
          ENDIF
          GOTO 564
        ENDIF
601     NBITS=NBITS/2

C zero padding ,to ensure linear convolution equivalence
        DO I=LDIM+1,2*LDIM
          SIGNAL(I)=CMPLX(0.)
        END DO
        NBITS=NBITS*2

C linear filtering in frequency domain after noise addition
          IF(LFILTERNUMBER(KK+2).NE.0)THEN
            KK=KK+2
            CALL FFT(SIGNAL,NUMSAM*NBITS,0)
            IF(LFILTERNUMBER(KK).EQ.1)CALL EQUALIZED_BUTTERWORTH(KK)
            IF(LFILTERNUMBER(KK).EQ.2)CALL BUTTERWORTH(KK)
            IF(LFILTERNUMBER(KK).EQ.3)CALL CHEBYCHEV(KK)
              DO I=1,NUMSAM*NBITS
                SIGNAL(I)=CONJG(FFTOUT(I))
              END DO
              CALL FFT(SIGNAL,NUMSAM*NBITS,0)
              DO I=1,NUMSAM*NBITS
                SIGNAL(I)=CONJG(FFTOUT(I))/CMPLX(FLOAT(NUMSAM*NBITS))
              END DO

              CALL COLOREDNOISECALCULATOR(KK)
          KK=KK-2
        WRITE(6,*)'CORREL COLOREDNOISEPOWER=',COLOREDNOISEPOWER
          ENDIF
701     NBITS=NBITS/2

C linear filtering in by a nonmatched (Butterworth)detection filter
        DO I=LDIM+1,2*LDIM
          SIGNAL(I)=CMPLX(0.)
        END DO
        NBITS=NBITS*2
        IF(IDETECTIONTYPE.EQ.2)THEN
            KK=KK+3
            CALL FFT(SIGNAL,NUMSAM*NBITS,0)
            CALL BUTTERWORTH(KK)
              DO I=1,NUMSAM*NBITS
                SIGNAL(I)=CONJG(FFTOUT(I))
```

```
            END DO
            CALL FFT(SIGNAL,NUMSAM*NBITS,0)
            DO I=1,NUMSAM*NBITS
              SIGNAL(I)=CONJG(FFTOUT(I))/CMPLX(FLOAT(NUMSAM*NBITS))
            END DO
          KK=KK-3
        ENDIF
        NBITS=NBITS/2

C the sampling time is shifted if desired
        CALL SAMPLING_TIME_SHIFTER(SIGNAL,NUMSAM*NBITS,LSHIFT)

        DO KSNR=ISNRINITIAL,ISNREND,ISNR
          IF(MODTYPE.EQ.11)CALL BPSK_C_DEMODULATOR
          IF(MODTYPE.EQ.12)CALL QPSK_C_DEMODULATOR
          IF(MODTYPE.EQ.13)CALL OKQPSK_C_DEMODULATOR
          IF(MODTYPE.EQ.14)CALL MSK_C_DEMODULATOR
        END DO
        CALL TIME_END
        RETURN
        END
```

```
C=================================================================
        SUBROUTINE PHASECOMPENSATOR(PSHIFT)
C the phase shift introduced by the TWT nonlinearity is compensated
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C7/ SIGNAL
        COMPLEX SIGNAL(16384)

        DO I=1,NUMSAM*NBITS
          SIGNAL(I)=SIGNAL(I)*CMPLX(COS(PSHIFT),-SIN(PSHIFT))
        END DO
        RETURN
        END
```

```
C===================================================================
        SUBROUTINE TWT(BACKOFF,PSHIFT)
C computes response of INTELSAT IV TWT nonlinearity using a best
C fit by modified Bessel functions
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C7/ SIGNAL
        COMPLEX SIGNAL(16384)
        DATA C1,C2,S1,S2/1.61245,.053557,1.71850,.242218/

C for saturation point compensation
        FACTORA=10.**(9./20.)
C for operation point compensation
        FACTORA=FACTORA*10.**(-BACKOFF/20.)

        A=SQRT(2.)*FACTORA
        C2A2=C2*A*A
        C1A=C1*A
        S2A2=S2*A*A
        S1A=S1*A
        ZP=BESSI0(C2A2)/EXP(C2A2)*C1A
        ZQ=BESSI1(S2A2)/EXP(S2A2)*S1A
        PSHIFT=ATAN(ZQ/ZP)
        DO K=1,NUMSAM*NBITS
          A=SQRT(REAL(SIGNAL(K)*CONJG(SIGNAL(K))))
          A=A*FACTORA
          C2A2=C2*A*A
          C1A=C1*A
          S2A2=S2*A*A
          S1A=S1*A
          ZP=BESSI0(C2A2)/EXP(C2A2)*C1A
          ZQ=BESSI1(S2A2)/EXP(S2A2)*S1A
          SIGNAL(K)=SIGNAL(K)*CMPLX(ZP,ZQ)

C to adjust the output power to 0 dB
          SIGNAL(K)=SIGNAL(K)*(10.**(-7.852484/20.))/SQRT(2.)
        END DO
        RETURN
        END
```

```
C=====================================================================
      FUNCTION BESSI0(X)
C Numerical Recipes routine to find the modified Bessel function
C of 0'th order

      REAL*8 Y,P1,P2,P3,P4,P5,P6,P7,
     *    Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9
      DATA P1,P2,P3,P4,P5,P6,P7/1.0D0,3.5156229D0,3.0899424D0,1.2067492D
     *0,
     *    0.2659732D0,0.360768D-1,0.45813D-2/
      DATA Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9/0.39894228D0,0.1328592D-1,
     *    0.225319D-2,-0.157565D-2,0.916281D-2,-0.2057706D-1,
     *    0.2635537D-1,-0.1647633D-1,0.392377D-2/
      IF (ABS(X).LT.3.75) THEN
        Y=(X/3.75)**2
        BESSI0=P1+Y*(P2+Y*(P3+Y*(P4+Y*(P5+Y*(P6+Y*P7)))))
      ELSE
        AX=ABS(X)
        Y=3.75/AX
        BESSI0=(EXP(AX)/SQRT(AX))*(Q1+Y*(Q2+Y*(Q3+Y*(Q4
     *      +Y*(Q5+Y*(Q6+Y*(Q7+Y*(Q8+Y*Q9))))))))
      ENDIF
      RETURN
      END
```

```
C==========================================================
      FUNCTION BESSI1(X)
C Numerical Recipes routine to find the modified Bessel function
C of 1'st  order

      REAL*8 Y,P1,P2,P3,P4,P5,P6,P7,
     *    Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9
      DATA P1,P2,P3,P4,P5,P6,P7/0.5D0,0.87890594D0,0.51498869D0,
     *    0.15084934D0,0.2658733D-1,0.301532D-2,0.32411D-3/
      DATA Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8,Q9/0.39894228D0,-0.3988024D-1,
     *    -0.362018D-2,0.163801D-2,-0.1031555D-1,0.2282967D-1,
     *    -0.2895312D-1,0.1787654D-1,-0.420059D-2/
      IF (ABS(X).LT.3.75) THEN
        Y=(X/3.75)**2
        BESSI1=X*(P1+Y*(P2+Y*(P3+Y*(P4+Y*(P5+Y*(P6+Y*P7))))))
      ELSE
        AX=ABS(X)
        Y=3.75/AX
        BESSI1=(EXP(AX)/SQRT(AX))*(Q1+Y*(Q2+Y*(Q3+Y*(Q4+
     *      Y*(Q5+Y*(Q6+Y*(Q7+Y*(Q8+Y*Q9))))))))
      ENDIF
      RETURN
      END
```

```
C==============================================================
      SUBROUTINE COLOREDNOISECALCULATOR(KK)
C routine to calculate the noise power after passing
C through a linear filter

       COMPLEX COLOREDNOISE(16384)
       COMPLEX COLOREDNOISE2(16384)
       REAL COLOREDNOISE3(128)
       DIMENSION LFILTERNUMBER(10)
       COMPLEX FFTOUT(16384)

       COMMON /C3/ NUMSAM
       COMMON /C4/ NBITS
       COMMON /C23/ SAMFREQ
       COMMON /C29/ FFTOUT
       COMMON /C35/ LFILTERNUMBER
       COMMON /C61/ COLOREDNOISE3
       COMMON /C62/ COLOREDNOISEPOWER

      NUMSAMOLD=NUMSAM
      NBITSOLD=NBITS
      SAMFREQOLD=SAMFREQ
       NUMSAM=32
       NBITS=32
       SAMFREQ=REAL(NUMSAM)
       DO I=1,NUMSAM*NBITS
         FFTOUT(I)=CMPLX(1.)
       END DO

           IF(LFILTERNUMBER(KK).EQ.1)CALL EQUALIZED_BUTTERWORTH(KK)
           IF(LFILTERNUMBER(KK).EQ.2)CALL BUTTERWORTH(KK)
           IF(LFILTERNUMBER(KK).EQ.3)CALL CHEBYCHEV(KK)

      K=0
      SUM=0.
      DO  I=1,NUMSAM
         SUM=SUM+REAL(FFTOUT(I)*CONJG(FFTOUT(I)))
        K=K+1
       END DO
      COLOREDNOISEPOWER=ABS(SUM)/REAL(NUMSAM)
      WRITE(6,*)' COLOREDNOISEPOWER=',COLOREDNOISEPOWER

      NUMSAM=NUMSAMOLD
      NBITS=NBITSOLD
      SAMFREQ=SAMFREQOLD

      RETURN
      END
```

```
C=============================================================
       SUBROUTINE SAMPLING_TIME_SHIFTER(SIGNAL,LDIM,LSHIFT)
C routine to shift an array by a number of array elements

        COMPLEX SIGNAL(16384)
       COMPLEX ARRAY(16384)

       IF(LSHIFT.EQ.0)RETURN
       NO1=LDIM-LSHIFT
       DO I=1,LSHIFT
         ARRAY(I)=SIGNAL(I)
       END DO
       DO I=1,NO1
         SIGNAL(I)=SIGNAL(I+LSHIFT)
       END DO
       DO I=1,LSHIFT
         SIGNAL(NO1+I)=ARRAY(I)
       END DO
       RETURN
       END
```

```
C==================================================================
        SUBROUTINE EQUALIZED_BUTTERWORTH(LCONTROL)
C routine to generate equalized Butterworth filter frequency response
C this routine is reentrant

        COMPLEX TRANSFERFUNC(16384)
        COMPLEX FFTOUT(16384)
        DIMENSION LFILTERNUMBER(10),CRITICALFREQ(10),LFILTERORDER(10)

        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C23/ SAMFREQ
        COMMON /C29/ FFTOUT
        COMMON /C36/ CRITICALFREQ
        COMMON /C37/ LFILTERORDER

        DELTAFREQ=SAMFREQ/REAL(NUMSAM*NBITS)
        N01=NUMSAM*NBITS/2+1
        N02=N01+1
        TRANSFERFUNC(1)=CMPLX(1.)

C generates the positive frequency part of the filter
        DO I=2,N01
          J=I-1
          A2=1./SQRT(1.+(DELTAFREQ*FLOAT(J)/CRITICALFREQ(LCONTROL)
     $)**(2*LFILTERORDER(LCONTROL)))
          TRANSFERFUNC(I)=CMPLX(A2)
        END DO

C computes the negative frequency part of the filter
        DO I=N02,NUMSAM*NBITS
          TRANSFERFUNC(I)=TRANSFERFUNC(NUMSAM*NBITS+2-I)
        END DO

C multiplies the frequency responses
        DO I=1,NUMSAM*NBITS
          FFTOUT(I)=FFTOUT(I)*TRANSFERFUNC(I)
        END DO

        RETURN
        END
```

```
C=================================================================
        SUBROUTINE BUTTERWORTH(LCONTROL)
C routine to generate Butterworth filter frequency response
C this routine is reentrant

        COMPLEX TRANSFERFUNC(16384)
        COMPLEX FFTOUT(16384)
        DIMENSION LFILTERNUMBER(10),CRITICALFREQ(10),LFILTERORDER(10)

        COMMON /C1/ PI,PI2
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C23/ SAMFREQ
        COMMON /C29/ FFTOUT
        COMMON /C36/ CRITICALFREQ
        COMMON /C37/ LFILTERORDER

        DELTAFREQ=SAMFREQ/REAL(NUMSAM*NBITS)
        N01=NUMSAM*NBITS/2+1
        N02=N01+1
        TRANSFERFUNC(1)=CMPLX(1.)
        A2=0.
        A3=PI2/4./REAL(LFILTERORDER(LCONTROL))

C generates the positive frequency part of the filter
        DO I=2,N01
          J=I-1
          A2=DELTAFREQ*FLOAT(J)/CRITICALFREQ(LCONTROL)
          TRANSFERFUNC(I)=CMPLX(1.)
          DO J=1,LFILTERORDER(LCONTROL)
            A4=REAL(LFILTERORDER(LCONTROL)-1+(2*J))*A3
            TRANSFERFUNC(I)=TRANSFERFUNC(I)/
     $CMPLX(-COS(A4),-SIN(A4)+A2)
          END DO
        END DO

C computes the negative frequency part of the filter
        DO I=N02,NUMSAM*NBITS
          TRANSFERFUNC(I)=CONJG(TRANSFERFUNC(NUMSAM*NBITS+2-I))
        END DO

C multiplies the frequency responses
        DO I=1,NUMSAM*NBITS
          FFTOUT(I)=FFTOUT(I)*TRANSFERFUNC(I)
        END DO

       RETURN
        END
```

```
C==============================================================
        SUBROUTINE CHEBYCHEV(LCONTROL)
C routine to generate Chebychev filter frequency response
C this routine is reentrant

        COMPLEX TRANSFERFUNC(16384)
        COMPLEX FFTOUT(16384)
        DIMENSION LFILTERNUMBER(10),CRITICALFREQ(10),LFILTERORDER(10)
        DIMENSION PASSBANDRIPPLE(10)

        COMMON /C1/ PI,PI2
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C23/ SAMFREQ
        COMMON /C29/ FFTOUT
        COMMON /C36/ CRITICALFREQ
        COMMON /C37/ LFILTERORDER
        COMMON /C51/ PASSBANDRIPPLE

        PASSBANDRIP=PASSBANDRIPPLE(LCONTROL)
        DELTAFREQ=SAMFREQ/REAL(NUMSAM*NBITS)
        NO1=NUMSAM*NBITS/2+1
        NO2=NO1+1
        A2=0.
        A3=PI2/4./REAL(LFILTERORDER(LCONTROL))

        PASSBANDRIP=SQRT(10.**(PASSBANDRIP/10.)-1.)
        ALPHA=1./PASSBANDRIP
        ALPHA=ALPHA+SQRT(1.+ALPHA*ALPHA)
        DUMMY1=(ALPHA**(1./REAL(LFILTERORDER(LCONTROL))))
        DUMMY2=1./DUMMY1
        RIN=(DUMMY1-DUMMY2)/2.
        ROUT=(DUMMY1+DUMMY2)/2.

C generates the positive frequency part of the filter
        DO I=1,NO1
          J=I-1
          A2=DELTAFREQ*FLOAT(J)/CRITICALFREQ(LCONTROL)
          TRANSFERFUNC(I)=CMPLX(2./PASSBANDRIP)
          DO J=1,LFILTERORDER(LCONTROL)
            PHASE=REAL(LFILTERORDER(LCONTROL)-1+(2*J))*A3
            TRANSFERFUNC(I)=TRANSFERFUNC(I)/CMPLX(2.)/
     $CMPLX(-COS(PHASE)*RIN,-SIN(PHASE)*ROUT+A2)
          END DO
        END DO

C computes the negative frequency part of the filter
        DO I=NO2,NUMSAM*NBITS
          TRANSFERFUNC(I)=CONJG(TRANSFERFUNC(NUMSAM*NBITS+2-I))
        END DO

C multiplies the frequency responses
        DO I=1,NUMSAM*NBITS
          FFTOUT(I)=FFTOUT(I)*TRANSFERFUNC(I)
        END DO

        RETURN
        END
```

```
C==========================================================
       SUBROUTINE HARD_LIMITER
C this routine performs ideal hard limiting

        COMPLEX SIGNAL(16384)

        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C7/ SIGNAL

       DO I=1,NUMSAM*NBITS
         SIGNAL(I)=SIGNAL(I)/CABS(SIGNAL(I))
       END DO

       RETURN
       END
```

```
C===========================================================
        SUBROUTINE CLIPPER
C this routine performs ideal clipping
        COMPLEX SIGNAL(16384)

        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C7/ SIGNAL

        DO I=1,NUMSAM*NBITS
          IF(CABS(SIGNAL(I)).GT.SQRT(2.))
     $SIGNAL(I)=SQRT(2.)*SIGNAL(I)/CABS(SIGNAL(I))
        END DO

        RETURN
        END
```

```
C==========================================================
      SUBROUTINE CONST
      COMMON /C1/ PI,PI2
      PI=3.14159265358979
      PI2=2.0*PI
      RETURN
      END
```

```
C==============================================================
        SUBROUTINE INITIALIZE
C this routine initializes the variables
        INTEGER INPUTDATA(4096)
        COMPLEX SIGNAL(16384)
        INTEGER OUTPUTDATA(4096)
        DIMENSION TRANSFERFUNC(16384)
        COMPLEX FFTOUT(16384)
        REAL COLOREDNOISE3(128)

        COMMON /C2/ INPUTDATA
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C7/ SIGNAL
        COMMON /C9/ GAUSSNOISE
        COMMON /C12/ OUTPUTDATA
        COMMON /C24/ TRANSFERFUNC
        COMMON /C29/ FFTOUT
        COMMON /C61/ COLOREDNOISE3
        COMMON /C62/ COLOREDNOISEPOWER
        COLOREDNOISEPOWER=1.
        DO I=1,128
          COLOREDNOISE3(I)=1.
        END DO

        DO 100 I=1,NBITS
          INPUTDATA(I)=0
          OUTPUTDATA(I)=0
100     CONTINUE

        DO 110 I=1,NUMSAM*NBITS
          SIGNAL(I)=CMPLX(0.0,0.0)
          TRANSFERFUNC(I)=0.0
          FFTOUT(I)=CMPLX(0.0,0.0)
110     CONTINUE

        RETURN
        END
```

```
C==============================================================
        SUBROUTINE MLCS
C this routine generates the 4-ary symbol sequences from
C binary symbol sequences

        INTEGER INPUTDATA(4096)

        COMMON /C2/ INPUTDATA
        COMMON /C4/ NBITS

        NBITS=NBITS*2
        DO J=NBITS,2*NBITS-2
          INPUTDATA(J)=INPUTDATA(J+1-NBITS)
        END DO
        INPUTDATA(2*NBITS-1)=0
        INPUTDATA(2*NBITS)=0
        RETURN
        END
```

```
C=============================================================
        SUBROUTINE BPSK_MODULATOR
C this routine generates the BPSK signal which is subject to
C various realization imperfections
C PHASEOFFSET represents a possible phase imbalance  minimum=0.
C                                                    maximum=1.
C AMPLITUDEOFFSET represents a possible amplitude imbalance
C                                                    minimum=0.
C                                                    maximum=1.

        INTEGER INPUTDATA(4096)
        COMPLEX SIGNAL(16384)
        CHARACTER*1 CH
        CHARACTER*1 PLOTCONTROL

        COMMON /C1/ PI,PI2
        COMMON /C2/ INPUTDATA
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C7/ SIGNAL
        COMMON /C21/ BITLENGTH
        COMMON /C30/ PLOTCONTROL
        COMMON /C33/ PHASEOFFSET
        COMMON /C34/ AMPLITUDEOFFSET

        COMMON /C55/ SIGNALPOWER
        SIGNALPOWER=0.

        BITLENGTH=1.
        K=1
        DO 110 I=1,NBITS
          IF ( INPUTDATA(I).EQ.1 ) C=1.0
          IF ( INPUTDATA(I).EQ.0 ) C=-1.0
          DO 100 J=1,NUMSAM
            SIGNAL(K)=CMPLX(SQRT(2.0/BITLENGTH)*ABS(C+AMPLITUDEOFFSET))*
     $CMPLX(COS((-PI/2.)+(1.-PHASEOFFSET)*(PI/2.)*C)
     $,SIN((-PI/2.)+(1.-PHASEOFFSET)*(PI/2.)*C))
            SIGNALPOWER=SIGNALPOWER+REAL(SIGNAL(K)*CONJG(SIGNAL(K)))
            K=K+1
100       CONTINUE
110     CONTINUE

        RETURN
        END
```

```
C=====================================================================
        SUBROUTINE BPSK_C_DEMODULATOR
C this routine demodulates the BPSK signal with a filter matched to
C modulator filter
C the effect of AWGN is added analitically
C DEMOD_PHASEERROR represents a possible static phase error;minimum=0.
C                                                            maximum=1.

        CHARACTER*1 CH

        COMPLEX SIGNAL(16384)
        INTEGER INPUTDATA(4096)
        INTEGER OUTPUTDATA(4096)
        CHARACTER*1 REPORTCONTROL
        REAL COLOREDNOISE3(128)

        COMMON /C1/ PI,PI2
        COMMON /C2/ INPUTDATA
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C7/ SIGNAL
        COMMON /C12/ OUTPUTDATA
        COMMON /C18/ ERRORRATIO
        COMMON /C21/ BITLENGTH
        COMMON /C40/KSNR
        COMMON /C55/ SIGNALPOWER
        COMMON /C57/ DEMOD_PHASEERROR
        COMMON /C61/ COLOREDNOISE3
        COMMON /C62/ COLOREDNOISEPOWER
        COMMON /C63/ IDETECTIONTYPE
        COMMON /C64/ EQUIVNOISEBANDW
        COMMON /C66/ ISYSTEMTYPE
        COMMON /C68/ ERROR

        K=1
        ERROR=0.
        DO 200 I=1,NBITS
          SUM=0.0

C the procedure when other detection filters are used
          IF(IDETECTIONTYPE.EQ.2)THEN
           SUM=REAL(SIGNAL(K+(NUMSAM/2)))
           ENERGY2=SUM*SUM
           Y=SQRT(ENERGY2/10**(-REAL(KSNR)/10.)/EQUIVNOISEBANDW)
           K=K+NUMSAM
           GOTO 160
          ENDIF

          DO 150 L=1,NUMSAM
            SUM=SUM+REAL(SIGNAL(K)*CMPLX(SQRT(2.0/BITLENGTH),0.)*
     $CMPLX(COS(DEMOD_PHASEERROR*PI/2.)
     $,-SIN(DEMOD_PHASEERROR*PI/2.)))
            K=K+1
150       CONTINUE
        ENERGY2=(ABS(SUM)/REAL(NUMSAM))**2

C....Y=SQRT(ENERGY2*(1-RHO)/4*No)
        Y=SQRT(ENERGY2/2./COLOREDNOISEPOWER/10**(-REAL(KSNR)/10.))
160     Y=ERFCC(Y/SQRT(2.))/2.

C...included for nonlinear system
        IF(SUM*REAL(INPUTDATA(I)*2-1).LT.0.)THEN
```

```
      ERROR=ERROR+(1.-Y)
    ELSE
      ERROR=ERROR+Y
    ENDIF

200 CONTINUE
    ERROR=ERROR/REAL(NBITS)

    IF(ISYSTEMTYPE.NE.2)THEN
      WRITE(6,*)' SNR=',KSNR,'   PB(ERR)=',ERROR
      WRITE(1,*)' SNR=',KSNR,'   PB(ERR)=',ERROR
    ENDIF

    RETURN
    END
```

```
C=====================================================================
        SUBROUTINE QPSK_MODULATOR
C this routine generates the QPSK signal which is subject to
C various realization imperfections
C PHASEOFFSET represents a possible phase imbalance  minimum=0.
C                                                     maximum=1.
C AMPLITUDEOFFSET represents a possible amplitude imbalance
C                                                     minimum=0.
C                                                     maximum=1.

        INTEGER INPUTDATA(4096)
        DIMENSION PNYQUIST(16384)
        COMPLEX SIGNAL(16384)

        COMMON /C1/ PI,PI2
        COMMON /C2/ INPUTDATA
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C7/ SIGNAL
        COMMON /C21/ BITLENGTH
        COMMON /C33/ PHASEOFFSET
        COMMON /C34/ AMPLITUDEOFFSET

        INTEGER IBIT,QBIT
        REAL ITHETA
        BITLENGTH=1.
        K=1
        T=0.0

        DO 200  I=1,NBITS,2

C the in-phase bit is updated
          IBIT=INPUTDATA(I)*2-1

C the quadrature bit is updated
          QBIT=INPUTDATA(I+1)*2-1

         DO 190 M=1,2*NUMSAM
            SIGNAL(K)=ABS(REAL(IBIT)+AMPLITUDEOFFSET)*REAL(IBIT)*
     $CMPLX(COS(PHASEOFFSET*PI/4.),SIN(PHASEOFFSET*PI/4.))
            SIGNAL(K)=SIGNAL(K)+ABS(REAL(QBIT)-AMPLITUDEOFFSET)
     $*REAL(QBIT)*CMPLX(COS((-PI/2.)*(1.+(PHASEOFFSET/2.)))
     $,SIN((-PI/2.)*(1.+(PHASEOFFSET/2.))))

            SIGNAL(K)=SIGNAL(K)*CMPLX(SQRT(2.0/(2.0*BITLENGTH)))
            K=K+1
            T=T+SAMINT
190       CONTINUE
200     CONTINUE

        RETURN
        END
```

```
C=================================================================
      SUBROUTINE QPSK_C_DEMODULATOR
C this routine demodulates the BPSK signal with a filter matched to
C modulator filter
C the effect of AWGN is added analitically
C DEMOD_PHASEERROR represents a possible static phase error;minimum=0.
C                                                        maximum=1.
      COMPLEX SIGNAL(16384)
      INTEGER INPUTDATA(4096)
      INTEGER OUTPUTDATA(4096)
      REAL COLOREDNOISE3(128)

      COMMON /C1/ PI,PI2
      COMMON /C2/ INPUTDATA
      COMMON /C3/ NUMSAM
      COMMON /C4/ NBITS
      COMMON /C7/ SIGNAL
      COMMON /C12/ OUTPUTDATA
      COMMON /C21/ BITLENGTH
      COMMON /C30/ PLOTCONTROL
      COMMON /C40/KSNR
      COMMON /C57/ DEMOD_PHASEERROR
      COMMON /C61/ COLOREDNOISE3
      COMMON /C62/ COLOREDNOISEPOWER
      COMMON /C66/ ISYSTEMTYPE
      COMMON /C68/ ERROR

      K=1
      ERROR=0.

      DO 700 I=1,NBITS,2
        SUM1=0.0
        SUM2=0.0

        DO 680 M=1,2*NUMSAM
          SUM1=SUM1+REAL(SIGNAL(K)*CMPLX(SQRT(1./BITLENGTH))
     $*CMPLX(COS(DEMOD_PHASEERROR*PI/4.)
     $,-SIN(DEMOD_PHASEERROR*PI/4.))
     $*CMPLX(1.,0.))
          SUM2=SUM2+REAL(SIGNAL(K)*CMPLX(SQRT(1./BITLENGTH))
     $*CMPLX(COS(DEMOD_PHASEERROR*PI/4.)
     $,-SIN(DEMOD_PHASEERROR*PI/4.))
     $*CMPLX(0.,1.))
          K=K+1
680     CONTINUE

C...decision for I channel
      ENERGY=(ABS(SUM1)/REAL(2*NUMSAM))**2
      Y=SQRT(ENERGY*2./COLOREDNOISEPOWER/10.**(-REAL(KSNR)/10.))
      Y=ERFCC(Y/SQRT(2.))/2.

C...included for nonlinear system
       IF(SUM1*REAL(INPUTDATA(I)*2-1).LT.0.)THEN
         ERROR=ERROR+(1.-Y)
       ELSE
         ERROR=ERROR+Y
       ENDIF

C...decision for Q channel
      ENERGY=(ABS(SUM2)/REAL(2*NUMSAM))**2
      Y=SQRT(ENERGY*2./COLOREDNOISEPOWER/10.**(-REAL(KSNR)/10.))
      Y=ERFCC(Y/SQRT(2.))/2.
```

```
C...included for nonlinear system
        IF(SUM2*REAL(INPUTDATA(I+1)*2-1).LT.0.)THEN
          ERROR=ERROR+(1.-Y)
        ELSE
          ERROR=ERROR+Y
        ENDIF

700     CONTINUE
        ERROR=ERROR/REAL(NBITS)
        SERROR=(1.-ERROR)**2
        SERROR=1.-SERROR

        IF(ISYSTEMTYPE.NE.2)THEN
          WRITE(6,*)' SNR=',KSNR,'   PB(ERR)=',ERROR
          WRITE(1,*)' SNR=',KSNR,'   PB(ERR)=',ERROR
        ENDIF

        RETURN
        END
```

```
C===========================================================
        SUBROUTINE OKQPSK_MODULATOR
C this routine generates the OKQPSK signal

         INTEGER INPUTDATA(4096)
        COMPLEX SIGNAL(16384)

        COMMON /C1/ PI,PI2
        COMMON /C2/ INPUTDATA
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C7/ SIGNAL
        COMMON /C21/ BITLENGTH

        REAL IBIT,QBIT
        BITLENGTH=1.
        K=1

        QBIT=REAL(INPUTDATA(NBITS)*2-1)

        DO I=1,NBITS,2

C the in-phase bit is updated
          IBIT=REAL(INPUTDATA(I)*2-1)
          DO  M=1,NUMSAM
            SIGNAL(K)=CMPLX(SQRT(2./2./BITLENGTH))*CMPLX(IBIT,-QBIT)
            K=K+1
          END DO

C the quadrature bit is updated
          QBIT=INPUTDATA(I+1)*2-1
          DO M=1,NUMSAM
            SIGNAL(K)=CMPLX(SQRT(2./2./BITLENGTH))*CMPLX(IBIT,-QBIT)
            K=K+1
          END DO

        END DO
        RETURN
        END
```

```
C===================================================================
        SUBROUTINE OKQPSK_C_DEMODULATOR
C this routine demodulates the OKQPSK signal with a filter matched to
C modulator filter
C the effect of AWGN is added analitically
C DEMOD_PHASEERROR represents a possible static phase error;minimum=0.
C                                                            maximum=1.

        COMPLEX SIGNAL(16384)
        INTEGER INPUTDATA(4096)
        INTEGER OUTPUTDATA(4096)
        REAL COLOREDNOISE3(128)

        COMMON /C1/ PI,PI2
        COMMON /C2/ INPUTDATA
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C7/ SIGNAL
        COMMON /C12/ OUTPUTDATA
        COMMON /C21/ BITLENGTH
        COMMON /C30/ PLOTCONTROL
        COMMON /C40/KSNR
         COMMON /C57/ DEMOD_PHASEERROR
        COMMON /C61/ COLOREDNOISE3
        COMMON /C62/ COLOREDNOISEPOWER
         COMMON /C66/ ISYSTEMTYPE
         COMMON /C68/ ERROR

        ERROR=0.
        K=1
        SUM2=0.
        DO I=1,NBITS,2
          SUM1=0.
          DO M=1,NUMSAM
            SUM1=SUM1+REAL(SIGNAL(K)
     $*CMPLX(COS(DEMOD_PHASEERROR*PI/4.)
     $,-SIN(DEMOD_PHASEERROR*PI/4.))
     $*CMPLX(SQRT(2./2./BITLENGTH),0.))
            SUM2=SUM2+REAL(SIGNAL(K)
     $*CMPLX(COS(DEMOD_PHASEERROR*PI/4.)
     $,-SIN(DEMOD_PHASEERROR*PI/4.))
     $*CMPLX(0.,SQRT(2./2./BITLENGTH)))
            K=K+1
          END DO

          IF(I.NE.1)THEN
C...decision about SUM2
            ENERGY=(ABS(SUM2)/REAL(2*NUMSAM))**2
            Y=SQRT(ENERGY*2./COLOREDNOISEPOWER/10.**(-REAL(KSNR)/10.))
            Y=ERFCC(Y/SQRT(2.))/2.
C...included for nonlinear system
         IF(SUM2*REAL(INPUTDATA(I-1)*2-1).LT.0.)THEN
          ERROR=ERROR+(1.-Y)
        ELSE
          ERROR=ERROR+Y
        ENDIF

          ELSE
            SUM2LAST=SUM2
          ENDIF

          SUM2=0.
```

```
          DO M=1,NUMSAM
            SUM1=SUM1+REAL(SIGNAL(K)
     $*CMPLX(COS(DEMOD_PHASEERROR*PI/4.)
     $,-SIN(DEMOD_PHASEERROR*PI/4.))
     $*CMPLX(SQRT(2./2./BITLENGTH),0.))
            SUM2=SUM2+REAL(SIGNAL(K)
     $*CMPLX(COS(DEMOD_PHASEERROR*PI/4.)
     $,-SIN(DEMOD_PHASEERROR*PI/4.))
     $*CMPLX(0.,SQRT(2./2./BITLENGTH)))
            K=K+1
          END DO

C...decision about SUM1
          ENERGY=(ABS(SUM1)/REAL(2*NUMSAM))**2
          V=SQRT(ENERGY*2./COLOREDNOISEPOWER/10.**(-REAL(KSNR)/10.))
          V=ERFCC(V/SQRT(2.))/2.
C...included for nonlinear system
         IF(SUM1*REAL(INPUTDATA(I)*2-1).LT.0.)THEN
           ERROR=ERROR+(1.-V)
         ELSE
           ERROR=ERROR+V
         ENDIF
        END DO

        SUM2=SUM2+SUM2LAST
C...decision about SUM2
        ENERGY=(ABS(SUM2)/REAL(2*NUMSAM))**2
        V=SQRT(ENERGY*2./COLOREDNOISEPOWER/10.**(-REAL(KSNR)/10.))
        V=ERFCC(V/SQRT(2.))/2.
C...included for nonlinear system
         IF(SUM2*REAL(INPUTDATA(NBITS)*2-1).LT.0.)THEN
           ERROR=ERROR+(1.-V)
         ELSE
           ERROR=ERROR+V
         ENDIF

        ERROR=ERROR/REAL(NBITS)
        SERROR=(1.-ERROR)**2
        SERROR=1.-SERROR

         IF(ISYSTEMTYPE.NE.2)THEN
           WRITE(6,*)' SNR=',KSNR,'   PB(ERR)=',ERROR
           WRITE(1,*)' SNR=',KSNR,'   PB(ERR)=',ERROR
         ENDIF

        RETURN
        END
```

```
C=================================================================
        SUBROUTINE MSK_MODULATOR
C this routine generates the BPSK signal which is subject to
C various realization imperfections

        INTEGER INPUTDATA(4096)
        COMPLEX SIGNAL(16384)

        COMMON /C1/ PI,PI2
        COMMON /C2/ INPUTDATA
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C7/ SIGNAL
        COMMON /C20/ SAMINT
        COMMON /C21/ BITLENGTH

        REAL IBIT,QBIT,M1,M2

        BITLENGTH=1.
        K=1
        QBIT=REAL(INPUTDATA(NBITS)*2-1)
        DO I=1,NBITS,2

C the in-phase bit is updated
          IBIT=REAL(INPUTDATA(I)*2-1)
          DO M=1,NUMSAM
        M1=COS((PI2/4./BITLENGTH)*REAL(M+MOD(I-1,2)*NUMSAM)*SAMINT)*IBIT
        M2=SIN((PI2/4./BITLENGTH)*REAL(M+MOD(I,2)*NUMSAM)*SAMINT)*QBIT
            SIGNAL(K)=CMPLX(SQRT(2./BITLENGTH))*CMPLX(M1,-M2)
            K=K+1
          END DO

C the quadrature bit is updated
          QBIT=INPUTDATA(I+1)*2-1
          DO M=1,NUMSAM
        M1=COS((PI2/4./BITLENGTH)*REAL(M+MOD(I,2)*NUMSAM)*SAMINT)*IBIT
        M2=SIN((PI2/4./BITLENGTH)*REAL(M+MOD(I-1,2)*NUMSAM)*SAMINT)*QBIT
            SIGNAL(K)=CMPLX(SQRT(2./BITLENGTH))*CMPLX(M1,-M2)
            K=K+1
          END DO
        END DO

        RETURN
        END
```

```
C================================================================
        SUBROUTINE MSK_C_DEMODULATOR
C this routine demodulates the BPSK signal with a filter matched to
C modulator filter
C the effect of AWGN is added analitically
C DEMOD_PHASEERROR represents a possible static phase error;minimum=0.
C                                                         maximum=1.

        COMPLEX SIGNAL(16384)
        INTEGER INPUTDATA(4096)
        INTEGER OUTPUTDATA(4096)
        REAL COLOREDNOISE3(128)

        COMMON /C1/ PI,PI2
        COMMON /C2/ INPUTDATA
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C7/ SIGNAL
        COMMON /C12/ OUTPUTDATA
        COMMON /C20/ SAMINT
        COMMON /C21/ BITLENGTH
        COMMON /C30/ PLOTCONTROL
        COMMON /C40/KSNR
        COMMON /C57/ DEMOD_PHASEERROR
        COMMON /C61/ COLOREDNOISE3
        COMMON /C62/ COLOREDNOISEPOWER
        COMMON /C66/ ISYSTEMTYPE
        COMMON /C68/ ERROR

        REAL M1,M2

        ERROR=0.
        K=1
        SUM2=0.

        DO I=1,NBITS,2

          SUM1=0.
          DO M=1,NUMSAM
        M1=COS((PI2/4./BITLENGTH)*REAL(M+MOD(I-1,2)*NUMSAM)*SAMINT)
        M2=SIN((PI2/4./BITLENGTH)*REAL(M+MOD(I,2)*NUMSAM)*SAMINT)
            SUM1=SUM1+REAL(SIGNAL(K)
     $*CMPLX(COS(DEMOD_PHASEERROR*PI/4.)
     $,-SIN(DEMOD_PHASEERROR*PI/4.))
     $*CMPLX(SQRT(2./BITLENGTH)*M1,0.))
            SUM2=SUM2+REAL(SIGNAL(K)
     $*CMPLX(COS(DEMOD_PHASEERROR*PI/4.)
     $,-SIN(DEMOD_PHASEERROR*PI/4.))
     $*CMPLX(0.,SQRT(2./BITLENGTH)*M2))
            K=K+1
          END DO

          IF(I.NE.1)THEN
C...decision about SUM2
            ENERGY=(ABS(SUM2)/REAL(2*NUMSAM))**2
            Y=SQRT(ENERGY*2./COLOREDNOISEPOWER/10.**(-REAL(KSNR)/10.))
            Y=ERFCC(Y/SQRT(2.))/2.
C...included for nonlinear system
         IF(SUM2*REAL(INPUTDATA(I-1)*2-1).LT.0.)THEN
          ERROR=ERROR+(1.-Y)
         ELSE
          ERROR=ERROR+Y
```

```
          ENDIF
            ELSE
              SUM2LAST=SUM2
            ENDIF

            SUM2=0.

            DO M=1,NUMSAM
          M1=COS((PI2/4./BITLENGTH)*REAL(M+MOD(I,2)*NUMSAM)*SAMINT)
          M2=SIN((PI2/4./BITLENGTH)*REAL(M+MOD(I-1,2)*NUMSAM)*SAMINT)
              SUM1=SUM1+REAL(SIGNAL(K)
      $*CMPLX(COS(DEMOD_PHASEERROR*PI/4.)
      $,-SIN(DEMOD_PHASEERROR*PI/4.))
      $*CMPLX(SQRT(2./BITLENGTH)*M1,0.))
              SUM2=SUM2+REAL(SIGNAL(K)
      $*CMPLX(COS(DEMOD_PHASEERROR*PI/4.)
      $,-SIN(DEMOD_PHASEERROR*PI/4.))
      $*CMPLX(0.,SQRT(2./BITLENGTH)*M2))
              K=K+1
            END DO

C...decision about SUM1
          ENERGY=(ABS(SUM1)/REAL(2*NUMSAM))**2
          Y=SQRT(ENERGY*2./COLOREDNOISEPOWER/10.**(-REAL(KSNR)/10.))
          Y=ERFCC(Y/SQRT(2.))/2.
C...included for nonlinear system
           IF(SUM1*REAL(INPUTDATA(I)*2-1).LT.0.)THEN
             ERROR=ERROR+(1.-Y)
           ELSE
             ERROR=ERROR+Y
           ENDIF

          END DO

          SUM2=SUM2+SUM2LAST
C...decision about SUM2
          ENERGY=(ABS(SUM2)/REAL(2*NUMSAM))**2
          Y=SQRT(ENERGY*2./COLOREDNOISEPOWER/10.**(-REAL(KSNR)/10.))
          Y=ERFCC(Y/SQRT(2.))/2.
C...included for nonlinear system
           IF(SUM2*REAL(INPUTDATA(NBITS)*2-1).LT.0.)THEN
             ERROR=ERROR+(1.-Y)
           ELSE
             ERROR=ERROR+Y
           ENDIF

          ERROR=ERROR/REAL(NBITS)
          SERROR=(1.-ERROR)**2
          SERROR=1.-SERROR

           IF(ISYSTEMTYPE.NE.2)THEN
             WRITE(6,*)' SNR=',KSNR,'   PB(ERR)=',ERROR
             WRITE(1,*)' SNR=',KSNR,'   PB(ERR)=',ERROR
           ENDIF
          RETURN
          END
```

```
C=============================================================
        SUBROUTINE FFT(V,INLENGTH,CONTROL)
C this routine performs Fourier transformation
C the length of the sequence is a power of 2  ; minimum=   32
C                                               maximum=16384
        CHARACTER*1 CH

        CHARACTER*1 PLOTCONTROL
        COMPLEX FFTOUT(16384)

        COMMON /C1/ PI,PI2
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C20/ SAMINT
        COMMON /C23/ SAMFREQ
        COMMON /C29/ FFTOUT
        COMMON /C30/ PLOTCONTROL

        COMPLEX U,W,T,V(16384)
        DIMENSION XRE(16384),XIM(16384),XMAG(16384)

        M=0
        IF ( NUMSAM*NBITS.EQ.32 ) M=5
        IF ( NUMSAM*NBITS.EQ.64 ) M=6
        IF ( NUMSAM*NBITS.EQ.128 ) M=7
        IF ( NUMSAM*NBITS.EQ.256 ) M=8
        IF ( NUMSAM*NBITS.EQ.512 ) M=9
        IF ( NUMSAM*NBITS.EQ.1024 ) M=10
        IF ( NUMSAM*NBITS.EQ.2048 ) M=11
        IF ( NUMSAM*NBITS.EQ.4096 ) M=12
        IF ( NUMSAM*NBITS.EQ.8192 ) M=13
        IF ( NUMSAM*NBITS.EQ.16384 ) M=14
        IF ( M.EQ.0 ) THEN
           PRINT*,' error in fft ... M=0'
           STOP
        ENDIF
        N=2**M

        DO 100 I=1,N
          FFTOUT(I)=V(I)
 100    CONTINUE

        NV2=N/2
        NM1=N-1
        J=1
        DO 8 I=1,NM1
          IF ( I.GE.J ) GOTO 5
          T=FFTOUT(J)
          FFTOUT(J)=FFTOUT(I)
          FFTOUT(I)=T
5         K=NV2
6         IF ( K.GE.J ) GOTO 7
          J=J-K
          K=K/2
          GOTO 6
7         J=J+K
8       CONTINUE
        DO 20 L=1,M
          LE=2**L
          LE1=LE/2
          U=(1.0,0.0)
          W=CMPLX(COS(PI/FLOAT(LE1)),-SIN(PI/FLOAT(LE1)))
```

```
      DO 20 J=1,LE1
        DO 10 I=J,N,LE
          IP=I+LE1
          T=FFTOUT(IP)*U
          FFTOUT(IP)=FFTOUT(I)-T
          FFTOUT(I)=FFTOUT(I)+T
10      CONTINUE
      U=U*W
20    CONTINUE
      RETURN
      END
```

```
C==============================================================
        SUBROUTINE LINEAR_AMP_DIST
C generates the transfer function for
C linear amplitude distortion

        COMPLEX TRANSFERFUNC(16384),ATTENUATION
        COMPLEX FFTOUT(16384)

        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C23/ SAMFREQ
        COMMON /C29/ FFTOUT
        COMMON /C38/ ALIN

        N01=NUMSAM*NBITS/2+1
        N02=N01+1
        AI=ALIN*SAMFREQ/FLOAT(NUMSAM*NBITS)

C the frequency response for positive frequencies is calculated
        DO I=1,N01
          AJ=REAL(I-1)*AI/20.
          TRANSFERFUNC(I)=CMPLX(10.**AJ)
        END DO

C the frequency response for negative frequencies is calculated
        DO I=N02,NUMSAM*NBITS
          TRANSFERFUNC(I)=(CMPLX(1.)/TRANSFERFUNC(NUMSAM*NBITS+2-I))
        END DO

C the signal is attenuated to prevent the amplification
C introduced by the linear amplitude distortion
        ATTENUATION=CMPLX(10.**(ALIN*SAMFREQ/2./20.))

        DO I=1,NUMSAM*NBITS
          FFTOUT(I)=FFTOUT(I)*TRANSFERFUNC(I)/ATTENUATION
        END DO

        RETURN
        END
```

```
C=====================================================================
        SUBROUTINE PARABOLIC_AMP_DIST
C generates the transfer function for
C parabolic amplitude distortion

        COMPLEX TRANSFERFUNC(16384)
        COMPLEX FFTOUT(16384)

        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C23/ SAMFREQ
        COMMON /C29/ FFTOUT
        COMMON /C39/ APAR

        NO1=NUMSAM*NBITS/2+1
        NO2=NO1+1
        AI=APAR*((SAMFREQ/FLOAT(NUMSAM*NBITS))**2.)

C the frequency response for positive frequencies is calculated
        DO I=1,NO1
          AJ=(REAL(I-1)**2.)*AI/20.
          TRANSFERFUNC(I)=CMPLX(10.**AJ)
        END DO

C the frequency response for negative frequencies is calculated
        DO I=NO2,NUMSAM*NBITS
          TRANSFERFUNC(I)=TRANSFERFUNC(NUMSAM*NBITS+2-I)
        END DO

C the frequency responses are multiplied
        DO I=1,NUMSAM*NBITS
          FFTOUT(I)=FFTOUT(I)*TRANSFERFUNC(I)
        END DO

        RETURN
        END
```

```
C==============================================================
        SUBROUTINE RIPPLE_AMP_DIST_SIN
C generates the transfer function for
C ripple amplitude distortion type I

        COMPLEX TRANSFERFUNC(16384)
        COMPLEX FFTOUT(16384)

        COMMON /C1/ PI,PI2
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C23/ SAMFREQ
        COMMON /C29/ FFTOUT
        COMMON /C80/ ARS,NSRIP

        NO1=NUMSAM*NBITS/2+1
        NO2=NO1+1
        AJ=SAMFREQ/REAL(NUMSAM*NBITS)
        C1=1./2./REAL(NSRIP)

C the frequency response for positive frequencies is calculated
        DO I=1,NO1
          AZ=ARS*SIN(2.*PI*(REAL(I-1)*AJ)/C1)/20.
          TRANSFERFUNC(I)=CMPLX(10.**AZ)
        END DO

C the frequency response for negative frequencies is calculated
        DO I=NO2,NUMSAM*NBITS
          TRANSFERFUNC(I)=CMPLX(1.)/TRANSFERFUNC(NUMSAM*NBITS+2-I)
        END DO

C the frequency responses are multiplied
        DO I=1,NUMSAM*NBITS
          FFTOUT(I)=FFTOUT(I)*TRANSFERFUNC(I)
        END DO
        RETURN
        END
```

```
C=================================================================
        SUBROUTINE RIPPLE_AMP_DIST_COS
C generates the transfer function for
C ripple amplitude distortion type II

        COMPLEX TRANSFERFUNC(16384)
        COMPLEX FFTOUT(16384)

        COMMON /C1/ PI,PI2
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C23/ SAMFREQ
        COMMON /C29/ FFTOUT
        COMMON /C41/ ARC,NCRIP

        NO1=NUMSAM*NBITS/2+1
        NO2=NO1+1
        AJ=SAMFREQ/REAL(NUMSAM*NBITS)
        C1=1./2./REAL(NCRIP)

C the frequency response for positive frequencies is calculated
        DO I=1,NO1
          AZ=ARC*COS(2.*PI*(REAL(I-1)*AJ)/C1)/20.
          TRANSFERFUNC(I)=CMPLX(10.**AZ)
        END DO

C the frequency response for negative frequencies is calculated
        DO I=NO2,NUMSAM*NBITS
          TRANSFERFUNC(I)=TRANSFERFUNC(NUMSAM*NBITS+2-I)
        END DO

C the frequency responses are multiplied
        DO I=1,NUMSAM*NBITS
          FFTOUT(I)=FFTOUT(I)*TRANSFERFUNC(I)
        END DO
        RETURN
        END
```

```
C=================================================================
        SUBROUTINE LINEAR_GROUP_DELAY
C generates the transfer function for
C linear group delay

        COMPLEX TRANSFERFUNC(16384)
        COMPLEX FFTOUT(16384)

        COMMON /C1/ PI,PI2
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C23/ SAMFREQ
        COMMON /C29/ FFTOUT
        COMMON /C42/ GLIN

        NO1=NUMSAM*NBITS/2+1
        NO2=NO1+1

C the frequency response for positive frequencies is calculated
        DO I=1,NO1
          AI=FLOAT(I-1)*SAMFREQ/FLOAT(NUMSAM*NBITS)
          PHI=PI*(GLIN/10.**3)*(AI**2)
          TRANSFERFUNC(I)=CMPLX(COS(PHI),-SIN(PHI))
        END DO

C the frequency response for negative frequencies is calculated
        DO I=NO2,NUMSAM*NBITS
          TRANSFERFUNC(I)=TRANSFERFUNC(NUMSAM*NBITS+2-I)
        END DO


C the frequency responses are multiplied
        DO I=1,NUMSAM*NBITS
          FFTOUT(I)=FFTOUT(I)*TRANSFERFUNC(I)
        END DO

        RETURN
        END
```

```
C==============================================================
        SUBROUTINE PARABOLIC_GROUP_DELAY
C generates the transfer function for
C parabolic group delay

        COMPLEX TRANSFERFUNC(16384)
        COMPLEX FFTOUT(16384)

        COMMON /C1/ PI,PI2
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C23/ SAMFREQ
        COMMON /C29/ FFTOUT
        COMMON /C43/ GPAR

        N01=NUMSAM*NBITS/2+1
        N02=N01+1

C the frequency response for positive frequencies is calculated
        DO I=1,N01
          AI=FLOAT(I-1)*SAMFREQ/FLOAT(NUMSAM*NBITS)
          PHI=2.*PI/3.*(GPAR/10.**3)*(AI**3)
          TRANSFERFUNC(I)=CMPLX(COS(PHI),-SIN(PHI))
        END DO

C the frequency response for negative frequencies is calculated
        DO I=N02,NUMSAM*NBITS
          TRANSFERFUNC(I)=CONJG(TRANSFERFUNC(NUMSAM*NBITS+2-I))
        END DO

C the frequency responses are multiplied
        DO I=1,NUMSAM*NBITS
          FFTOUT(I)=FFTOUT(I)*TRANSFERFUNC(I)
        END DO

        RETURN
        END
```

```
C==================================================================
      SUBROUTINE RIPPLE_GROUP_DELAY_SIN
C generates the transfer function for
C ripple group delay type I

      COMPLEX TRANSFERFUNC(16384)
      COMPLEX FFTOUT(16384)

      COMMON /C1/ PI,PI2
      COMMON /C3/ NUMSAM
      COMMON /C4/ NBITS
      COMMON /C23/ SAMFREQ
      COMMON /C29/ FFTOUT
      COMMON /C44/ GRS,NGSRIP

      NO1=NUMSAM*NBITS/2+1
      NO2=NO1+1
      AJ=SAMFREQ/REAL(NUMSAM*NBITS)
      C1=1./2./REAL(NGSRIP)

C the frequency response for positive frequencies is calculated
      DO I=1,NO1
        PHI=(GRS*C1/(10.**3))*SIN(2.*PI*FLOAT(I-1)*AJ/C1)
        TRANSFERFUNC(I)=CMPLX(COS(PHI),-SIN(PHI))
      END DO

C the frequency response for negative frequencies is calculated
      DO I=NO2,NUMSAM*NBITS
        TRANSFERFUNC(I)=CONJG(TRANSFERFUNC(NUMSAM*NBITS+2-I))
      END DO

C the frequency responses are multiplied
      DO I=1,NUMSAM*NBITS
        FFTOUT(I)=FFTOUT(I)*TRANSFERFUNC(I)
      END DO

      RETURN
      END
```

```
C=============================================================
        SUBROUTINE RIPPLE_GROUP_DELAY_COS
C generates the transfer function for
C ripple group delay type II

        COMPLEX TRANSFERFUNC(16384)
        COMPLEX FFTOUT(16384)

        COMMON /C1/ PI,PI2
        COMMON /C3/ NUMSAM
        COMMON /C4/ NBITS
        COMMON /C23/ SAMFREQ
        COMMON /C29/ FFTOUT
        COMMON /C45/ GRC,NGCRIP

        NO1=NUMSAM*NBITS/2+1
        NO2=NO1+1
        AJ=SAMFREQ/REAL(NUMSAM*NBITS)
        C1=1./2./REAL(NGCRIP)

C the frequency response for positive frequencies is calculated
        DO I=1,NO1
          PHI=(GRC*C1/(10.**3))*COS(2.*PI*FLOAT(I-1)*AJ/C1)
          TRANSFERFUNC(I)=CMPLX(COS(PHI),SIN(PHI))
        END DO

C the frequency response for negative frequencies is calculated
        DO I=NO2,NUMSAM*NBITS
          TRANSFERFUNC(I)=TRANSFERFUNC(NUMSAM*NBITS+2-I)
        END DO

C the frequency responses are multiplied
        DO I=1,NUMSAM*NBITS
          FFTOUT(I)=FFTOUT(I)*TRANSFERFUNC(I)
        END DO

        RETURN
        END
```

```
C==============================================================
        SUBROUTINE PRBSGENERATOR
C generates the pseudo random bit sequence

        CHARACTER*1 CH
        INTEGER INPUTDATA(4096)
        CHARACTER*1 PRINTCONTROL
        CHARACTER*1 REPORTCONTROL
        INTEGER REGISTER(24)

        COMMON /C2/ INPUTDATA
        COMMON /C4/ NBITS
        COMMON /C13/ PRINTCONTROL
        COMMON /C14/ REGISTER
        COMMON /C15/ NREG
        COMMON /C28/ REPORTCONTROL
        COMMON /C46/ TAPPOSITION
        COMMON /C47/ NTAP

        INTEGER TAPPOSITION(4),NTAP,SUM
        INTEGER SEQUENCELENGTH,SEQUENCE(16384)
        INDEX=1
        DO I=1,13
         INDEX=INDEX*2
         IF(INDEX.EQ.NBITS)NREG=I
        END DO
        REGISTER(1)=1
        DO I=2,NREG
        REGISTER(I)=ABS(REGISTER(I-1)-1)
        END DO
        DO 40 I=1,4
        TAPPOSITION(I)=0
40      CONTINUE

C   First tap is always connected.
        TAPPOSITION(1)=1

C   Determine the new tap positions.
        IF ( NREG.EQ.2 ) THEN
          NTAP=2
          TAPPOSITION(2)=2
          GOTO 100
        ENDIF

        IF ( NREG.EQ.3 ) THEN
          NTAP=2
          TAPPOSITION(2)=3
          GOTO 100
        ENDIF

        IF ( NREG.EQ.4 ) THEN
          NTAP=2
          TAPPOSITION(2)=4
          GOTO 100
        ENDIF

        IF ( NREG.EQ.5 ) THEN
          NTAP=2
          TAPPOSITION(2)=4
          GOTO 100
        ENDIF
```

```
      IF ( NREG.EQ.6 ) THEN
        NTAP=2
        TAPPOSITION(2)=6
        GOTO 100
      ENDIF

      IF ( NREG.EQ.7 ) THEN
        NTAP=2
        TAPPOSITION(2)=7
        GOTO 100
      ENDIF

      IF ( NREG.EQ.8 ) THEN
        NTAP=4
        TAPPOSITION(2)=5
        TAPPOSITION(3)=6
        TAPPOSITION(4)=7
        GOTO 100
      ENDIF

      IF ( NREG.EQ.9 ) THEN
        NTAP=2
        TAPPOSITION(2)=6
        GOTO 100
      ENDIF

      IF ( NREG.EQ.10 ) THEN
        NTAP=2
        TAPPOSITION(2)=8
        GOTO 100
      ENDIF

      IF ( NREG.EQ.11 ) THEN
        NTAP=2
        TAPPOSITION(2)=10
        GOTO 100
      ENDIF

      IF ( NREG.EQ.12 ) THEN
        NTAP=4
        TAPPOSITION(2)=7
        TAPPOSITION(3)=9
        TAPPOSITION(4)=12
        GOTO 100
      ENDIF

      IF ( NREG.EQ.13 ) THEN
        NTAP=4
        TAPPOSITION(2)=10
        TAPPOSITION(3)=11
        TAPPOSITION(4)=13
        GOTO 100
      ENDIF

100   DO  J=1,2**NREG-1
        INPUTDATA(J)=REGISTER(1)

        SUM=REGISTER(1)
        DO 120 I=2,NTAP

          IF ( SUM.EQ.REGISTER(TAPPOSITION(I)) ) GOTO 110
          SUM=1
```

```
          GOTO120
110       SUM=0
C   end {exor}
120     CONTINUE
C   end {sum of the taps}
        DO 130 I=1,NREG-1
          REGISTER(I)=REGISTER(I+1)
130     CONTINUE
        REGISTER(NREG)=SUM
      END DO
      INPUTDATA(2**NREG)=0
      RETURN
      END
```

```
C==================================================================
      FUNCTION ERFCC(X)
C computes the complementary error function
C with Chebychev normalization

      Z=ABS(X)
      T=1./(1.+0.5*Z)
      ERFCC=T*EXP(-Z*Z-1.26551223+T*(1.00002368+T*(.37409196+
     *    T*(.09678418+T*(-.18628806+T*(.27886807+T*(-1.13520398+
     *    T*(1.48851587+T*(-.82215223+T*.17087277)))))))))
      IF (X.LT.0.) ERFCC=2.-ERFCC

      RETURN
      END
```

```
C=========================================================================
              SUBROUTINE TIME_BEGIN
!           CPU AND ELAPSED TIME COMPUTATION (together TIME_END)
!           PROGRAM
        external lib$init_timer
        external lib$get_vm
        COMMON HANDLE
!
        integer*4 dyn_array_adr
        nlongwords = 50
        handle = 0 !zaman bilgisi virtual bellekde saklanacak
!
        CALL lib$get_vm(nlongwords*4,dyn_array_adr)!virtual bellek dizayni
!
        CALL lib$init_timer(handle) !timer set edildi
!
              RETURN
                     END
              SUBROUTINE TIME_END
!           CPU AND ELAPSED TIME COMPUTATION (together TIME_BEGIN)
!           PROGRAM
        external lib$show_timer
        external lib$free_timer
              COMMON HANDLE
        CALL lib$show_timer(handle) !isin son durumu ekrana yazildi
!
        CALL lib$free_timer(handle) !yeni is icin timer serbest birakildi
              RETURN
                   END
```

BIBLIOGRAPHY

[1] Marsan J.J.,Benedetto S. ,Biglieri E., Castellani V., Elia M., Presti L.,Pent M., "Digital Simulation of Communication Systems with TOPSIM III," IEEE Journal on Selected Areas in Communications, Vol SAC-2 No1, pp29-42 , January 1984

[2] Modestino J. W. ,Matis K.R. , "Interactive Simulation of Digital Communication Systems," IEEE Journal on Selected Areas in Communications, Vol SAC-2 No1, pp51-76, January 1984

[3] Proakis J.G., Digital Communications, Mc Graw-Hill Inc. , 1983

[4] Ziemer R.E., Peterson R.L., Digital Communications and Spread Spectrum Systems, MacMillan Publishing Company New York,1985

[5] Jones J.J., "Filter Distortion and Intersymbol Interference Effects on PSK signals," IEEE Transactions on Communication Technology, Vol Com-19 No2, pp120-132 , April 1971

[6] Celebiler M.I.,Shimbo O., "The Probability of Error due to Intersymbol Interference and Gausian Noise in Digital Communication Systems " IEEE Transactions on Communication Technology, Vol Com-19 No2, pp113-119 , April 1971

[7] Lugannani R., "Intersymbol Interference and Probability of Error in Digital Systems," IEEE Transactions on Information Theory, Vol IT-15,pp682-688,November 1969

[8] Ho E.Y.,Yeh Y.S., "A New Approach For Evaluating Error Probability in the Presence of Intersymbol Interference and Additive Gausian Noise," Bell System Technical Journal,pp2249-2265, November 1970

[9] Murty M. S. , " CONSIM: A Software for the Computer Aided Design and Analysis of Communication Systems ," Ph. D. Thesis, University of Ottawa ,1983

[10] Sunde E.D., Communication Systems Engineering Theory, John Wiley & Sons Inc.,pp265-307,1969

[11] Feher K., Digital Communications: Satellite/Earth Station Engineering, Prentice Hall Inc., 1983

[12] Morais D.H., Sewerinson A., Feher K., "The Effects of Amplitude and Delay Slope Components of Frequency Selective Fading on QPSK, OK-QPSK and 8PSK Systems," IEEE Transactions on Communications, Vol COM-27 ,No 12,pp1849-1853, December 1979

[13] Oppenheim A. V., Schafer R.W. Digital Signal Processing Prentice Hall Inc. , 1976

[14] Jeruchim M.C., "A Survey of Interference Problems and Applications to Geostationary Satellite Networks," Proceedings of IEEE, Vol-65,No3, pp317-331, March 1977

[15] Benedetto S. ,Biglieri E., Castellani V., " Combined Effects of Intersymbol Interference in M-ary CPSK Systems," IEEE Transactions on Communication Technology, Vol Com-21 No9,pp997-1008, September 1973

[16] Shimbo O.,Fang R., " Unified Analysis of a Class of Digital Systems in Additive Noise and Interference," IEEE Transactions on Communication Technology, Vol Com-21 No10,pp1075-1091, October 1973

[17] Benedetto S.,De Vincentiis G.,Luvison A., "Error Probability in the Presence of Intersymbol Interference andd Additive Noise for Multilevel Digital Signals," IEEE Transactions on Communication Technology, Vol Com-21 No3 ,pp181-190, March 1973

[18] Yao K.,Tobin R.M., "Moment Space Upper and Lower Error Bounds for Digital Systems,with Intersymbol Interference," IEEE Transactions on Information Theory, Vol IT-22,pp65-74,January1976

[19] Jeruchim M.C.,Blum R., Modelling Nonlinear Amplifiers for

Communication Simulation,pp1469-1472 , 1989

[20] Davisson L.D.,Milstein L.B. , "On The Performance of Digital Communication Systems With Bandpass Limiters-Part1:One Link System," IEEE Transactions on Communication Technology, Vol Com-20 No5,pp972-976, October1972

[21] Bhargava V.K. , Haccoun D. , Matyas R. , Nuspl P. P. Digital Communications By Satellite,John Wiley & Sons,1981

[22] Benedetto S. ,Biglieri E.,Daffara R., "Modelling of Nonlinear Satellite Links ," IEEE Transactions on Aerospace and Electronic Systems- A Volterra Series Approach,Vol AES-15,No4, pp494-506,July 1979

[23] Benedetto S. ,Biglieri E., Castellani V.,Digital Transmission Theory, Prentice Hall Inc.,1987

[24] Jeruchim M.C., "Techniques for Bit Error Rate Estimation,"IEEE Journal on Selected Areas in Communications, Vol SAC-2 No1,pp153-170, January 1984

[25] Palmer L.C., "Computer Modelling and Simulation of Communications Satellite Channels," IEEE Journal on Selected Areas in Communications, Vol SAC-2 No1,pp89-102 , January1984

[26] Pinto E.L.,Brandao J.C., "On the Efficient Use of Computer Simulated Digital Signals to Evaluate Performance Parameters," IEEE Journal on Selected Areas in Communications, Vol SAC-6 No1, pp29-42, January1988

[27] Durukan E., "Computer Simulation of Digital Communication Systems," M.S. Thesis , Boğaziçi University,1989

[28] Prabhu V. K., "Some Considerations of Error Bounds in Digital Systems, "Bell System Technical Journal,Vol-50, pp3127-3151,December1971

[29] Golub G.H., Welsh J.H., "Calculation of Gauss Quadrature Rules," Mathematical Computation,Vol-23,pp221-230,April 1969

[30] Gautchi W., "On the Construction of Gausian Quadrature Rules from Modified Moments," Mathematical Computation,Vol-24,pp245-260, April 1970