NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

TRELLIS CODED CPFSK

by

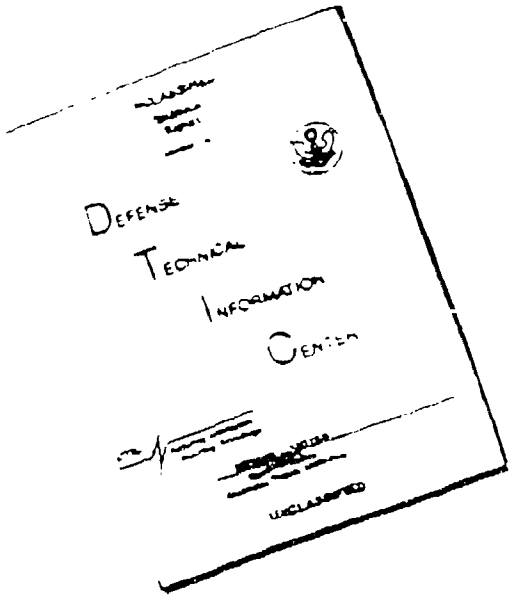
Zafer Inceoglu

December, 1992

Thesis Advisor : Paul Moose
Second Reader : Tri Ha

Approved for public release; distribution is unlimited.

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified		1b. RESTRICTIVE MARKINGS	
2a. SECURITY CLASSIFICATION AUTHORITY		3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited.	
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			
4. PERFORMING ORGANIZATION REPORT NUMBER(S)		5. MONITORING ORGANIZATION REPORT NUMBER(S)	
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School	6b. OFFICE SYMBOL (*if applicable*) 32	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School	
6c. ADDRESS (City, *State, and ZIP Code*) Monterey, CA 93943-5000		7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943-5000	
8a. NAME OF FUNDING / SPONSORING ORGANIZATION	8b. OFFICE SYMBOL (*if applicable*)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER	
8c. ADDRESS (City, *State, and ZIP Code*)		10.SOURCE OF FUNDING NUMBERS	
		PROGRAM ELEMENT NO	PROJECT NO
			TASK NO
			WORK UNIT ACCESSION NO
11. TITLE *(Include security Classification)* Trellis Coded CPFSK			
12. PERSONAL AUTHOR(S) Zafer Inceoglu			
13a. TYPE OF REPORT Master's Thesis	13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (*Year, Month, Day*) 1992 December 10	15. PAGE COUNT 54
16. SUPPLEMENTARY NOTATION The views expressed in this thesis are those of the author and do not reflect official policy or position of the Department of Defense or the U.S. Government			

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (*Continue on reverse if necessary and identify by block number*)
MSK; Trellis coded; CPFSK

19. ABSTRACT (*Continue on reverse if necessary and identify by block number*)

Continuous Phase Frequency Shift Keying (CPFSK) is a potentially attractive modulation scheme with constant envelop and good spectral characteristics, for energy constrained and band-limited satellite channels.

This research deals with Orthogonal Frequency Shift Keying (FSK), Minimum Shift Keying (MSK) which is a special case of CPFSK, uncoded quaternary CPFSK and finally coded quaternary CPFSK. Orthogonal FSK is simulated by making the modulation index (h) equal to one, and all the other simulations are performed with h=1/2.

A rate 1/2 convolutional encoder with constraint lengths (k), k=2, 3, and 4 are used in coded quaternary CPFSK simulations. Good coding gains are obtained with only a slight increase in receiver complexity. Soft decision with the Viterbi Algorithm was applied to all CPFSK and one MSK application and hard decision was applied to Orthogonal FSK and another MSK application.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT [X] UNCLASSIFIED/UNLIMITED [ ] SAME AS REPORT [ ] DTIC USERS	21. ABSTARCT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Paul Moose.	22b. TELEPHONE (*Include Area Code*) (408) 659-1938 / 22c. OFFICE SYMBOL EC / Me

DD Form 1473, JUN 86 *Previous editions are obsolete*
S/N 0102-LF-014-6603

Approved for public release; distribution is unlimited.

TRELLIS CODED CPFSK

by

Zafer Inceoglu
Lieutenant Junior Grade, Turkish Navy
B.S., Turkish Naval Academy, 1986

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1992

Author: _____
Zafer Inceoglu

Approved by: _____
Paul Moose, Thesis Advisor

_____
Tri Ha, Second Reader

_____
Michael A. Morgan, Chairman
Department of Electrical and Computer Engineering

ABSTRACT

Continuous Phase Frequency Shift Keying (CPFSK) is a potentially attractive modulation scheme with constant envelope and good spectral characteristics for energy constrained and band-limited satellite channels.

This research deals with Orthogonal Frequency Shift Keying (FSK), Minimum Shift Keying (MSK) which is a special case of CPFSK, uncoded quaternary CPFSK and finally coded quaternary CPFSK. Orthogonal FSK is simulated by making the modulation index (h) equal to one, and all the other simulations are performed with h=1/2.

A rate 1/2 convolutional encoder with constraint lengths (k), k=2, 3, and 4 are used in coded quaternary CPFSK simulations. Good coding gains are obtained with only a slight increase in receiver complexity.

Soft decision with the Viterbi Algorithm was applied to all CPFSK and one MSK application. Hard decision was applied to Orthogonal FSK and another MSK application.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

In appreciation for their time, effort, and patience, many thanks go to my instructors, advisors, and the staff and faculty of the Electrical and Computer Engineering Department, NPS. A special note of thanks goes to my advisor Prof. Paul Moose for his support and guidance.

I. INTRODUCTION

An increasing need for communication between individuals and nations is making satellite communication more and more actractive every day. Physical constraints of space based applications force us to choose bandwidth and energy efficient modulation techniques.

Continuous Phase Modulation (CPM) is a digital modulation scheme with continuous phase and constant envelope. Continuous phase makes it spectrally efficient, and constant envelope seems desirable where non-linearities are present in space segment power amplifiers and repeaters.

Coding is used to reduce reception errors. The trade-off in coding is a decrease in bandwidth efficiency. This thesis mainly presents the research for rate 1/2 convolutional quaternary CPM schemes with different constraint lengths.

The starting point for this research was the simulation of rate 2/3 trellis coded CPFSK. The complexity and apparent poor performance of this code forced us to first study orthogonal FSK, and then continue with MSK, uncoded quaternary CPFSK and finally finish with rate 1/2 trellis coded CPFSK. Due to the limitation of time, we were unable to return to the 2/3 rate codes.

The simulations were done using PRO-MATLAB software run on

Sun SPARC station computers. Hardware limitations and the complexity of the CPM schemes made this research difficult and very time consuming.

Chapter II gives a brief review of Continuous Phase Modulation. Chapter lII focuses on Orthogonal FSK, and two different receiver structures for MSK. Chapter IV presents uncoded quaternary CPFSK and coded quaternary CPFSK with different constraint lengths. Finally, Chapter V focuses on the final conclusions and opportunities for future research.

An identical string of fifty thousand information bits has been used for each simulation to provide a comparison between systems.

II. BACKGROUND

This chapter focuses on the basic characteristics of Continuous Phase Frequency Shift Keying. In CPFSK, frequency shifts are used to transmit information. The phase of the CPFSK signal is continuous during frequency shifts. This prevents the large spectral side lobe effects that are observed in FSK, and PSK. There is also memory in CPFSK due to the continuous phase. Phase trellises will be presented in this chapter later on.

A. CPFSK SIGNAL DESCRIPTION

Equation (2.1) represents the transmitted signal in CPFSK, [Ref. 1]

$$s(t)=\sqrt{2\frac{E_s}{T_s}}\cos[2\pi f_c t+\Theta(t;I)+\Theta_0] \qquad (2.1)$$

where T_s is the symbol duration, E_s the energy per symbol, $\Theta(t;I)$ is the time varying phase modulation induced by the input data sequence, f_c is the carrier frequency, and finally, Θ_0 is the initial phase of the carrier assumed to be zero for simplicity. Equation (2.2) represents the phase of the carrier in the interval $nT_s<t<(n+1)T_s$.

$$\Theta(t;I) = \pi h \sum_{k=-\infty}^{n-1} I_k + 2\pi h q(t-nT_s) \qquad (2.2)$$

The parameter h is called the modulation index. Typically h is chosen to be a rational number $h=\frac{m}{p}$ where m and p are relatively prime positive numbers, in order that $\Theta(t;I)$ takes on a finite number of states. The $q(t)$ is a phase shaping pulse signal, I is the input signal. For M-ary CPFSK, $M=2^n$ and I_n is one of the $\{\pm 1, \pm 3, \ldots\ldots\ldots, \pm(M-1)\}$ information symbols.

Phase shaping signals can be in a rectangular or raised cosine form. A rectangular form is used in this research. Phase trajectories for rectangular pulse shapes are piecewise linear. There are p phase states when m is even and $2p$ phase states when m is odd. Smoother phase trajectories can be obtained by using raised cosine pulse shapes.Phase trajectories for binary CPFSK are illustrated in Fig. 2.1.

B. ERROR PERFORMANCE

Minimum Shift Keying (MSK) was used as a baseline modulation scheme; a special case of binary CPFSK with $h=1/2$. In coded CPFSK cases we also made comparisons with MSK, because, although we send two bits at a time, one of them is an information bit, and the other is a redundant coding bit.

Equation (2.3) represents the probability of error in the

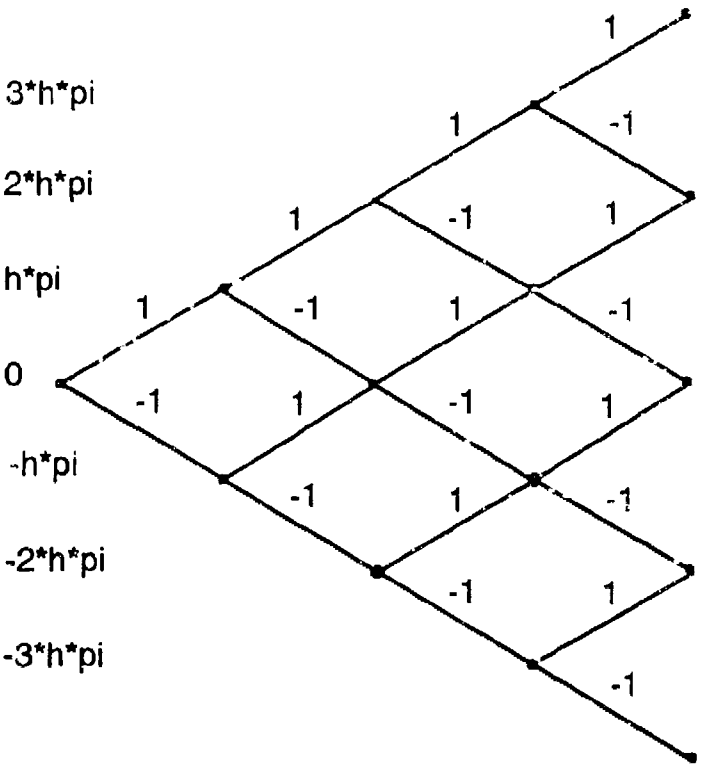


Fig. 2.1: Phase trellis for binary CPFSK.

presence of additive white Gaussian noise for MSK. [Ref. 2]

$$P_e = Q\left(\sqrt{2\frac{E_b}{N_0}}\right) \qquad (2.3)$$

The error performance can be improved by increasing the signal energy, but in satellite applications there are limitations on power sources.

Error performances for each modulation scheme are given at the end of each section.

C. BANDWIDTH

Bandwidth is the range of frequencies which contains 99 percent of the total signal power. A smaller value of h will increase the bandwidth efficiency with an increase in probability of error.

III. FSK-MSK

This chapter focuses on orthogonal FSK and MSK. The same modulator and bit mapper was used in all simulations. Input bits were mapped to modulation symbols according to Table 3.1.

TABLE 3.1: MAPPING FOR ORTHOGONAL FSK AND MSK.

input bit	mapper (I)
0	-1
1	1

A. ORTHOGONAL FSK

Orthogonal FSK simulation is accomplished by using binary CPFSK with h=1. Coherent detection is used in the demodulator and hard decisions are made to detect the received signal. By making h=1, we have two phase states; 0 and π. The phase trellis for this system is illustrated in Fig 3.1. The odd symbols represent the system at state 0, and even symbols represent the system at state π. Multiplying the received signal by $e^{j\pi t/T_s}$ offsets the total phase to 0 at the end of each symbol and decreases the receiver complexity. The maximum value of the correlator outputs for the two modulation symbols is selected as the decision variable.

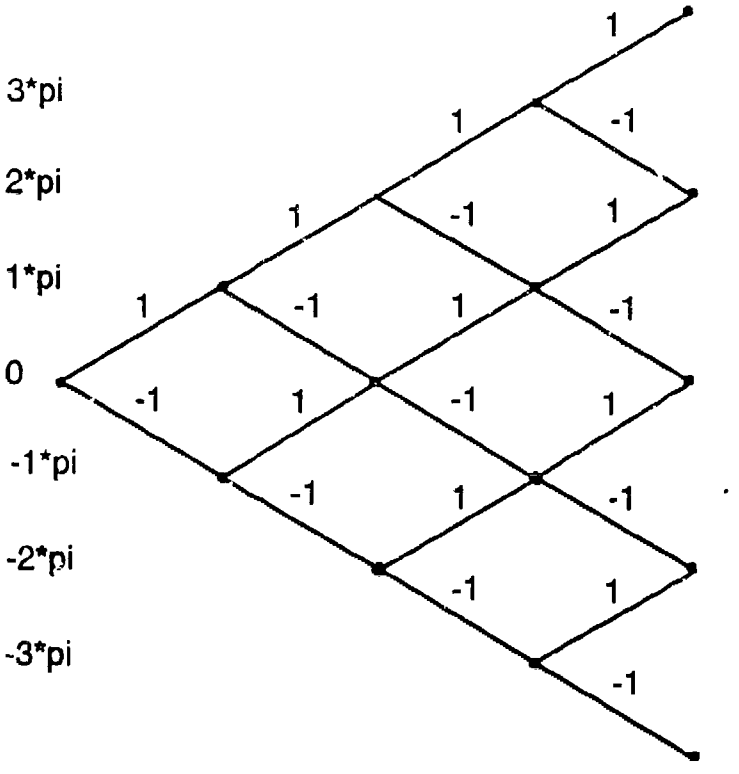


Fig. 3.1: Phase trellis for Orthogonal FSK

The MATLAB source code used for this system is listed in Appendix A. Equation (3.1) represents the probability of bit error for orthogonal FSK. [Ref. 2]

$$P_e = Q\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{3.1}$$

The performance of this system is illustrated in Fig. 3.2. The orthogonal FSK simulation was done for simulation verification, and as guidance to MSK simulation.

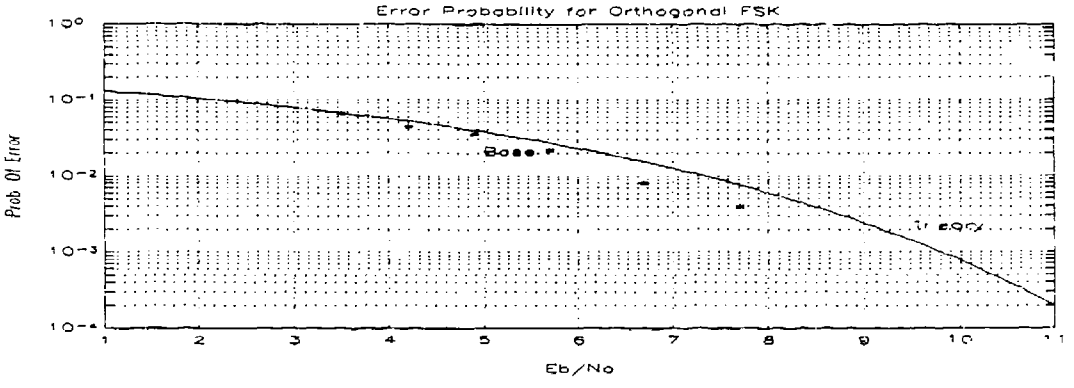


Fig. 3.2: Performance of Orthogonal FSK

B. MSK WITH VITERBI ALGORITHM DECODING

MSK is a special case of binary CPFSK with h=1/2 where the phase changes $\pm\pi/2$ at every symbol (see Fig. 3.3). The total phase is $\pm\pi/2$ for odd symbols and the total phase is 0 or π for even symbols. When the received signal is multiplied by $e^{j2\pi t/4T_s}$, it offsets the phase such that the total phase changes either by 0 or by π at every symbol. This reduces the number of states required in the Viterbi decoder. [Ref. 3]

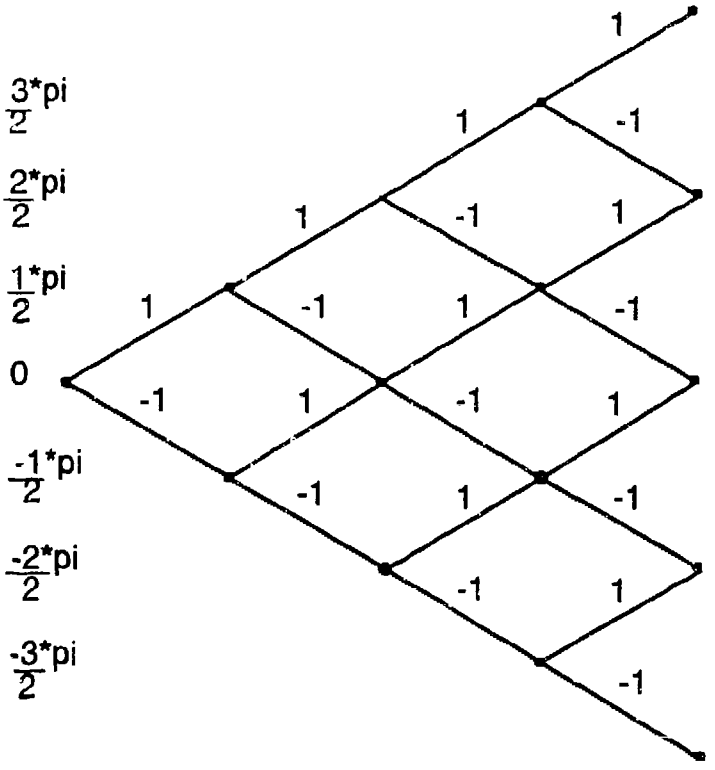


Fig. 3.3: Phase Trellis for MSK

The state trellis for the MSK waveform is illustrated in Fig. 3.4. There are two states, 0 and π. There are two paths out of each node corresponding to the two possible values of the input. Information bit 1 produces a change in state and information bit 0 produces no change. The Euclidean distances of the transmit symbol waveforms from the received signal are used as input to the Viterbi algorithm; the Viterbi algorithm chooses the path with the minimum total distance.

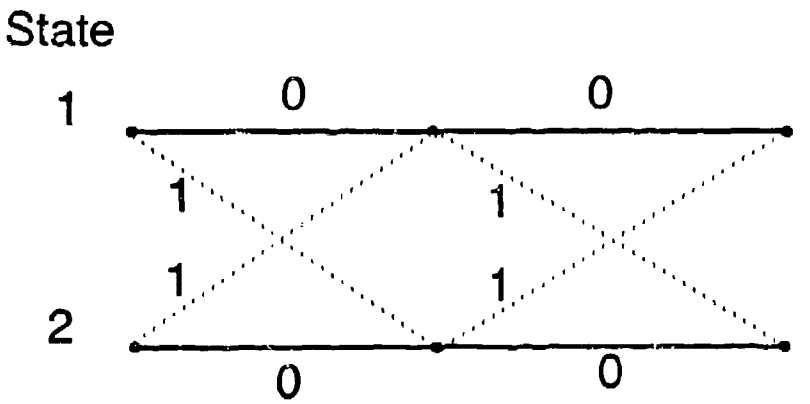


Fig. 3.4: State Trellis for MSK

The transition matrix of the MSK with Viterbi algorithm decoding is illustrated in Table 3.2.

TABLE 3.2: VITERBI TRANSITION MATRIX.

$$\begin{Vmatrix} 1 & 0 & 2 & 2 & 1 & 3 \\ 1 & 1 & 1 & 2 & 0 & 4 \end{Vmatrix}$$

Each row of the matrix corresponds to a state. The first column of the matrix represent the states that we are coming from, the second column represents the input value, and the third column represents the corresponding number of the point

in the signal constellation. Columns 4, 5, 6 are the same respectively. The MATLAB source code for this system is given in Appendix B.

Theory vs. simulation of the MSK bit error probability with Viterbi detection is illustrated in Fig. 3.5.

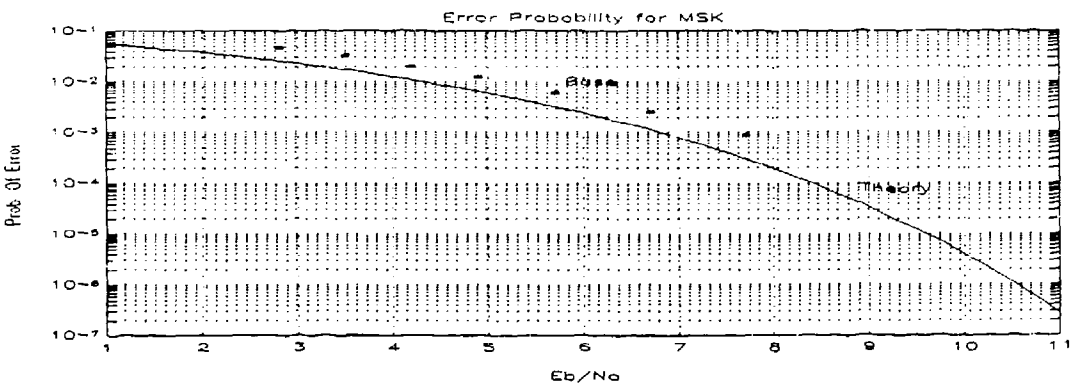


Fig. 3.5: Performance of MSK with Viterbi Detection.

C. MSK WITH DETECTION USING TWO SYMBOLS AT A TIME

Equation (2.1) can be rewritten as (3.2). [Ref. 4]

$$s(t) = \sqrt{2\frac{E_b}{T_b}}\cos[\Theta(t)]\cos(2\pi f_c t) - \sqrt{2\frac{E_b}{T_b}}\sin[\Theta(t)]\sin(2\pi f_c t) \tag{3.2}$$

Equation (3.3) shows that the phase of an MSK signal increases or decreases linearly within the bit duration. [Ref. 4]

$$\Theta(t) = \Theta(0) \mp \pi\frac{t}{2T_b} \tag{3.3}$$

The minus sign in (3.3) represents information signal 0, and the plus sign represents one. The signal phase can have values of $\pm\pi/2$ at odd multiple of T_b or 0 and π at even multiple of T_b (see (3.3) and Fig. 3.3).

When (3.3) is inserted in (3.2), it is clear that the input sequence during $(0, T_b)$ affects only the quadrature component of (3.2), and not the in-phase component over the signalling interval $(0, 2T_b)$. Similarly the input sequence during $(T_b, 2T_b)$ affects only the in-phase component of (3.2), and not the quadrature component over the signalling interval $(T, 3T_b)$.

Table 3.3 was constructed from the information discussed above. [Ref. 4]

TABLE 3.3: EFFECT OF INFORMATION BITS ON PHASE.

INFORMATION BITS	$\Theta(0)$	$\Theta(T_b)$
0	π	$\pi/2$
0	0	$-\pi/2$
1	0	$\pi/2$
1	π	$-\pi/2$

The hard decision in the demodulator is done according to Table 3.3, and the MATLAB source code for this system is illustrated in Appendix C.

Theory vs. simulation of MSK bit error probability for detection using two symbols at a time is illustrated in Fig. 3.6.

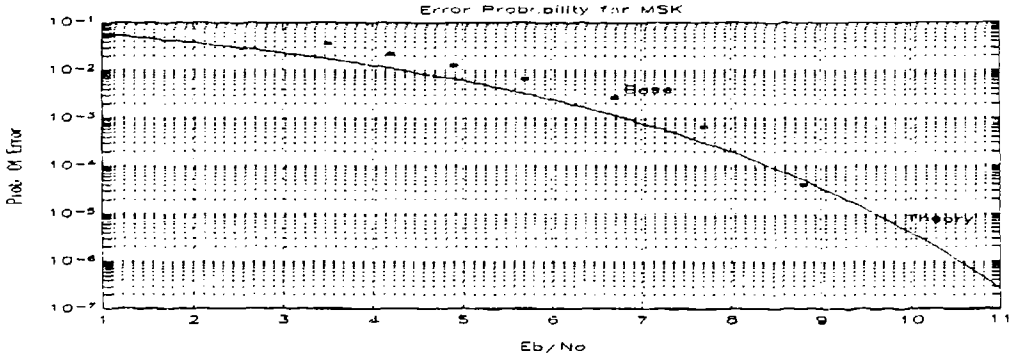


Fig. 3.6: Performance of MSK for detection using two symbols at a time.

IV. CODED CPFSK SCHEMES

Channel coding can be used to reduce the number of errors when a message is transmitted over a noisy channel. To achieve this goal, redundancy is added to the information sequence.

The encoder used in this research is a linear, sequential, finite state machine which is implemented by using shift registers. Rate 1/2 encoders with constraint lengths of 2, 3, and 4 are used. The transmitter block diagram for these systems is illustrated in Fig 4.1, and receiver block diagram is illustrated in Fig. 4.2.

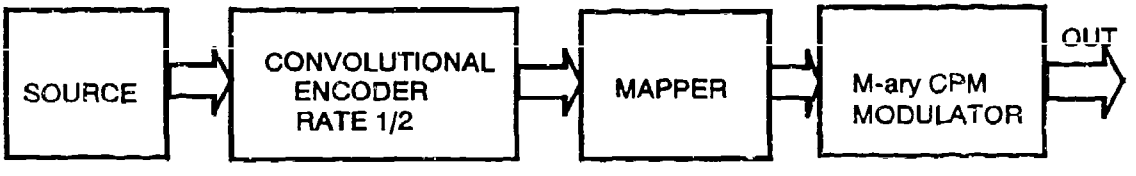


Fig. 4.1: Transmitter Block Diagram.

Fig. 4.2: Receiver block diagram.

The bit mapping used in this chapter is illustrated in Table 4.1. Two coded bits are sent in each symbol duration.

TABLE 4.1: MAPPING FOR CODED 4-CPFSK.

input dibits		mapper (I)
00	---------->	-3
01	---------->	-1
10	---------->	1
11	---------->	3

The phase trellis for the uncoded quaternary CPFSK with

h=1/2 is illustrated in Fig. 4.3. By combining coding techniques presented in this chapter, the probability of choosing the wrong information sequence will be reduced significantly.

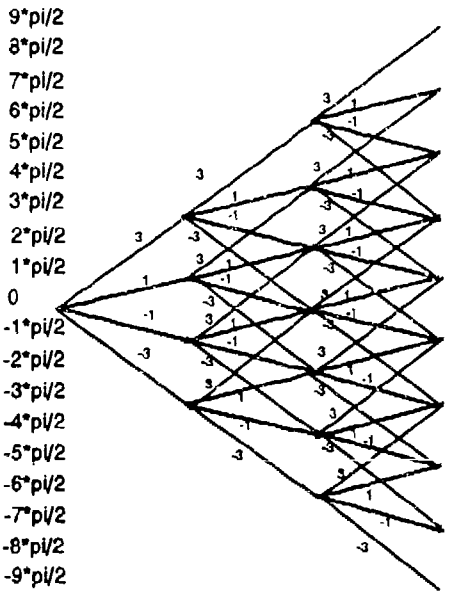


Fig. 4.3: Phase trellis for quaternary CPFSK.

The demodulator used in this chapter is illustrated in Fig 4.4. As is seen, the received signal is first offset and then passed through four correlators. Equation (4.1) represents the correlator outputs.

$$C_n = \frac{1}{T_s} \int_0^{T_s} r^*(t) \cdot s_n(t)\, dt \qquad \textbf{(4.1)}$$

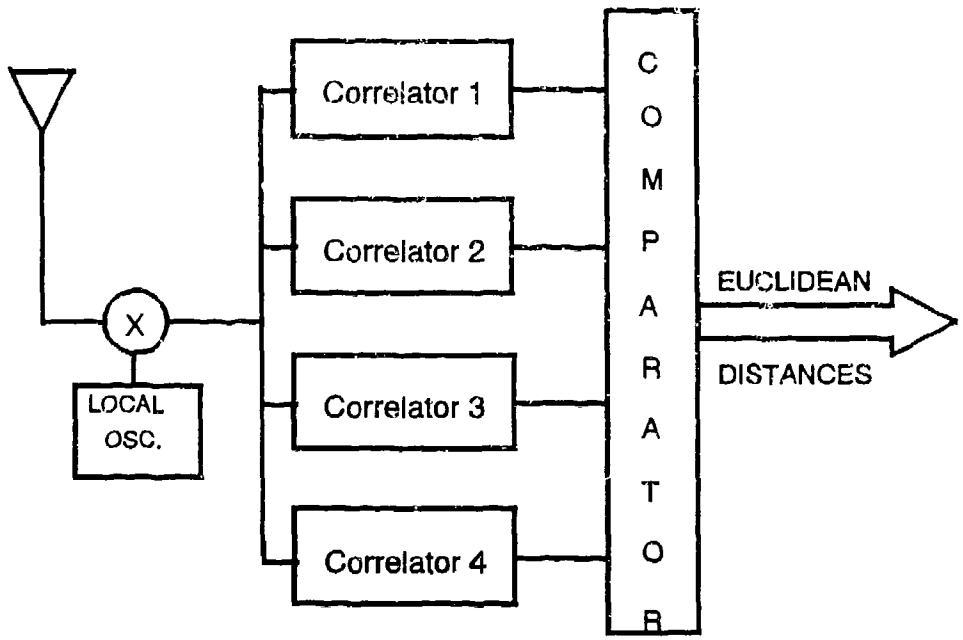


Fig. 4.4: Demodulator.

Real parts of the correlator outputs are taken, and are used as elements of a four dimensional received signal vector. The euclidean distances of this vector are calculated in the comparator from each of the vectors in the signal constellation. The vectors of the signal constellation are determined as follows.

In the absence of noise, the received signal will be $r(t) = e^{(j\Theta_k)}\, e^{(j\pi h I_k t/T_s)}$ where the phase state $\Theta_k = (0, \pi)$ and the transmitted information sequence $I_k = (-3, -1, 1, 3)$. Thus there are eight vectors in the signal constellation determined by;

$$C_{kn} = \frac{1}{T_s} Re\left[\int_0^{T_s} r^*(t)\, s_n(t)\, dt\right] \qquad (4.2a)$$

$$C_{kn} = \frac{1}{T_s} Re\left[e^{(j\theta_k)} \int_0^{T_s} e^{(j\pi h(I_n - I_k)\, t/T_s} dt\right] \qquad (4.2b)$$

$$C_{kn} = \mp(1)\, \frac{1}{T_s} Re\left[\int_0^{T_s} e^{(j\pi h(I_n - I_k)\, t/T_s} dt\right] \qquad (4.2c)$$

$$C_{kn} = \Psi_{kn} \qquad (4.2d)$$

Table 4.2 illustrates the Ψ_{kn} in a tabulated form. Each row of Table 4.2 represents a single point in the signal constellation.

TABLE 4.2: SIGNAL CONSTELLATION.

I_k / I_n	-3	-1	1	3
-3 (θ_k=0)	1	0	0	0
-3 (θ_k=π)	-1	0	0	0
-1 (θ_k=0)	0	1	0	0
-1 (θ_k=π)	0	-1	0	0
1 (θ_k=0)	0	0	1	0
1 (θ_k=π)	0	0	-1	0
3 (θ_k=0)	0	0	0	1
3 (θ_k=π)	0	0	0	-1

Equation (4.1) is represented as (4.3) in the MATLAB program.

$$C_n = \sum_{k=0}^{T_s/\Delta t} r^*(k\Delta t)\, s_n(k\Delta t) \qquad (4.3)$$

where Δt is the sampling period, and $T_s/\Delta t$ is the number of samples per symbol (N). In MATLAB simulation, Table 4.2 becomes Table 4.3.

TABLE 4.3: SIGNAL CONSTELLATION FOR MATLAB APPLICATION.

I_k / I_n	-3	-1	1	3
-3 (Θ_k=0)	N	0	1	0
-3 (Θ_k=π)	-N	0	-1	0
-1 (Θ_k=0)	0	N	0	1
-1 (Θ_k=π)	0	-N	0	-1
1 (Θ_k=0)	1	0	N	0
1 (Θ_k=π)	-1	0	-N	0
3 (Θ_k=0)	0	1	0	N
3 (Θ_k=π)	0	-1	0	-N

A. CODED 4-CPFSK WITH CONSTRAINT LENGTH TWO

The coding used in this section is created by using two shift-registers. [Ref. 5] The encoder is illustrated in Fig. 4.5, and the state trellis for this system is illustrated in Fig 4.6 (b).

The demodulator calculates the euclidean distances of the received signal vector from the eight vectors illustrated in Table 4.4. At each symbol duration, eight distance values are entered to the Viterbi algorithm and the true path is chosen according to minimum distance.

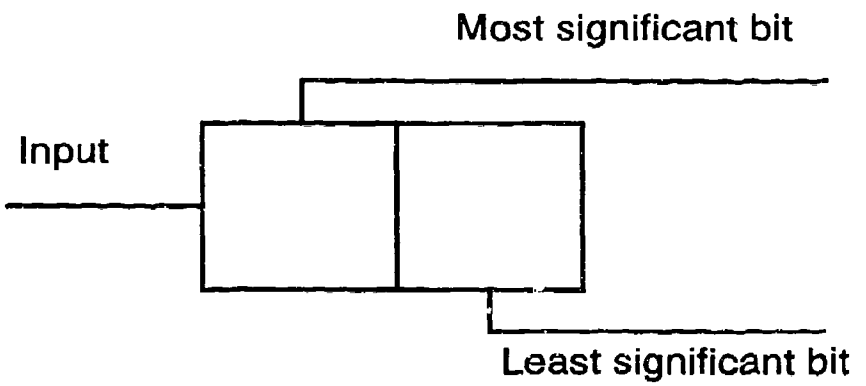


Fig. 4.5: Encoder.

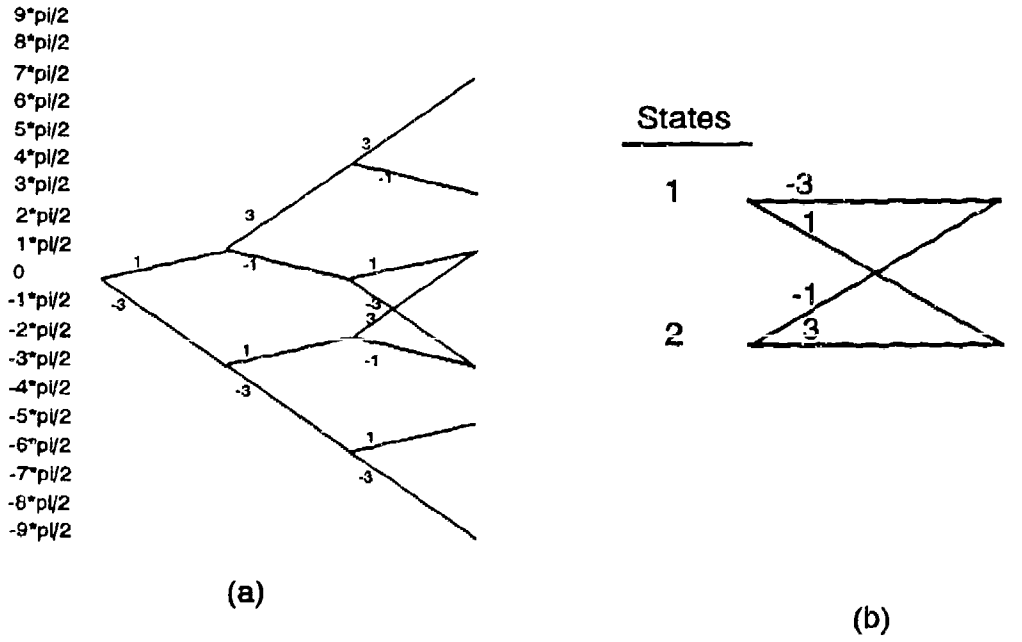


Fig. 4.6: Phase Trellis (a), and State Trellis for coded 4-CPFSK with k=2 (b).

When comparing Fig. 4.3 and Fig 4.6 (a), it is clear that at each symbol duration, the probability of choosing the wrong state is reduced by this simple coding. Later, in the following sections, more complicated codes will be studied and the coding gains that are obtained will be seen more clearly.

TABLE 4.4: SIGNAL CONSTELLATION.

$$\begin{Vmatrix} 0,-1,0,-N & 0,1,0,N & 1,0,N,0 & -1,0,-N,0 \\ 0,-N,0,-1 & 0,N,0,1 & N,0,1,0 & -N,0,-1,0 \end{Vmatrix}$$

The transition matrix for this system is illustrated in Table 4.5.

TABLE 4.5: VITERBI TRANSITION MATRIX.

$$\begin{Vmatrix} 1 & 0 & 1 & 1 & 0 & 2 & 2 & 1 & 3 & 2 & 1 & 4 \\ 1 & 2 & 5 & 1 & 2 & 6 & 2 & 3 & 7 & 2 & 3 & 8 \end{Vmatrix}$$

Each row of the matrix corresponds to a state. The first column of the matrix represent the states that we are coming from, the second column represents the input value, and the third column represents the corresponding point in the signal constellation that is illustrated in Table 4.4; the points are numbered rowwise. All other columns are the same respectively.

The performance of this system is illustrated in Fig. 4.7. The coding gain obtained will not be worth what must be sacrificed in bandwidth efficiency (see Fig. 4.7). Since this

was the initial experiment in the use of coding, it provided a baseline for cur study. The MATLAB source code for this system is listed in Appendix D.

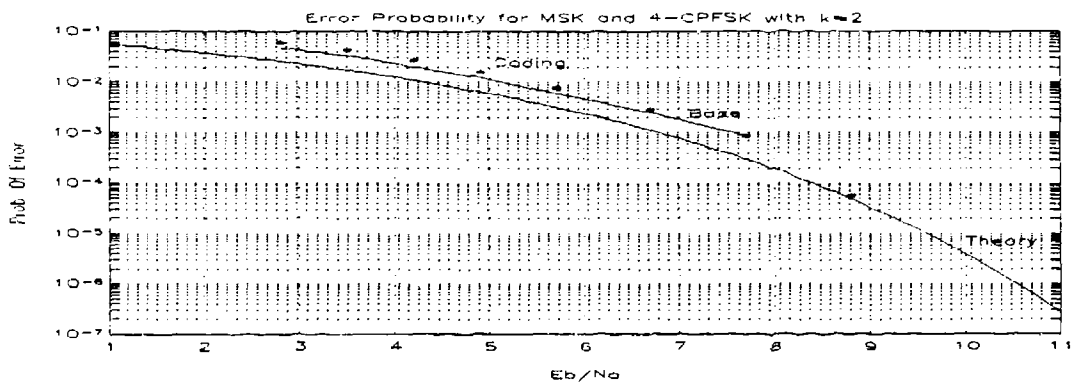


Fig 4.7: Performance of the coded 4-CPFSK with k=2.

B. CODED 4-CPFSK WITH CONSTRAINT LENGTH THREE.

The coding used in this section was created by using three shift-registers. The encoder is illustrated in Fig. 4.8, and the state trellis for this system is illustrated in Fig 4.9. [Ref. 5]

The demodulator calculates the euclidean distances of the received signal vector from the eight vectors presented in

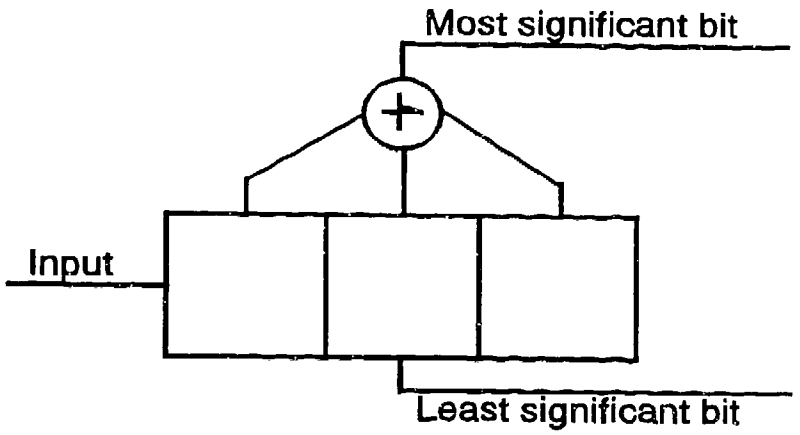


Fig 4.8: Encoder.

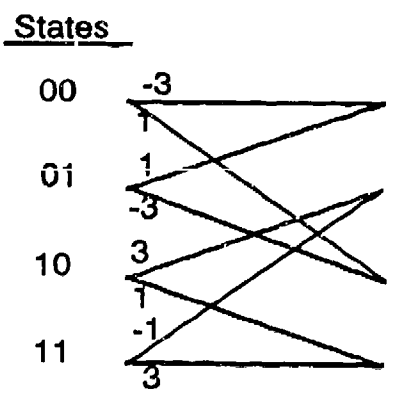


Fig 4.9: State Trellis for coded 4-CPFSK with k=3.

Table 4.4. At each symbol duration, eight distance values are entered to the Viterbi decoder and the true path is chosen according to minimum distance.

The transition matrix for this system is given in Table 4.6.

TABLE 4.6: VITERBI TRANSITION MATRIX.

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

The performance of this system is illustrated in Fig. 4.10.

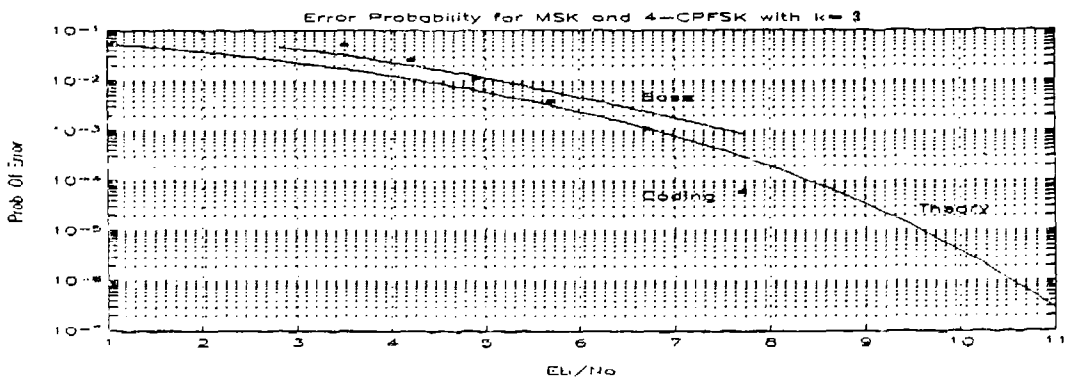


Fig. 4.10: Performance of coded 4-CPFSK with k=3.

The coding starts working efficiently after 6.7 dB E_b/N_0. At this signal-to-noise ratio almost 1 dB gain is obtained and at 7.6 dB E_b/N_0, the coding gain reached almost 2 dB.

Signal-to-noise ratios for 8.8 dB and 10 dB were also tested and no errors were detected in fifty thousand information bits.

The MATLAB source code for this system is listed in Appendix E.

C. CODED 4-CPFSK WITH CONSTRAINT LENGTH FOUR.

The final coding studied, uses four shift registers. The encoder is illustrated in Fig. 4.11, and the state trellis for this system is illustrated in Fig. 4.12. [Ref. 5]

In this application, the demodulator calculates the euclidean distances of the received signal vector from eight different vectors presented in Table 4.4. At each symbol duration eight distances are entered to the Viterbi decoder, and the true path is chosen according to minimum distance.

The transition matrix of this system is illustrated in Table 4.7 and the MATLAB source code is listed in Appendix F.

The performance of this coding is illustrated in Fig. 4.13. The coding starts working efficiently at 5.9 dB E_b/N_0. At this signal-to-noise ratio almost 1 dB gain is obtained and at 6.8 dB E_b/N_0, the coding gain reached almost 2 dB.

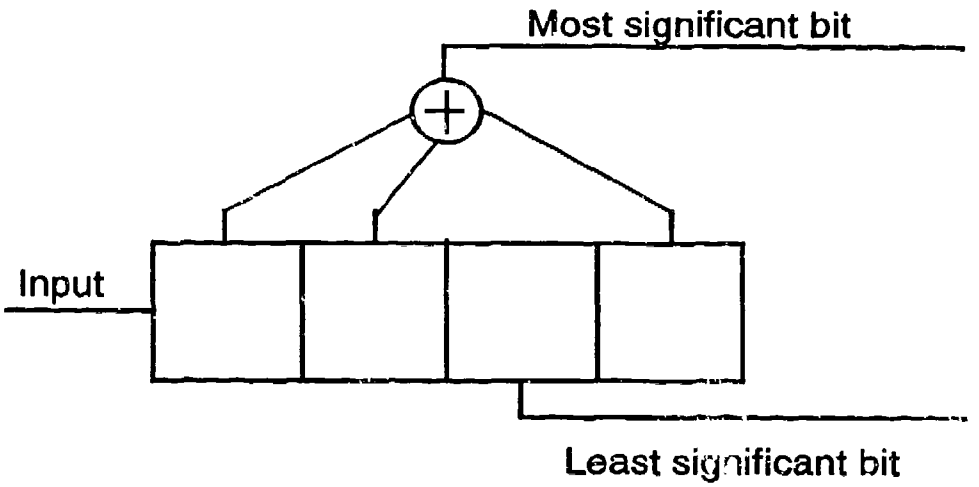


Fig. 4.11: Encoder.

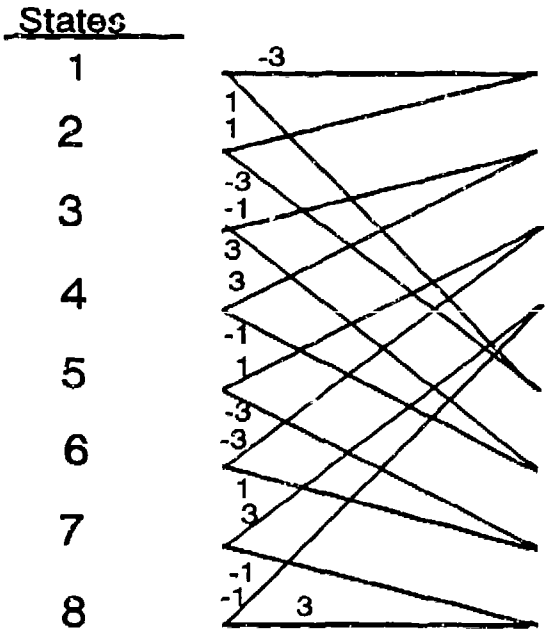


Fig. 4.12: Trellis for coded 4-CPFSK with k=4.

TABLE 4.7: VITERBI TRANSITION MATRIX.

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

Signal-to-noise ratios of 7.7 dB, 8.8 dB, and 10 dB are also tested and no errors were detected for fifty thousand information bits.

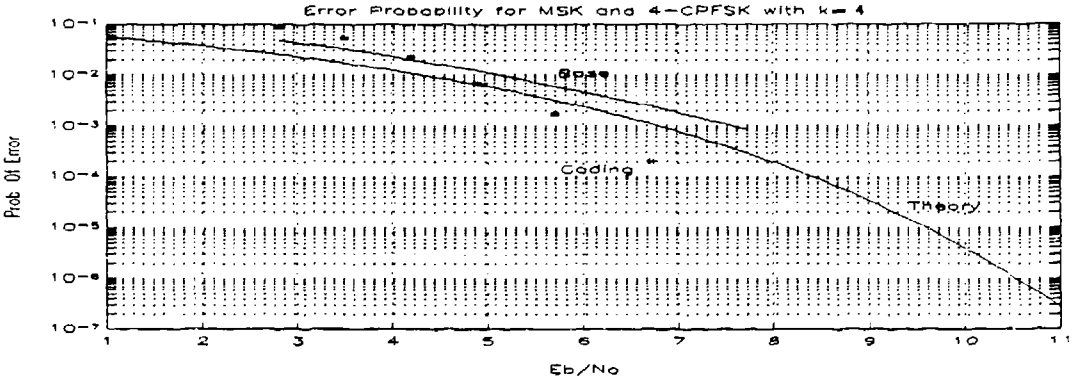


Fig. 4.13: Performance of the coded 4-CPFSK with k=4.

V. CONCLUSIONS AND FURTHER RESEARCH

This thesis mainly presented the simulation results for rate 1/2 convolutional quaternary CPM schemes with different constraint lengths. Using different codes, the minimum merge length increased, and larger minimum Euclidean distances were obtained. Better error performances were obtained as a result of these codes.

This research can be extended in several ways. First, rate 1/2 codes can be improved by using different constraint lengths, and shift register connections. The same encoders can also be tested with different modulation indexes.

More complicated rate 1/2 codes with h=1/2, can be simulated very easily by making small changes to the MATLAB source codes that are listed in Appendices D thru F. The signal constellation that is presented in Table 4.4 can be used for every new code.

Second, rate 2/3 codes for different constraint lengths, shift register connections, and modulation indexes can be studied, and best systems with minimum receiver complexity, and maximum power-bandwidth performances can be found.

APPENDIX A

```
%This program creates a m by n binary input matrix.

function [random_matrix]=random(m,n);
a=rand(m,n);
b=ones(m,n)*0.5;
random_matrix=floor(a+b);


%   This program is used as mapper. It converts 0 to -1 and 1
% to 0.

function [mapper_output]=mapper(input);
[m,a]=size(input);
for x=1:a
  y=ones(m,1)*2^(2-x);
  z=[z,y];
end;
mapper_output=input.*z;
mapper_output=mapper_output';
mapper_output=(mapper_output)-1;


%   This program is used as CPFSK modulator. Inputs to this
% program are "T=symbol duration", "h=modulation index", and
% "modulator_input=mapper output".

function [mod_output]=modulator(T,h,modulator_input);
z=length(modulator_input);
teta0=0;
for sample=1:z
    I=modulator_input(sample);
    if sample==1
      teta=0; %Sets the initial phase to zero.
    else
      teta0=teta0+modulator_input(sample-1);
      teta=pi*h*teta0;
    end
    time=0:0.0005:T-0.0005;
    time=time+(sample-1)*T;
    xx=pi*I*h*(time/T-(sample-1))+teta;
    mod_output(sample,:)=(exp(i*xx));
end
```

```
%   This program is used to add Gaussian noise to modulator
% output. Inputs to this program are "m as modulator output",
% "coeff as standard deviation of the noise", and "T as symbol
% duration". The constant 0.005 as sampling period.

function [awgn_noise]=awgn_ch(m,coeff,T)
rand('seed',99);
rand('normal');
awgn_noise=coeff*(rand(m,T/0.005)+j*rand(m,T/0.005));



%   This program is used as CPFSK demodulator. Inputs to this
% program are; "T as symbol duration", "h as modulation index"
% "mod_output as received signal". In this demodulator hard
% decision is used.

function [demod_output]=demodulator(T,h,mod_output);
map=[-1 1];
[m,a]=size(mod_output);
time=0:0.0005:T-0.0005;
for sample=1:m
  x=rem(sample,2);
  phase_coeff=[(1-x)*pi];
  phase=exp(i*phase_coeff);
  for m_ary=1:2
      xx=exp(i*time*pi*h*map(m_ary)/T);
      match=xx*conj(mod_output(sample,:)');
      match=phase*match;
      match_out=[match_out match];
  end
[y,i]=max(real(match_out));
demod_output(sample)=map(ceil(i/1));
match_out=[ ];
end
```

```
function [Nb,I,e] =check(x,y)
%                       SYMBOL ERROR CHECK
%                         Paul H. Moose
%                    Naval Postgraduate School
%                         09-01-91
%
% This m-file locates the positions in vectors x and y that do
not
% agree. It returns a one in e if they do not agree and a zero
if they
% do agree. e is the error vector if x and y are binary. I is
a vector
% of error location numbers. Nb is the sum of the elements of
e.
e=(x~=y);
I=find(e);
Nb=sum(e);



%   This program is used as master program. It calls different
% functions and simulates whole communication system.

clear
T=0.015; % "T" is used as symbol duration.
h=1;     % "h" is used as modulation index.
m=50000;
n=1;     % "m" and "n" are used to create binary matrix.
coeff=0; % "coeff" is used as standard deviation of the noise.
[input_matrix]=random(m,n);
[mapper_output]=mapper(input_matrix);
[mod_output]=modulato(T,h,mapper_output);
[awgn_noise]=awgn_ch(m,coeff,T);
mod_output=mod_output+awgn_noise;
[demodulator_output]=demodula(T,h,mod_output);
error=check(mapper_output,demodulator_output);
```

APPENDIX B

```
%   This program is used as CPFSK demodulator. Inputs to this
% program are; "T as symbol duration", "h as modulation index"
% "mod_output as received signal". In this demodulator hard
% decision is used.

function [demod_output]=demodulator(T,h,mod_output);
map=[-1 1];
[m,a]=size(mod_output);
time=0:0.0005:T;
for sample=1:m
  mod=mod_output(sample,:)*exp(j*(sample-1)*pi/2); %Offsets
                                                   %the phase
  for m_ary=1:2
      xx=exp(i*time*pi*h*map(m_ary)/T);
      match=xx*conj(mod_output(sample,:)');
      match_out=[match_out match];
  end
demod_output(sample,:)=(real(match_out));
match_out=[ ];
end
R=(j*demod_output(:,1)+demod_output(:,2))/31;
[demod_output]=eucdis(2,R); %Calculates the euclidean distance
                            %of the received signal



function D = eucdis(q,R)
%                       EUCLIDEAN DISTANCE METRICS
%                            Paul H. Moose
%                       Naval Postgraduate School
%                              06-17-92
%
% This M-file finds Euclidean distance of elements
% in vector R from 2^q unit amplitude vectors
% equally spaced on the unit circle. It stores these as rows
of D.
%
N=2^q;
L=length(R);
index=1:N;
dph=2*pi/N;
M0=exp(j*dph.*(index-1));
for l=1:L
D(l,:)=abs(R(l).*ones(M0)-M0);
end
```

```
function PHN = softv(k,K,Np,PH,T,D)
%                Soft Viterbi Decoder
%           Paul H. Moose
%       Univ. degli Studi di Padova
%            17-05-91
%
%   This M-file decodes k bit msgwords from 2^n real metrics
% (These may, for exam., represent the "distance" of the
% received modulation value from each of 2^n modulation
values.)
% The state transition information for a 2^K state trellis is
in
% the 2^K by 3*2^k matrix T. Each of the 2^k entering paths
to
% each state has its source state (one of 2^K), path msgword
(one
% of 2^k) and path codeword (one of 2^n) listed in the state
row.
%    The path histories are kept in matrix PH that is 2^K by
3*Np.
% The path history for each state contains source state, path
% weight and path codeword  for Np previous states.
%   The output PHN is the update of PH, the new path history.
% The decoded code word is in the last column of PHN.(They
should
% "merge".
%     The past histories are undated on the basis of the
"minimum
% metric". You can change this to the "maximum metric" if
desired as
% indicated in the comments in the code.
%
for j=1:2^K
    X(j,2)=D(T(j,3))+PH(T(j,1),2); %path weight
    X(j,1)=T(j,1);%path source state
        X(j,3)=T(j,2);%path code word T(j,3).Chg to T(j,2) for
msgword.
        for l=2:2^k
        wt = D(T(j,3*l)) +PH(T(j,3*l-2),2);
        if wt < X(j,2) %The < selects min metric
        X(j,2)=wt;
        X(j,1)=T(j,3*l-2);
                X(j,3)=T(j,3*l-1);  %Chg  to  T(j,3*l)  for
codeword.
        else
        end
        end
% We now need to append old paths to new paths to get
survivors.
PHN(j,:)= [X(j,:) PH(X(j,1),1:3*Np-3)];
end
```

```
%   This program is used as master program. It calls different
% functions and simulates whole communication system. All
% other functions are same as in Appendix A.

clear
temp=[1 0 1 2 1 4;1 2 0 3]; %Is used as Viterbi path
                                                    %matrix.
T=0.015;
h=1/2;
m=50000;
n=1;
coeff=0;
[input_matrix]=random(m,n);
[mapper_output]=mapper(input_matrix);
[mod_output]=modulato(T,h,mapper_output);
[awgn_noise]=awgn_ch(m,coeff,T);
mod_output=mod_output+awgn_noise;
[demodulator_output]=demodula(T,h,mod_output);
viterbi_path_matrix=temp;
TT=zeros(2,60);
for x=1:m
  D=demodulator_output(x,:);
  [TT]=softv(1,1,20,TT,viterbi_path_matrix,D);
  received_signal(x)=TT(1,60);
end
input_matrix=input_matrix(19:m);
received_signal=received_signal(1:m-19);
error=check(received_signal,input_matrix);
```

APPENDIX C

```
%   This program is used as CPFSK demodulator. Inputs to this
% program are; "T as symbol duration", "h as modulation index"
% "in as received signal". In this demodulator hard decision
% is used.

function [demodulator_output]=demodula(T,h,in)
N=T/.0005;
[k,a]=size(in);
p =1:a;
y3=-1;
for m=3:k
  phi1=exp(j*pi*(m-1)/2)*exp(j*(pi/2).*(p/N));
  phi2=exp(j*pi*(m-2)/2)*exp(j*(pi/2).*(p/N));
  x=rem(m,2);
  if (x==0)
    y=sum(real(in(m-2,:)).*abs(imag(phi2)))    +
sum(real(in(m-1,:)).*abs(imag(phi1)));
   if ((y<0 & y3>0) | (y>0 & y3<0))
       d=1;
   else
       d=-1;
   end
  else
    y=sum(imag(in(m-2,:)).*abs(real(phi2)))    +
sum(imag(in(m-1,:)).*abs(real(phi1)));
   if ((y>0 & y3<0) | (y<0 & y3>0))
       d=-1;
   else
       d=1;
   end
  end
  y3=y;
  demodulator_output(m)=d;
end
```

```
%   This program is used as master program. It calls different
% functions and simulates whole communication system. All
% other functions are same as in Appendix A.

clear
T=0.015;
h=1/2;
m=50000;
n=1;
coeff=0;
[input_matrix]=random(m,n);
[mapper_output]=mapper(input_matrix);
[mod_output]=modulato(T,h,mapper_output);
[awgn_noise]=awgn_ch(m,coeff,T);
mod_output=mod_output+awgn_noise;
[demodulator_output]=demodula(T,h,mod_output);
error=check(mapper_output(2:m-2),demodulator_output(4:m));
```

APPENDIX D

```
%   This program simulates the encoder that is given in
% Fig. 4.4. The input to this program is a m by 1 binary
% input matrix. At the end it gives a coded m by 2 binary
% output . First, two shift registers are initialized and then
% input are shifted.

function [tcm_output]=tcm(input_matrix);
[m,a]=size(input_matrix);
ff1=[0;input_matrix(:,1)]; % Intializes the first flip-flop;
ff1=ff1(1:m,1);
ff2=[0;ff1];               % Intializes the second flip-flop;
ff2=ff2(1:m,1);
tcm_output(:,1)=ff1;        % Most significent bit.
tcm_output(:,2)=ff2;        % Least significicent bit.


%   This program is used as mapper. It simulates the
% Table 4.1.

function [mapper_output]=mapper(input);
[m,a]=size(input);
for x=1:a
  y=ones(m,1)*2^(3-x);
  z=[z,y];
end
mapper_output=input.*z;
mapper_output=mapper_output';
mapper_output=sum(mapper_output)-3;
```

```
%   This program is used as CPFSK demodulator. Inputs to this
% program are; "T as symbol duration", "h as modulation index"
% "mod_output as received signal".

function [demod_output]=demodulator(T,h,mod_output);
map1=[-3 -1 3];
[m,a]=size(mod_output);
time=0:0.0005:T;
for sample=1:m
  mod=exp(j*(sample-1)*pi/2)*mod_output(sample,:);
  for m_ary=1:4
      xx=exp(i*time*pi*h*map1(m_ary)/T);
      match1=xx*conj(mod');
      match_out=[match_out match1'];
  end
demod_output(sample,:)=distance(real(match_out));
match_out=[];
end



%   This program calculates the euclidean distances  of the
% received signal from signal constellation.

function [dist]=distance(R);
matrix=[0 -1 0 -31        % This matrix represents the signal
        0 1 0 31                                %constellation.
        1 0 31 0
        -1 0 -31 0
        0 -31 0 -1
        0 31 0 1
        31 0 1 0
        -31 0 -1 0];
d=ones(8,1);
dd=d*R;
dd=(dd-matrix).^2;
dist=sum(dd');
```

```
%   This program is used as master program. It calls different
% functions and simulates whole communication system. All
% other functions are same as in Appendix A.

clear
temp=[1 0 1 1 0 2 2 1 3 2 1 4
      1 2 5 1 2 6 2 3 7 2 3 8];
T=0.015;
h=1/2;
m=50000;
n=1;
coeff=0;
[input_matrix]=random(m,n);
[tcm_output]=tcm(input_matrix);
[mapper_output]=mapper(tcm_output);
[mod_output]=modulato(T,h,map);
[awgn_noise]=awgn_ch(m,coeff,T);
mod_output=mod_output+awgn_noise;
[demodulator_output]=demodula(T,h,mod_output);
viterbi_path_matrix=temp;
TT=zeros(2,120);
for x=1:m
  D=demodulator_output(x,:);
  [TT]=softv(2,1,40,TT,viterbi_path_matrix,D);
  received_signal(x)=TT(1,120);
end
ww=mb(2,received_signal);
ww=ww(1:2:2*m);
error=check(input_matrix(1:m-41)',ww(42:m));
```

APPENDIX E

```
%   This program simulates the encoder that is given in
% Fig. 4.. The input to this program is a m by 1 binary
% input matrix. At the end it gives a coded m by 2 binary
% output . First, the three shift registers are initialized
% and then input is shifted.

function [tcm_output]=tcm(input_matrix);
[m,a]=size(input_matrix);
ff1=[0;input_matrix(:,1)]; % Initialize the first flip-flop.
ff1=ff1(1:m,1);
ff2=[0;ff1];               % Initialize the second flip-flop.
ff2=ff2(1:m,1);
ff3=[0;ff2];               % Initialize the third flip-flop.
ff3=ff3(1:m;1);
tcm_output(:,1)=abs(abs(ff1-ff2)-ff3); % Most significant bit.
tcm_output(:,2)=ff2;                   % Least significant bit.
```

```
function [PHN]= viterbi(k,Np,PH,T,D)
%       Soft Viterbi Decoder
%          Paul H. Moose
%       Mercury Digital Communications
%            06-09-91
%
%   This M-file decodes k bit msgwords from 2^n real metrics
% (These may, for example, represent the "distance" of the
% received modulation value from each of 2^n modulation
values.)
% The state transition information for a 2^K state trellis is
in
% the 2^K by 3*2^k matrix T. Each of the 2^k entering paths
to
% each state has its source state (one of 2^K), path msgword
(one
% of 2^k) and path codeword (one of 2^n) listed in the state
row.
%   The path histories are kept in matrix PH that is 2^K by
3*Np.
% The path history for each state contains source state, path
% weight and path codeword for Np previous states.
%   The output PHN is the update of PH, the new path history.
% The decoded msg word is in the last column of PHN.(They
should
% "merge".
%     The past histories are undated on the basis of the
"minimum
% metric". You can change this to the "maximum metric" if
desired as
% indicated in the comments in the code.
%
P=PH(:,2)';
wt=T(:,3:3:3*2^k)';
ux=T(:,1:3:3*2^k-2)';
aa=D(wt(:));
bb=P(ux(:));
wt(:)=aa+bb; %This contains all weights(columns) for each
state(row)
[a,b]=min(wt); % Use max(wt) here for maximum
X(:,2)=a';
X(:,1)=diag(T(:,3.*b-2));
X(:,3)=diag(T(:,3.*b-1));%path msgword. Chg to 3.*b for
codeword
%We now need to append old paths to new paths to get
survivors.
PHN=[X PH(X(:,1),1:3*Np-3)];
```

```
%   This program is used as master program. It calls different
% functions and simulates whole communication system. All
% other functions are same as in Appendix A.

clear
temp=[1 0 1 1 0 2 2 2 5 2 2 6
      3 3 7 3 3 8 4 1 3 4 1 4
      2 0 2 2 0 1 1 2 6 1 2 5
      3 1 4 3 1 3 4 3 8 4 3 7];
T=0.015;
h=1/2;
m=50000;
n=1;
coeff=0;
[input_matrix]=random(m,n);
[tcm_output]=tcm(input_matrix);
[mapper_output]=mapper(tcm_output);
[mod_output]=modulato(T,h,map);
[awgn_noise]=awgn_ch(m,coeff,T);
mod_output=mod_output+awgn_noise;
[demodulator_output]=demodula(T,h,mod_output);
viterbi_path_matrix=temp;
TT=zeros(4,120);
for x=1:m
  D=demodulator_output(x,:);
  [TT]=viterbi(2,40,TT,viterbi_path_matrix,D);
  received_signal(x)=TT(1,120);
end
ww=mb(2,received_signal);
ww=ww(1:2:2*m);
error=check(input(1:m-41)',ww(42:m));
```

APPENDIX F

```
%   This program simulates the encoder that is given in
% Fig. 4. The input to this program is a m by 1 binary
% input matrix. At the end it gives a coded m by 2 binary
% output . First, the four shift registers are initialized
% and then input is shifted.

function [tcm_output]=tcm(input_matrix);
[m,a]=size(input_matrix);
ff1=[0;input_matrix(:,1)];
ff1=ff1(1:m,1);
ff2=[0;ff1];
ff2=ff2(1:m,1);
ff3=[0;ff2];
ff3=ff3(1:m,1);
ff4=[0;ff3];
ff4=ff4(1:m,1);
tcm_output(:,1)=abs(ff1-abs(ff2-ff4));
tcm_output(:,2)=ff3;
```

```
%   This program is used as master program. It calls different
% functions and simulates whole communication system. All
% other functions are same as in Appendix A.

clear
temp=[ 1 0 1 1 0 2 2 2 6 2 2 5
       4 3 7 4 3 8 3 1 3 3 1 4
       5 2 5 5 2 6 6 0 2 6 0 1
       7 3 7 7 3 8 8 1 4 8 1 5
       1 2 5 1 2 6 2 0 2 2 0 1
       3 3 8 3 3 7 4 1 3 4 1 4
       5 0 2 5 0 1 6 2 5 6 2 6
       7 1 4 7 1 3 8 3 7 8 3 8];
T=0.015;
h=1/2;
m=50000;
n=1;
coeff=0;
[input_matrix]=random(m,n);
[tcm_output]=tcm(input_matrix);
[mapper_output]=mapper(tcm_output);
[mod_output]=modulato(T,h,map);
[awgn_noise]=awgn_ch(m,coeff,T);
mod_output=mod_output+awgn_noise;
[demodulator_output]=demodula(T,h,mod_output);
viterbi_path_matrix=temp;
TT=zeros(8,120);
for x=1:m
  D=demodulator_output(x,:);
  [TT]=viterbi(2,40,TT,viterbi_path_matrix,D);
  received_signal(x)=TT(1,120);
end
ww=mb(2,received_signal);
ww=ww(1:2:m*2);
error=check(input_matrix(1:m-42)',ww(43:m));
```

LIST OF REFERENCES

1. Proakis, John G., *Digital Communication*, pp. 172-190, pp. 627-642, McGraw-Hill, New York, 1989.

2. Couch, Leon W., *Digital and Analog Communication Systems*, pp. 535-550, Macmillan, New York, 1987.

3. Blahut, Richard E., *Digital Transmission of Information*, pp. 256-263, Addison-Wesley New York, 1990.

4. Haykin, Simon, *Communication Systems*, pp. 561-572, John Wiley & Sons, 1983.

5. Pizzi, Steven V., and Wilson, Stephen G., "*Convolutional Coding Combined with Continuous Phase Modulation*," *IEEE Trans. on Communications*, Vol Com-33, No1, pp. 20-29.

INITIAL DISTRIBUTION LIST

		No. of Copies
1.	Defense Technical Information Center Cameron station Alexandria, VA 22304-6145	2
2.	Library, Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Chairman, Code EC Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Professor Paul Moose, Code EC/Me Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
5.	Professor Tri Ha, Code EC/Ha Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, CA 93943-5000	1
6.	Deniz Kuvvetleri Komutanliği Personel Eğitim Daire Başkanliği Ankara, TURKEY	1
7.	Deniz Harp Okulu Komutanliği Tuzla İstanbul, TURKEY	1
8.	Gölcük Tersanesi Komutanliği Gölcük Kocaeli, TURKEY	1
9.	Taşkizak Tersanesi Komutanliği Kasimpaşa İstanbul, TURKEY	1