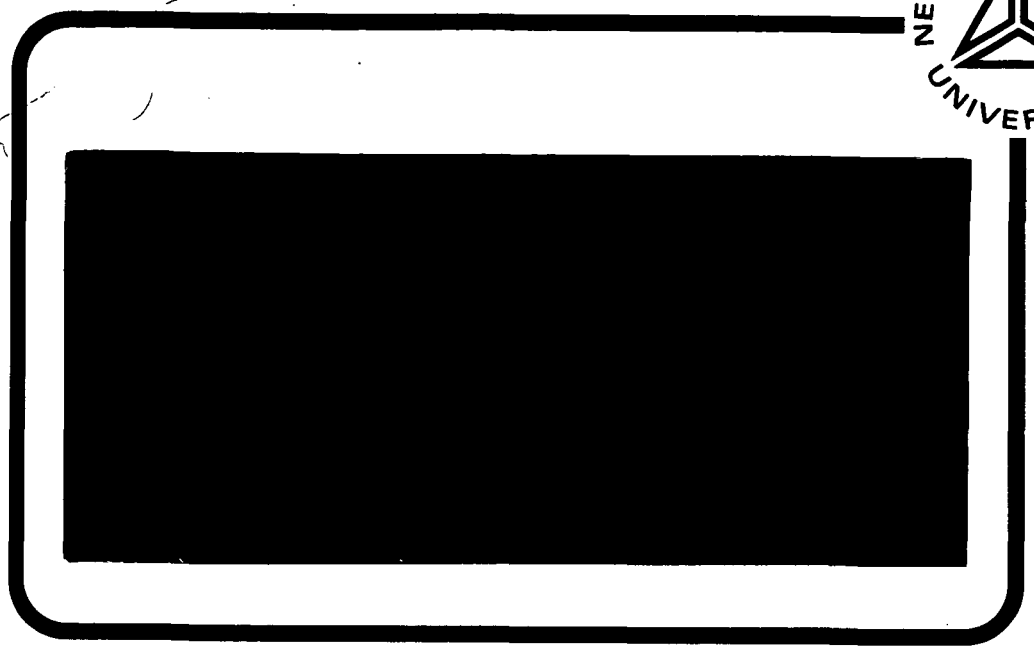

THE KLIPSCH SCHOOL OF

# ELECTRICAL AND COMPUTER ENGINEERING

TECHNICAL REPORT SERIES

# CODED THROUGHPUT PERFORMANCE SIMULATIONS FOR THE TIME-VARYING SATELLITE CHANNEL

**Li Han**

**NMSU-ECE-95-010 December 1995**

# CODED THROUGHPUT PERFORMANCE SIMULATIONS

# FOR THE

# TIME-VARYING SATELLITE CHANNEL

BY

LI HAN, B.S.

A Thesis submitted to the Graduate School

in partial fulfillment of the requirements

for the Degree

Master of Science in Electrical Engineering

New Mexico State University

Las Cruces, New Mexico

December 1995

"Coded Throughput Performance Simulations for the Time-Varying Satellite Channel," a thesis prepared by Li Han in partial fulfillment of the requirements for the degree, Master of Science, has been approved and accepted by the following:

---

Timothy J. Pettibone
Dean of the Graduate School

---

William E. Ryan
Chair of the Examining Committee

---

Date

Committee in charge:

Dr. William E. Ryan, Chair

Dr. Sheila B. Horan

Dr. James P. Reilly

Dr. Michael D. Ross

## ACKNOWLEDGMENTS

First, I deeply appreciate Dr. William E. Ryan, my thesis advisor, for his excellent direction of this research work, his always helpful suggestions, his help in organizing this thesis, and his suggestions as the writing evolved. It was really my pleasure to work with him.

I also thank Dr. William Osborne for his helpful advise, suggestions and discussions, Dr. Sheila Horan for her discussion on the Poisson process as well as her friendship, and Sharmin Ara for her help reading the first version of this thesis. I thank also Dr. James P. Reilly and Dr. Michael Ross serving on my thesis committee.

I thank NASA Goddard Space Flight Center for their support under Grant NAG 5-1491 and giving me the chance to work on this interesting project.

Finally, I would like to give special thank to my family and wife for their continuing love, encouragement and support.

# VITA

[redacted]

1989--B.S. Degree in Opto-Electronic from Tianjin University, P. R. China

1989-1992--Power and Dispatching Communication Bureau, Ministry of Energy, P. R. China

1994-1995--Research Assistant, Manuel Lujan Jr. Space Tele-Engineering Center, New Mexico State University

## PUBLICATIONS

William E. Ryan and Li Han, "Modulation and Coding for Short-Duration Deterministically Time-Varying Channels," *1995 International Conference on Communications*, Seattle, WA.

William E. Ryan, Li Han, and Paul A. Quintana "Design of a Low-Orbit to Geostationary Satellite Link for Maximal Throughput," submitted, *IEEE Transactions on Communications*.

## FIELD OF STUDY

Major Field: Telecommunications

Design and Simulation of Telecommunication and Channel Coding

## ABSTRACT

CODED THROUGHPUT PERFORMANCE SIMULATIONS

FOR THE

TIME-VARYING SATELLITE CHANNEL

BY

LI HAN, B.S.

Master of Science in Electrical Engineering

New Mexico State University

Las Cruces, New Mexico, 1995

Dr. William E. Ryan, Chair

The design of a reliable satellite communication link involving the data transfer from a small, low-orbit satellite to a ground station, but through a geostationary satellite, was examined. In such a scenario, the received signal power to noise density ratio increases as the transmitting low-orbit satellite comes into view, and then decreases as it then departs, resulting in a short-duration, time-varying communication link. The optimal values of the small satellite antenna beamwidth, signaling rate, modulation scheme and the theoretical link throughput (in bits per day)

have been determined in [1]. The goal of this thesis is to choose a practical coding scheme which maximizes the daily link throughput while satisfying a prescribed probability of error requirement. We examine the throughput of both fixed rate and variable rate concatenated forward error correction (FEC) coding schemes for the additive white Gaussian noise (AWGN) channel, and then examine the effect of radio frequency interference (RFI) on the best coding scheme among them. Interleaving is used to mitigate degradation due to RFI. It was found that the variable rate concatenated coding scheme could achieve 74 percent of the theoretical throughput, equivalent to 1.11 Gbits/day based on the cutoff rate $R_0$. For comprasion, 87 percent is achievable for AWGN-only case.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF APPENDIX FIGURES

# Chapter 1

# INTRODUCTION

We are concerned with reliable data transmission from a small, low-earth-orbit (LEO) satellite to a ground station, but through a geostationary satellite as in Fig. 1.1. The advantage of this approach is that a single ground station, which tracks only the geostationary (GEO) satellite, may be shared by a multiplicity of small satellites. In such a scenario, the received carrier power to noise density ratio, $C/N_0$, will increase as the transmitting low-orbit satellite comes into view, and then decrease as it departs, resulting in a short-duration, time-varying satellite communication channel. $C/N_0$ and $E_c/N_0$ "profiles" are illustrated in Fig. 1.2, where $E_c$ is the energy per channel bit. The overall goal of the project is to design a modulation and coding scheme which maximizes the link throughput in bits per day for a given $E_c/N_0$ profile and a prescribed decoded error rate requirement.

The whole project has been divided up into the following two sub-projects. The first sub-project [2] is a link design that covers the orbital simulation and the cutoff rate $R_0$ analysis, where the cutoff rate $R_0$ is a measure of the Shannon channel capacity from information theory. Because the simulation and analysis results from the first sub-project will be used in this thesis, we will discuss these results later in

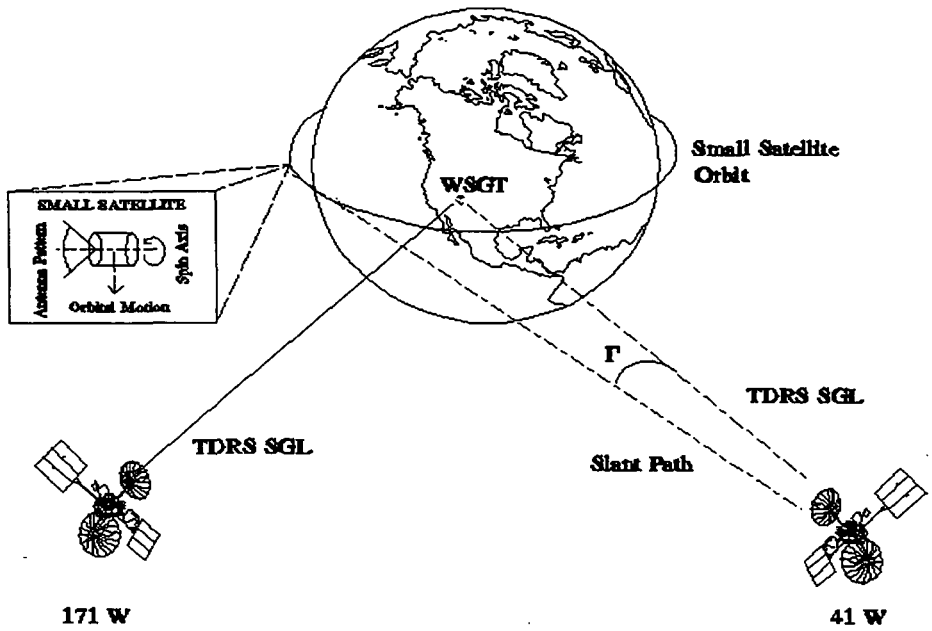


**Fig. 1.1** Studied Satellite Channel

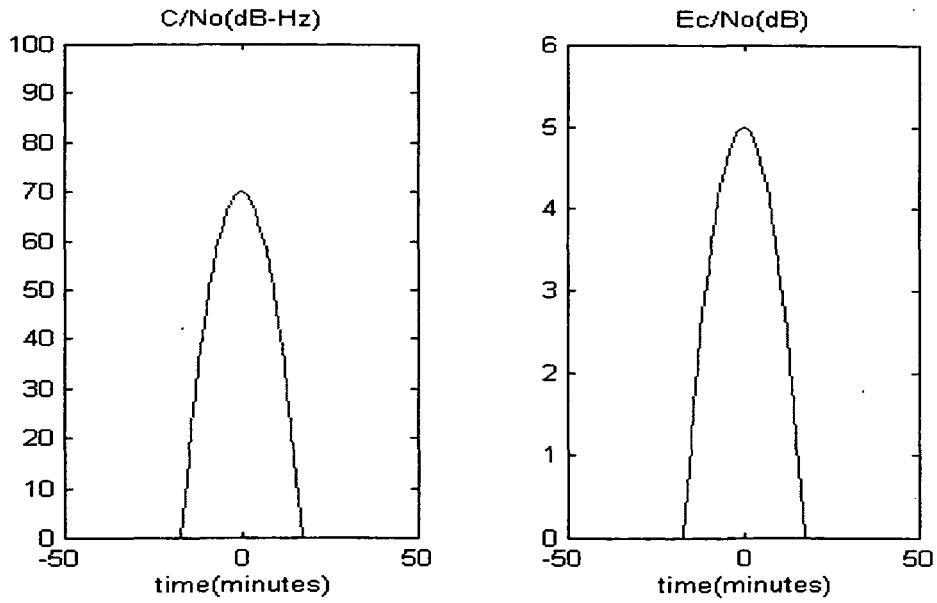


**Fig. 1.2** $C/N_0$ and $E_c/N_0$ profiles

this chapter. This thesis focuses on the second sub-project which is the error-control protocol design. The goal of the thesis is to find a near-optimal practical error-control coding scheme, where optimality is in the sense of maximizing the daily data throughput.

In this thesis, the effect of high-level pulsed RFI is considered in addition to AWGN. In practice, a power-limited low-orbit small satellite might be transmitting to a ground station via a geostationary satellite repeater at the same time and using the same frequency band as intentional or unintentional transmissions from a ground-based emitter (e.g., radar). When the data bit duration is shorter than that of the RFI pulses, the degradation can be very severe.

FEC coding schemes which concatenate convolutional with Reed-Solomon (RS) codes, with interleaving have already proved to be efficient counter-measures against RFI [3]. Both fixed and variable rate coding schemes will be examined in this thesis.

### 1.1 Review of the First Sub-Project Results

In the first sub-project, a LEO satellite with an orbital period of 102.86 minutes (14 orbits/day) and an inclination angle of 100 degrees was assumed. Further, the LEO satellite was assumed to have the following parameters.

- Spin stabilization in a nadir orientation, i.e., the long axis of the satellite intersects the center of the earth.
- Transmitter power: 3 W

- Antenna: circularly polarized, helical, non-gimbaled, pointing away from the earth's center. And the following 3 dB beamwidths $B$ are considered: 70, 52, 36, 28, 20, and 14 degrees.
- Frequency: S-band (2.2-2.3 Ghz)

The GEO satellite was taken to be NASA's Tracking and Data Relay Satellite (TDRS) located at 41 degrees west, in geostationary orbit. The TDRS parameters assumed were:

- G/T: 8.9 dB/K
- Antenna: 16 foot parabolic, circularly polarized, and capable of open-loop tracking.

The theoretical daily throughput, which is the number of information bits transmitted while $E_c(t)/N_0 \geq 0$ dB*, based on the cutoff rate $R_0$ is defined as

$$T_{R_0\_daily} = \int_{\substack{one \\ day}} R_0(t)dt \qquad \text{(bits)} \qquad (1.1)$$

where the integration is over the time in a day for which $E_c(t)/N_0 \geq 0$ dB.

Assuming MPSK signaling on an AWGN channel with two-sided power spectral density $N_0/2$, the soft-decision expression for $R_0(t)$ is then given [4] by

* In practice, carrier recovery for an MPSK signal is difficult below this value [5].

$$R_0(t) = R_c - R_c \log_M\left(\sum_{k=0}^{M-1} \exp\left(-\frac{E_s(t)}{N_0}\sin^2\left(\frac{\pi k}{M}\right)\right)\right) \quad \text{(bits/sec)} \qquad (1.2)$$

where $\frac{E_s(t)}{N_0}$ ($E_s = E_c \log_2 M$) is the symbol SNR, and $R_c$ is the code bit rate. From (1.1) and (1.2), we note that $T_{R_0}$ is a function of an $E_c/N_0$ profile together with the signal set size $M$.

The main result of the first sub-project shows that, for the given system scenario, the theoretical daily throughput $T_{R_0\_daily}$ is maximized at 1.5 Gbits when the small satellite antenna beamwidth uses 20 degrees, the modulation scheme uses binary phase shift key (BPSK) or QPSK, and uses $R_c$ = 4.5 Mcbps as the code bit rate. We also note that, with 20 degrees antenna beamwidth, the LEO and GEO satellites have two contacts per day, which last about (3 min + 4.4 min) = 7.4 min while $E_c(t)/N_0 \geq$ 0 dB.

In this thesis we estimate the throughput achievable by practical fixed and variable rate coding schemes, for both the AWGN and RFI situations. If we were to apply a code to the optimal ($B$ = 20 degrees, $R_c$ = 4.5 Mcbps, BPSK or QPSK) $E_c/N_0$ profile of the first sub-project to simulate its throughput, the simulation time would be unmanageable because the number of code bits for this optimal case is approximately (7.4 min)(60 min/sec)(4.5 Mcbps) $\approx 2 \times 10^{10}$ code bits. Thus, we instead consider a single contact in the total $C/N_0$ profile whose form lends itself to analysis. Specifically, we shall assume a $C/N_0$ profile (actually, contact) of the form

$$C(t)/N_0 = A_0 \exp(-t^2/2\sigma^2) \tag{1.3}$$

where the maximum value, $A_0$, of $C(t)/N_0$ occurs at time t = 0 by our choice of time origin, and $\sigma$ is a measure of the duration of the communication link. Fig. 1.3 shows the Gaussian approximation of (1.3) for the $C/N_0$ profile.

The $C/N_0$ profile of (1.3) gives rise to an $E_c/N_0$ profile through the chosen code bit rate $R_c$:

$$E_c(t)/N_0 = (A_0/R_c) \exp(-t^2/2\sigma^2) \tag{1.4}$$

As mentioned, we shall assume communication is possible only when $E_c(t)/N_0 \geq 0$ dB. The interval $[-t_0, t_0]$ for which this is true is easily found by solving for $t_0$ in the equation $E_c(t)/N_0 = 1$, yielding

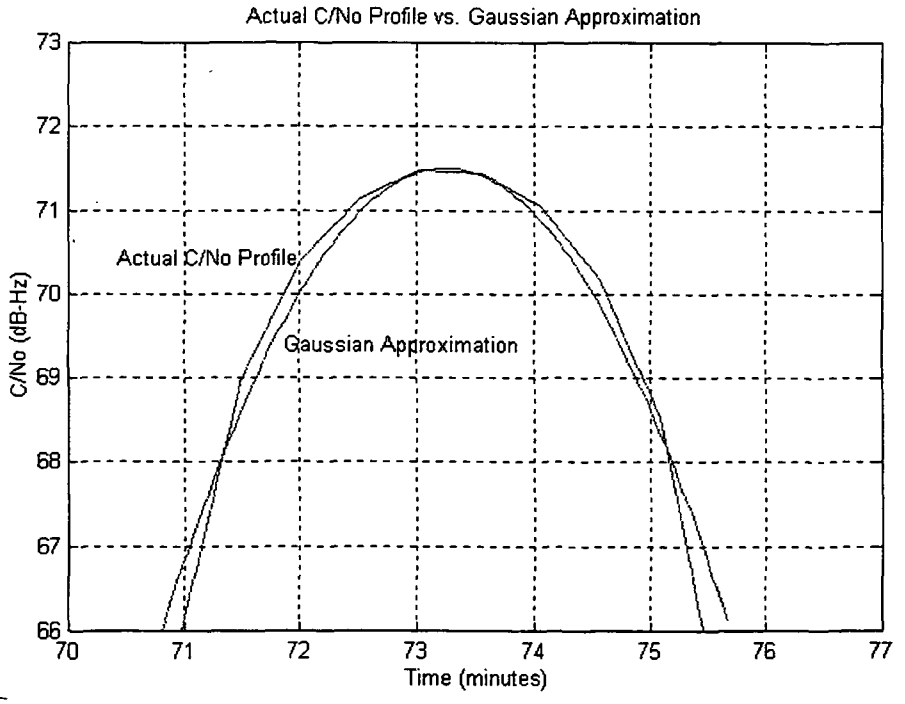


**Fig. 1.3** Gaussian Approximation of $C(t)/N_0$ Profile

$$t_0 = \sigma \sqrt{2 \ln(A_0 / R_c)} \tag{1.5}$$

which exists only if $R_c \leq A_0$.

Then the theoretical throughput for this single contact is

$$T_{R_0} = \int_{-t_0}^{t_0} R_0(t)dt \qquad \text{(bits)} \tag{1.6}$$

According to the analysis of [1], $T_{R_0}$ is maximized when $R_c = A_0/2$ so that $(E_c(t)/N_0)_{Max} = E_c(0)/N_0 \approx 3$ dB.

Hence, hereafter in this thesis we shall assume the optimal $E_c/N_0$ profile is

$$E_c(t)/N_0 = 2 \exp(-t^2/2\sigma^2) \tag{1.7}$$

Further, we assume this profile is known a *priori* to the variable rate scheme since it is computable within reasonable accuracy from the known orbits of the communicating satellites.

Ryan and Han [1] also showed that if $t_0$ corresponds to 15,000 8-bit RS symbols, the maximum $T_{R_0}$ at the optimal signaling rate $R_c$ and optimal modulation scheme, BPSK or QPSK, was equal to 175,300 bits. This leads us to define the throughput efficiency $\mathcal{E}$ as $\mathcal{E} = T_{coded} / T_{R_0} = T_{coded} /175{,}300$, where $T_{coded}$ is the coded throughput for a single contact. We shall use the value of $t_0$ indicated above and, hence, this expression for $\mathcal{E}$ through out the thesis. Note that once the efficiency for a particular coding scheme is estimated as above, it may be applied to the daily

theoretical throughput $T_{R_0,daily}$, to estimate the daily practical throughput. That is, the estimated daily throughput for code $\mathcal{C}$ is given by

$$T_{C,daily} \approx \mathcal{E}_{\mathcal{C}} T_{R_0,daily} \tag{1.8}$$

## 1.2 Outline of the Rest of Thesis

The reminder of this thesis is outlined as follows. The system and channel model, including the RFI model, are described in Chapter 2. Then in Chapter 3, we numerically estimate the throughput efficiencies for the AWGN channel. Three coding schemes, one fixed rate and two variable rate schemes, will be examined. Both the numerical procedures and results will be discussed in detail. Chapter 4 will describe the effect of RFI on the best coding scheme of Chapter 3. We also discuss the convolutional interleaving technique. Finally, a summary and conclusions will be made in Chapter 5. In Appendix A, we will give a brief introduction to punctured convolutional codes, and in Appendix B we will describe all the programs used in this thesis, both in Matlab and in C, also give a sample simulation program, written in C, for the AWGN-plus-RFI channel.

# Chapter 2

# SYSTEM AND RFI MODELS

## 2.1 System Model

The complete system transmission model is shown in Fig. 2.1. The left-hand side is the small, low-orbit satellite transmission system, the right-hand side is the ground station receiving system, and in between (e.g., AWGN plus RFI) is the physical time-varying channel.

The FEC coding schemes employed in this system are concatenated codes which use a convolutional code as the inner code and an 8-bit-symbol RS code as the outer code. Because RFI will usually create a burst error, when this burst error covers more than one channel code bit, we add a convolutional interleaver between the inner convolutional code and the physical channel, which has the affect of scattering the RFI over widely separated code bits. The details of the convolutional interleaver will be discussed in Chapter 4.

From the view of the outer RS code, the section between a-a and b-b can be considered as a super-channel. It is known that, the errors at the output of the Viterbi decoder occur in bursts [6], thus this super-channel is a burst error channel. We may improve the performance of the RS code by adding another (symbol) interleaver

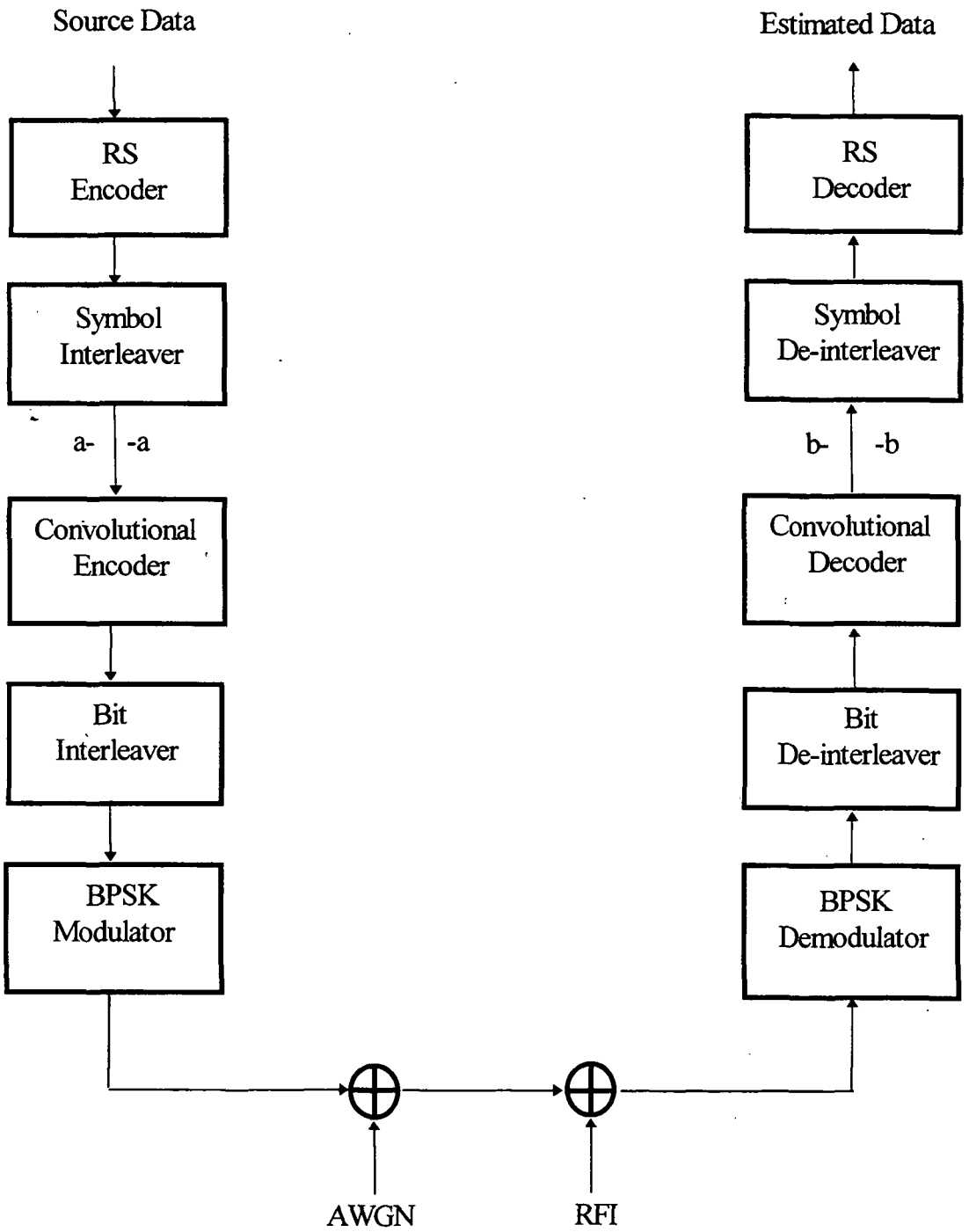


**Fig. 2.1** System Model

between the inner code and the outer code to make the errors out of this super-channel appear as random as possible.

The data transmission and reception process then proceeds as follows. The data at the transmitting terminal input are RS encoded, symbol interleaved, convolutionally encoded, and bit interleaved. The data are then BPSK (or QPSK) modulated and sent through the satellite channel. At the ground terminal receiver side, the received data need to go through the BPSK demodulator, bit de-interleaver, Viterbi decoder, symbol de-interleaver and RS decoder, in that sequence. We shall require that the probability that the RS decoder is unable to decode, $P_{cw}$, be no more than $10^{-5}$:

$$P_{cw} \leq 10^{-5}$$

We shall call $P_{cw}$ the probability of code-word error for simplicity.

## 2.2 Discrete-Time Equivalent Model for Simulation

Because computer simulations are necessarily performed in discrete time, it is necessary to derive an equivalent discrete-time system model for our simulations. The next sub-section does this for the AWGN channel and the sub-section after that does it for the AWGN-plus-RFI channel.

### 2.2.1 AWGN-Only Discrete-Time Model

In order to derive the discrete-time model, we begin with a simplified BPSK

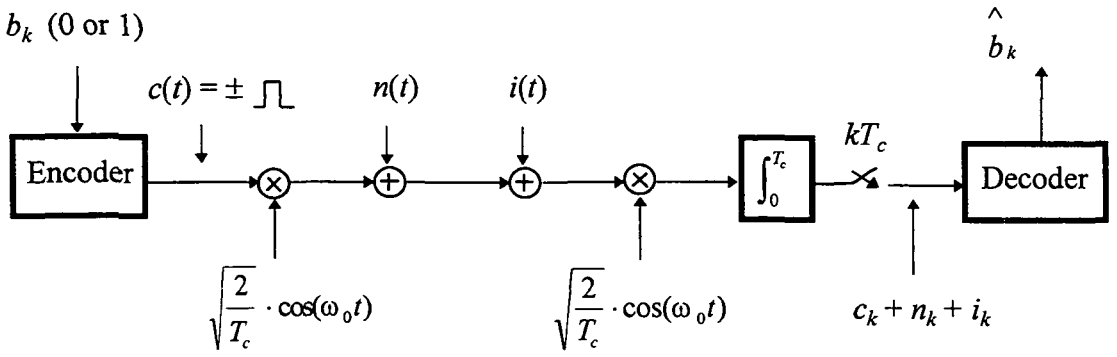


**Fig. 2.2** Simplified System Model

system model as in Fig. 2.2. Here $b_k$ is the $k^{th}$ information bit, $\omega_0$ is the transmitting carrier frequency, $T_c$ is the code bit duration, $n(t)$ is the AWGN continuous wave, $i(t)$ is the pulsed continuous wave (CW) RFI, and $c_k$, $n_k$ and $i_k$ are the signal, noise and RFI's values at sample times $kT_c$ , respectively. Finally, $\hat{b}_k$ is the estimated data.

When the additive noise is white and Gaussian with spectral density $N_0/2$, it is easy to derive [7] that $n_k$ is zero mean Gaussian with variance $N_0/2$ and we write $n_k \sim \eta(0, N_0/2)$. Further, it can easily be shown that $c_k = \pm 1$ so that $E_c = 1$. The resulting discrete-time AWGN-only channel model is then as shown in Fig. 2.3.

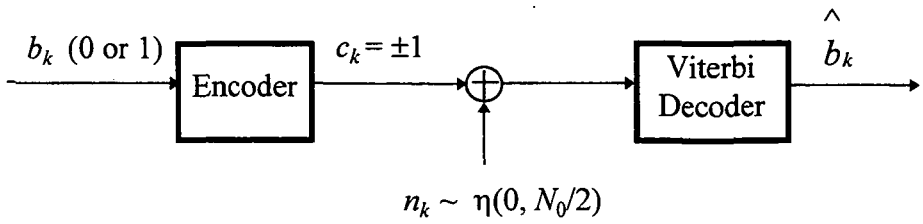


**Fig. 2.3** Discrete-Time AWGN-only Channel Model

### 2.2.2 AWGN plus RFI Discrete-Time Channel Model

The RFI model can be fully characterized by RFI type, RFI arrival rate, RFI-to-code bit duration ratio $\beta$, and RFI power. In this section, we will first give the assumptions about these RFI parameters, then give the RFI model derivation, followed by some necessary RFI amplitude calibration.

*Assumptions*

- Only the pulsed CW RFI is considered in this thesis, and the CW RFI has the form

$$\begin{aligned} i(t) &= a \cdot \cos(\omega_I t + \phi) && \text{for} \quad \tau_1 < t < \tau_2 \\ &= 0 && \text{otherwise} \end{aligned} \tag{2.1}$$

  where $a$ is the RFI's pulse amplitude, $\omega_I$ is its radian frequency, $\phi$ is its random phase, and $\tau_1$ is its starting time, $\tau_2$ is its ending time, thus, the RFI's duration $\tau$ is equal to $\tau_2 - \tau_1$.

- Two kinds of RFI arrival models, periodic and Poisson, are studied. For the periodic arrival RFI, a duty cycle is used to characterize its arrival rate. For example, if the duty cycle is equal to $10^{-2}$, the RFI starts after every 100 channel bits. The Poisson process, a very important counting stochastic process, is the other RFI arrival model. The definition of the Poisson process, denoted by $N(t)$ with parameter mean $\lambda t$ and arrival rate $\lambda$, can be found in many books (such as [8], [9]). In this thesis, the thing we do need to know about Poisson process is not

the process itself but the time interval between two adjacent arrivals of RFI events, called the interarrival time. Fig. 2.4 shows the definition of the interarrival time, denoted by $T_n$. Ochi [8] proved that the interarrival time of a Poisson process with arrival rate $\lambda$ obeys an exponential probability law with mean $1/\lambda$. Thus, the probability density function (pdf) of the interarrival time is

$$p(T_n) = \lambda \cdot e^{-\lambda T_n} \tag{2.2}$$

Fig. 2.5 shows the example of both periodic and Poisson arrival models for RFI.

- The RFI-to-code bit duration ratio $\beta$ is defined as $\tau/T_c$ . Basically, this ratio tells us how many channel code bits will be affected by each RFI pulse. The total number of channel code bits affected by RFI is equal to either $\lceil \beta \rceil$ or $\lceil \beta \rceil + 1$ depending on the RFI's starting time $\tau_1$, where $\lceil \beta \rceil$ is the ceiling fuction of $\beta$

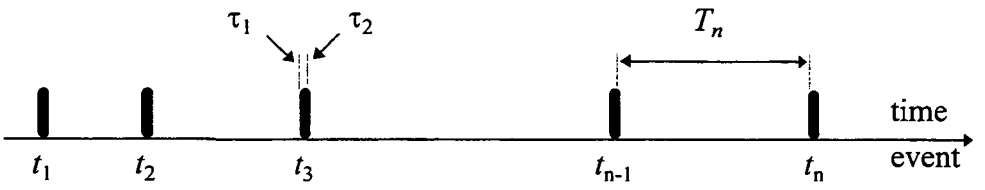


**Fig. 2.4** Definition of Interarrival Time

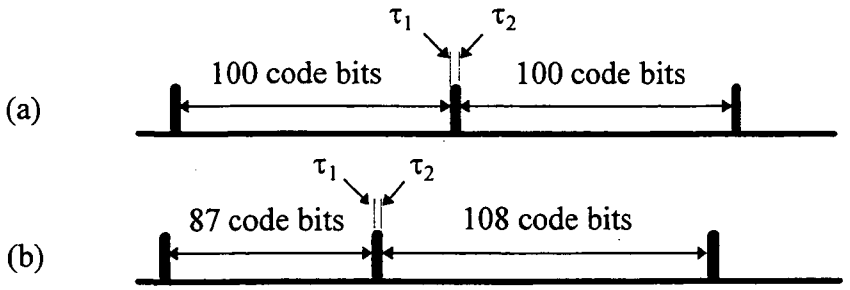


**Fig. 2.5** Examples of RFI arrival type (a) Periodic (b) Poisson

whose value leads to the next nearest integer from β. For example (see Fig. 2.6), when β = 3.4, then RFI affects

$$4 \text{ bits, if } \tau_1 \le 0.6 \qquad \text{or} \qquad 5 \text{ bits, if } \tau_1 > 0.6$$

In general, $\tau_1$ has the form $\tau_1 = K_1 \cdot T_c + K_2$, where $K_1$ is an integer and $0 \le K_2 < T_c$. However in the above example, we have simplified the notation, letting $\tau_1 = K_2$.

- In the literature (e.g., [10],[11]), some authors assume that RFI has infinite power. We shall instead assume that the power of the RFI is 20 dB more than that of the channel code bit.

*Model Derivation*

Base on the above assumptions, (2.1), and focusing on the RFI part of the system model of Fig. 2.2 (see Fig. 2.7), the discrete-time RFI value $i_k$ can be derived as follows. When the interval $(\tau_1, \tau_2)$ does not include the $k^{th}$ code bit, $i_k = 0$. Otherwise,

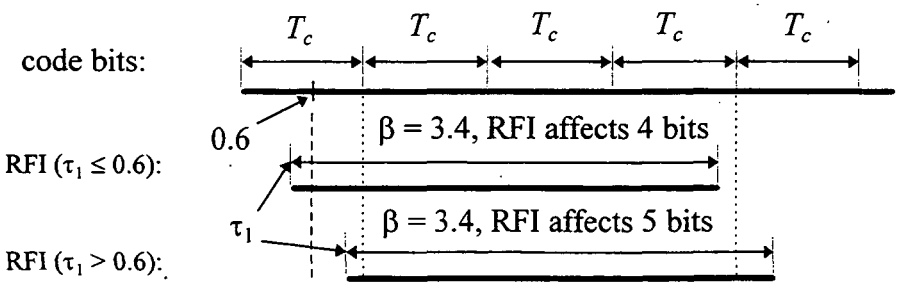


**Fig. 2.6** Examples of Total Number of Code Bits Affected by RFI

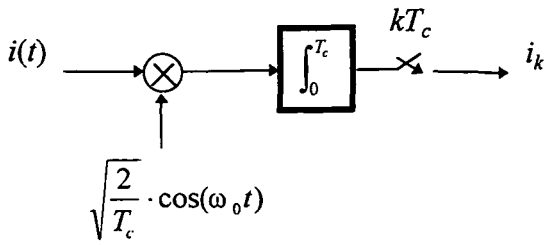


**Fig. 2.7** RFI Model

$$i_k = \int_{\max(\tau_1, kT_c)}^{\min(\tau_2, (k+1)T_c)} a \cdot \cos(\omega_I t + \phi) \cdot \sqrt{\frac{2}{T_c}} \cos(\omega_0 t) \cdot dt \tag{2.3}$$

$$= \int_{\max(\tau_1, kT_c)}^{\min(\tau_2, (k+1)T_c)} \frac{a}{2} \cdot \sqrt{\frac{2}{T_c}} \cdot \cos[(\omega_I - \omega_0) \cdot t + \phi] \cdot dt$$

$$+ \int_{\max(\tau_1, kT_c)}^{\min(\tau_2, (k+1)T_c)} \frac{a}{2} \cdot \sqrt{\frac{2}{T_c}} \cdot \cos[(\omega_0 + \omega_I) \cdot t + \phi] \cdot dt$$

The high frequency component (i.e., the second term) will be filtered out by the low pass filter (integrator), hence

$$i_k = \int_{\max(\tau_1, kT_c)}^{\min(\tau_2, (k+1)T_c)} \frac{a}{2} \cdot \sqrt{\frac{2}{T_c}} \cdot \cos[(\omega_I - \omega_0) \cdot t + \phi] \cdot dt \tag{2.4}$$

Now defining $\Delta\omega = (\omega_I - \omega_0)$, and substituting into (2.4), we obtain

$$i_k = \int_{\max(\tau_1, kT_c)}^{\min(\tau_2, (k+1)T_c)} \frac{a}{2} \cdot \sqrt{\frac{2}{T_c}} \cdot \cos[\Delta\omega \cdot t + \phi] \cdot dt \tag{2.5}$$

$$= \frac{a}{2 \cdot \Delta\omega} \cdot \sqrt{\frac{2}{T_c}} \cdot \sin(\Delta\omega \cdot t + \phi) \Big|_{\max(\tau_1, kT_c)}^{\min(\tau_2, (k+1)T_c)} \tag{2.6}$$

By assuming that the satellite system has a baseband bandwidth of $1/T_c$, we find it useful to write $\Delta\omega$ as

$$\Delta\omega = 2\pi \cdot (\alpha / T_c) \qquad \text{for } 0 < \alpha < 1 \qquad (2.7)$$

Finally, the discrete-time channel model for the AWGN-plus-RFI case is shown in Fig. 2.8.

*Calibration*

In this section, we calibrate the RFI amplitude, *a*, based on the system model we used and employing our assumption of an RFI-to-signal power ratio of 20 dB. We have

$$P_{RFI} = 100 \cdot P_{BPSK} = 100 \cdot (E_c / T_c) = 100 / T_c \qquad (\text{since } E_c = 1) \qquad (2.8)$$

But $$P_{RFI} = \frac{1}{T_c} \cdot \int_0^{T_c} i(t)^2 \, dt$$

$$= \frac{1}{T_c} \int_0^{T_c} a^2 \cos^2(\omega_I \cdot t + \phi) dt$$

$$= \frac{a^2}{2}$$

Substituting above into (2.8), we have

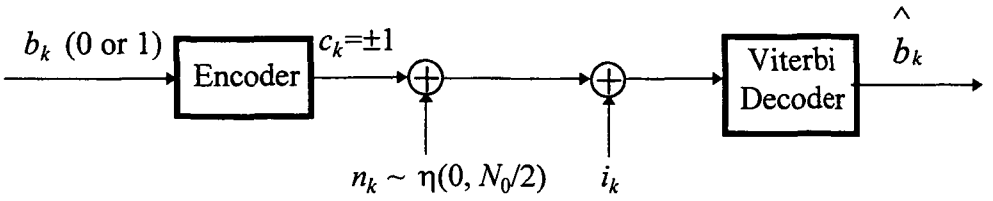


**Fig. 2.8** Discrete-Time Channel Model

$$P_{RFI} = \frac{a^2}{2} = \frac{100}{T_c} \qquad \Rightarrow \qquad a = \sqrt{\frac{200}{T_c}} \tag{2.9}$$

Now by substituting (2.7) and (2.9) into (2.6), we can further simplify the discrete-time RFI value $i_k$ to

$$i_k = \frac{\sqrt{\frac{200}{T_c}} \cdot \sqrt{\frac{2}{T_c}}}{2 \cdot 2\pi \cdot \alpha / T_c} \cdot \sin\left(\frac{2\pi \cdot \alpha}{T_c} \cdot t + \phi\right)\Big|_{\max(\tau_1, kT_c)}^{\min(\tau_2, (k+1)T_c)}$$

$$= \frac{5}{\alpha\pi}\left[\sin\left[\frac{2\pi\alpha}{T_c}\min(\tau_2, (k+1)T_c) + \phi\right] - \sin\left[\frac{2\pi\alpha}{T_c}\max(\tau_1, kT_c) + \phi\right]\right] \tag{2.10}$$

# Chapter 3

# THROUGHPUT ESTIMATION METHODS AND RESULTS FOR THE AWGN-ONLY CHANNEL

## 3.1 Selected Coding Schemes

One fixed rate and two variable rate concatenated FEC coding schemes are examined for the AWGN channel in this thesis, as shown below:

*Fixed Rate Coding Scheme*

Code 1: rate 1/2, constraint length 7 convolutional (inner) code concatenated with a (255, 223) RS (outer) code over Galois Field (GF) (256) [6] (see Fig. 3.1). This coding scheme is the NASA standard.

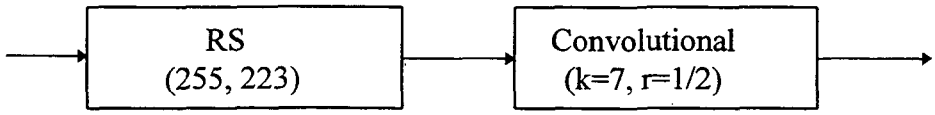


**Fig. 3.1** Concatenated FEC Coding Scheme 1 (Fixed Rate)

*Variable Rate Coding Schemes*

Code 2: variable rate (1/2, 2/3, 3/4, 4/5), constraint length 7 convolutional (inner) code concatenated with a (255, 223) RS (outer) code over GF(256) (see Fig. 3.2). For convolutional code rates larger than 1/2, we assume punctured

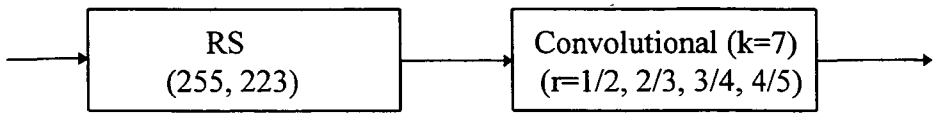


**Fig. 3.2** Concatenated FEC Coding Scheme 2 (Variable Rate)

coding [12]. Also, we considered higher rate codes (e.g., 5/6, 6/7, ...), but the $P_{cw}$ was never achievable at these rates.

Code 3: variable rate (1/2, 2/3, 3/4, 4/5), constraint length 7 convolutional (inner) code concatenated with a (255, $k$=223,225,...,255) RS (outer) code over GF(256) (see Fig. 3.3).

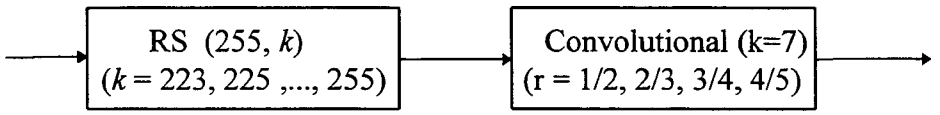


**Fig. 3.3** Concatenated FEC Coding Scheme 3 (Variable Rate)

By using variable rate coding schemes, better throughput efficiency can be obtained, and hence better daily throughput. But the complexity of the system is increased. This tradeoff needs to be considered when a final decision which is made on which coding scheme is to be used.

### 3.2 Throughput Estimation Procedures and Their Program Flow Charts

Before we discuss the throughput estimation procedures, the relevant system conditions, requirements, and assumptions are summarized in Table 3.1. As mentioned, we will estimate the coded throughput efficiencies in one contact time interval $[-t_0, t_0]$. Since it is easier to perform our computations when time is in units of

symbol number, we let $t_0$ corresponds to 15,000* 8-bit symbols. Then the $E_c$ /$N_0$ profile becomes

$$E_c(s)/N_0 = 2\ exp(-s^2/2\sigma^2) \tag{3.1}$$

where $s$ is the symbol number, and $s \in$ [-15000, 15000]. The value of the variance can be determined from our earlier assumption of $E_c$ /$N_0$ = 0 dB at time $s = \pm$ 15000:

$$2\ exp(-15000^2/2\sigma^2) = 0\ \text{dB} = 1 \qquad \Rightarrow \qquad \sigma = 12{,}740 \tag{3.2}$$

Such an $E_c$ /$N_0$ profile is given in Fig. 3.4.

Since the error rate performances of both the RS codes and the convolutional codes for the AWGN channel are well known (see below), it is possible to compute these quantities without actually simulating the system of Fig. 2.3.

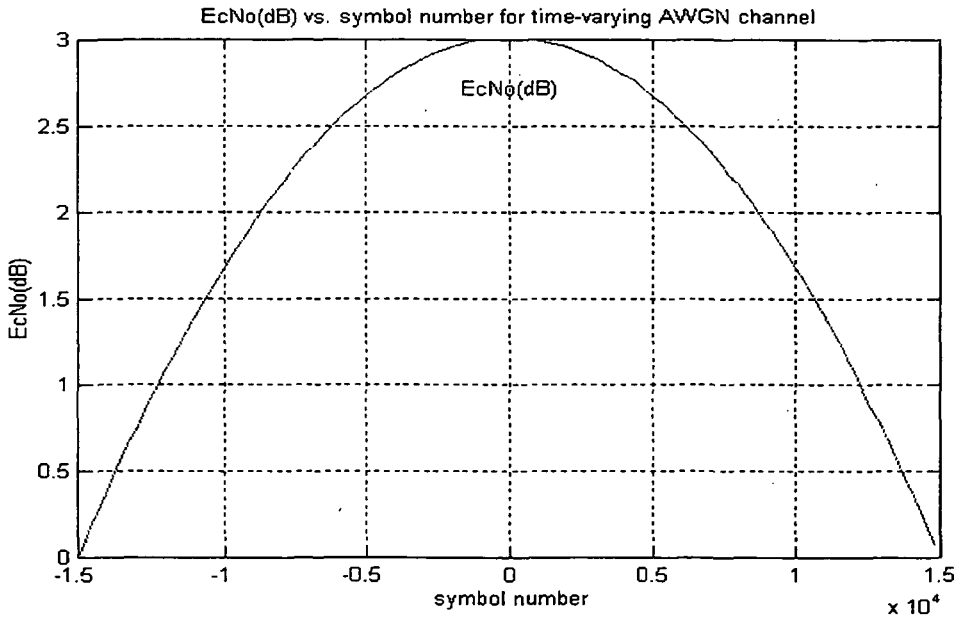


**Fig. 3.4** Time-Varying Gaussian Shaped $E_c$ (*s*)/$N_0$ Channel Profile

* This number has to be large enough to accommodate many RS codewords.

**Table 3.1** Summary of the Simulation System

| Items | Requirements |
|---|---|
| $E_c/N_0$ Profile | $E_c(s)/N_0 = 2\ exp(-s^2/2\sigma^2)$, $s \in [-15000,15000]$, $\sigma = 12{,}740$ |
| Modulation Scheme | BPSK |
| FEC Coding Schemes | Concatenated Codes 1, 2, 3 |
| $P_{cw}$ Requirement | $P_{cw} \leq 10^{-5}$ |
| Assumptions | $E_c/N_0 \geq 0$, Symbol Interleaver Used |

As indicated in Table 3.1, we will require a decoded outer RS codeword error probability of $P_{cw} \leq 10^{-5}$. Hence, the coded throughput $T_e$ is actually the total number of data bits corresponding to $P_{cw} \leq 10^{-5}$. For code 3, the rate of the outer code and then the inner code was varied until $P_{cw} \leq 10^{-5}$ was attained.

The codeword error rate $P_{cw}$ is given [6] by

$$P_{cw} = \sum_{j=e+1}^{255} \binom{255}{j} P^j \cdot (1-P)^{255-j} \tag{3.3}$$

where $e = \left\lfloor \frac{255-k}{2} \right\rfloor$ is the error correction bound for the outer RS code. The quantity $P$ is the 8-bit symbol error probability at the output of the convolutional decoder

which, assuming interleaving between the two codes, is related to the convolutional decoder bit error rate $p$ as $P = (1-p)^8$. In turn, when Viterbi decoding is used, $p$ is upper bounded [6] as

$$p < \frac{1}{k'} \cdot \sum_{d=d_f}^{\infty} b_d \cdot Q\left(\sqrt{\frac{2dE_c}{N_0}}\right) \tag{3.4}$$

where the convolutional code has rate $k'/n'$, minimum free distance $d_f$, and information weight spectrum $\{b_d\}$. Although $E_c$ is time-varying, we assume that it is relatively constant over the duration of a codeword as will be the case in practice where several thousands codewords are transmitted per second.

We now consider the throughputs $T_a$, $a \in \{1,2,3\}$, of above three coding schemes. For both fixed and variable rate coding schemes, $T_a$ can be computed as follows

$$T_a = 8 \cdot (255) \sum_{c=1}^{N_{cw}} r(c) \qquad a \in \{1,2,3\} \tag{3.5}$$

where $N_{cw}$ is the number of codewords transmitted in the interval [-15000, 15000 ] which achieve $P_{cw} \leq 10^{-5}$ and $r(c)$ is the composite (convolutional plus RS) rate of the $c^{th}$ codeword. For the fixed rate code, $r(c) = \frac{1}{2} \cdot \frac{223}{255}$ for all $c$, so that $T_1 = (30{,}000) \cdot (1/2) \cdot (223/255) = 104{,}941$ bits

Finally, the throughput estimation procedure then consists of the following three steps:

Step 1: Generate a time-varying Gaussian shaped $E_c(s)/N_0$ channel as in Fig. 3.4,

where $s$ is the symbol number in the interval [-15000, 15000].

Step 2: For each $E_c(s)/N_0$ value, calculate $p$ by (3.4) and then calculate $P_{cw}$ by (3.3). If $P_{cw} < 10^{-5}$, increment $N_{cw}$ by 1. Otherwise, for code 2 and 3, adjust the rate(s) so that $P_{cw} \leq 10^{-5}$, and store $r(c)$ for later use.

Step 3: When all of the 30,000 symbols are exhausted*, calculate $T_a$ by (3.5), then compare them with the theoretical throughput $T_{R_0} = 175{,}300$ bits to obtain their efficiencies.

Fig. 3.5 and Fig. 3.6 show the flowcharts of the Matlab simulation programs for code 1 and code 3, respectively.

In Fig. 3.5, we first set the maximum symbol value $N_{symMax}$ = 15,000, and initialize the first symbol $N_{sym}(1) = -N_{symMax}$. $E_c(s)/N_0$ is then obtained from (3.1). $P_{cw}$ is then calculated from (3.3) and (3.4) and compared with our threshold $10^{-5}$. Then we move to the next symbol and update the parameters for that symbol, and repeat the process until $N_{sym}(c)$ extends $N_{symMax}$. Finally, we calculate the coded efficiency as $\varepsilon_a = T_a/175{,}300$, and save and plot the results.

For code 3, the procedure is similar to the that of above, except we need to adjust (using higher rate) the code rates so that $P_{cw}$ is just less than $10^{-5}$. To do this, At the beginning, we put the convolutional code rates into a vector, denoted by $\text{code\_rate}_{1x4} = [1/2, 2/3, 3/4, 4/5]$, and $i$ is the index of $\text{code\_rate}_{1x4}$. Then for each $E_c(s)/N_0$

* The last codeword will typically extend beyond the last (30,000$^{th}$) symbol

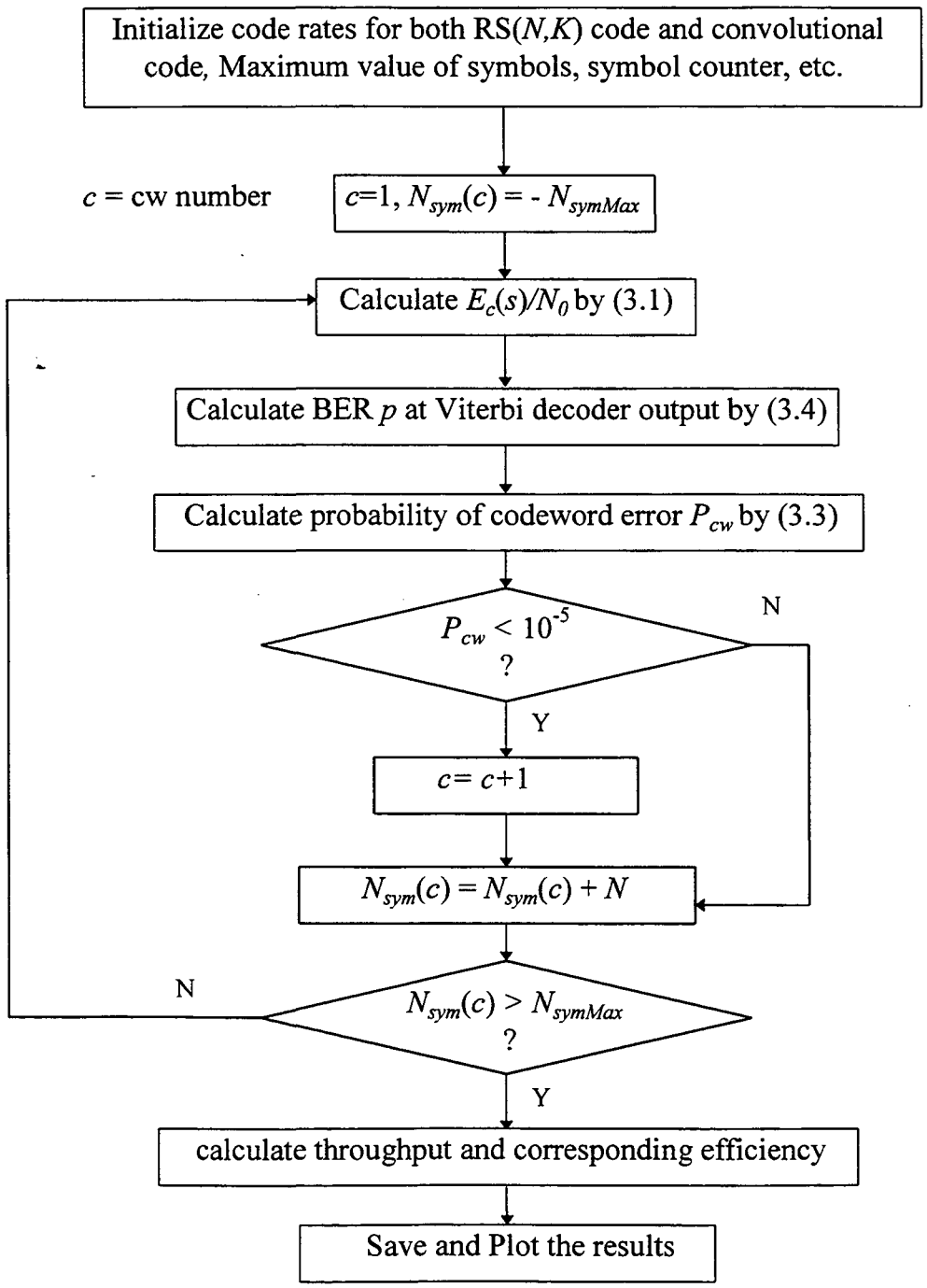


**Fig. 3.5** Flowchart of Matlab Program for Fixed Rate Coding Scheme

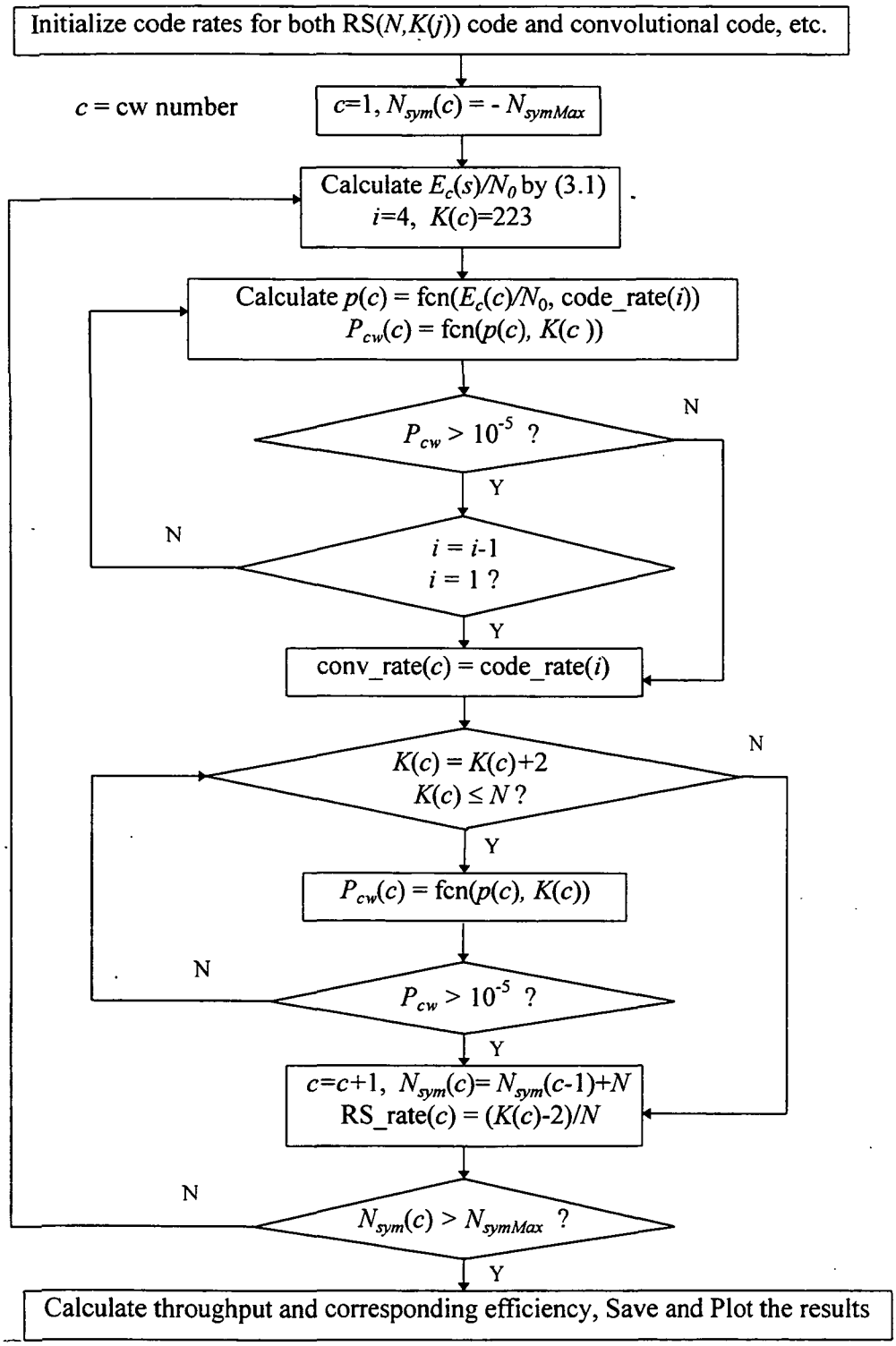


**Fig. 3.6** Flowchart of Matlab Program for Variable Rate Coding Scheme

value, we start using rate 4/5 first (i.e., start at $i = 4$), and set the code rate of RS $K(c)$ = 223. Then we calculate $p$ and $P_{cw}$ as before. If $P_{cw} \leq 10^{-5}$, keep the convolutional code rate but increment the RS rate, denoted by ($N$=255, $K(c)$), $K(c)$ by 2, and re-calculate $P_{cw}$ until $K(c) > N$ or $P_{cw} > 10^{-5}$. We then store the current convolutional code rate and previous RS rate (i.e., $K(c)$-2) for $s$th symbol (i.e. $i$th codeword). If $P_{cw} > 10^{-5}$, decrement convolutional code rate index $i$ by 1 (i.e., move to the next lower rate), re-calculate $P_{cw}$ until $P_{cw} \leq 10^{-5}$ or $i$=1, and then adjust $K(c)$ as above and store the final code rates for the later throughput calculation.

### 3.3 Numerical Results

Fig. 3.7 plots the $P_{cw}$ profile for coding scheme 1, the fixed rate scheme. From the $P_{cw}$ profile, we note that $P_{cw} \leq 10^{-5}$ for all 30,000 symbols, so that the throughput is $T_1$ = (30,000)·(1/2)·(223/255) = 104,941 bits, using the cutoff rate criterion, we observe the efficiency $\mathcal{E}_1 = T_1 / T_{R_0} = 0.60$.

The $P_{cw}$ profile also shows us how powerful this NASA standard coding scheme is, so that we do not need to go further to check the performance of longer constraint length codes. But on the other hand, we note that $P_{cw}$ is much less than the prescribed threshold of $10^{-5}$, which means that we waste a lot of coding overhead. In order to transfer as many information bits as possible, we try to use higher code rates in the higher SNR region, giving rise to coding schemes 2 and 3.

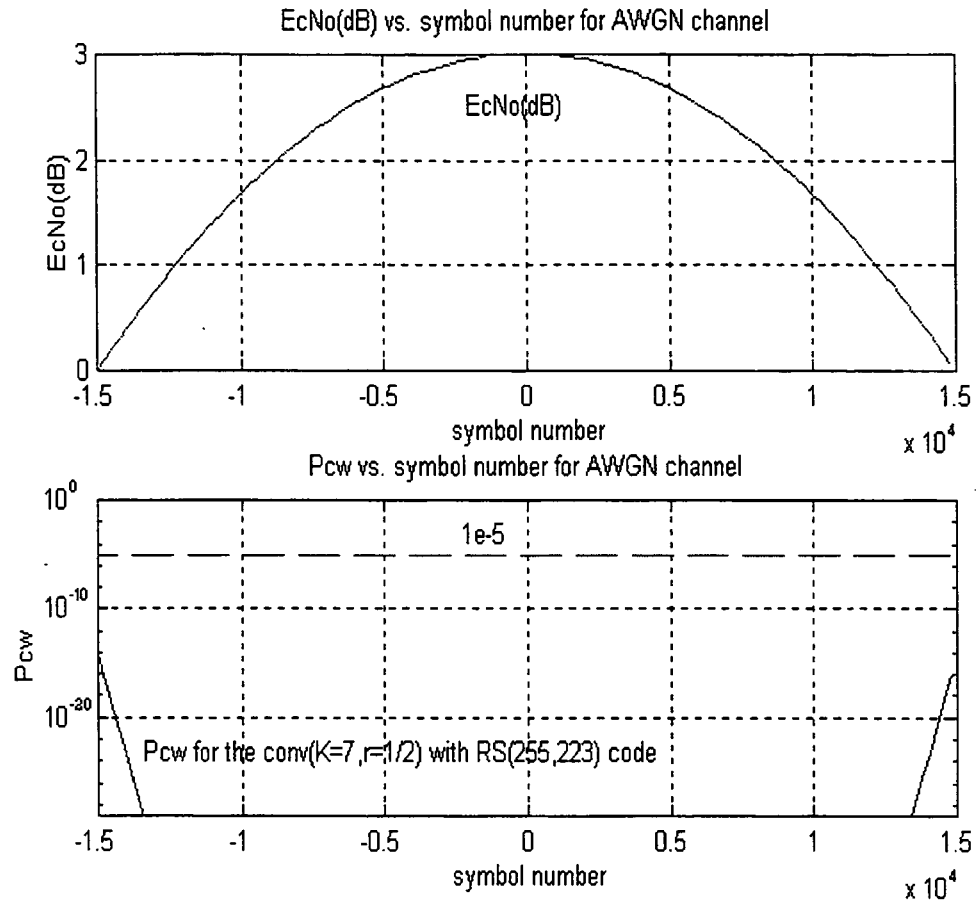


**Fig. 3.7** $P_{cw}$ Profile of Coding Scheme 1 for AWGN Channel

Fig. 3.8 and Fig 3.9 plot the $P_{cw}$ profiles and the varying code rates for coding scheme 2 and scheme 3, respectively. As can be seen, the $P_{cw}$ of coding scheme 2 is getting close to the threshold , and the $P_{cw}$ of coding scheme 3 is almost right below the threshold line. The throughputs for these two codes computed from (3.5) yield $T_2$ =144,740 bits, $T_3$ = 153,320 bits. The efficiencies are then $\mathcal{E}_2 = T_2 / T_{R_0} = 0.83$, $\mathcal{E}_3 = T_3 / T_{R_0} = 0.87$. This significant improvement is not surprising since the higher rate codes are used in the higher SNR region.

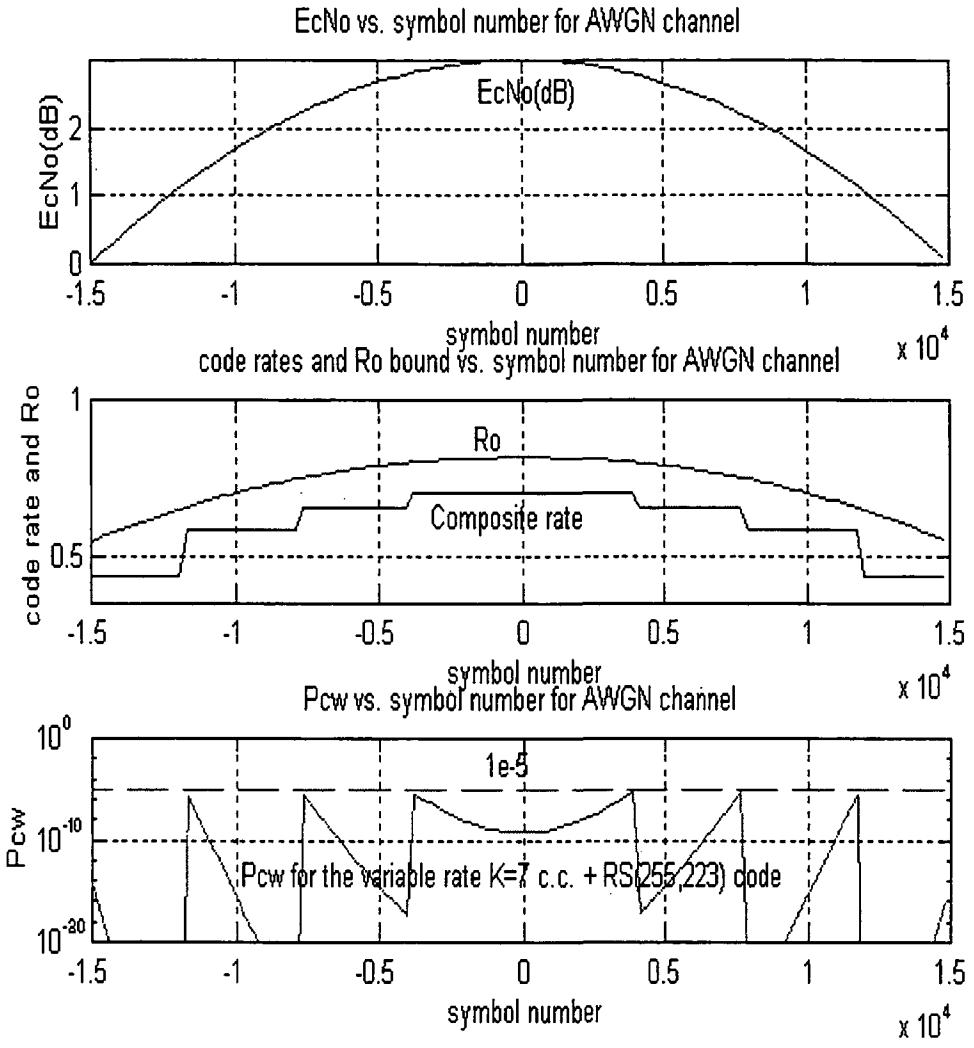


**Fig. 3.8** $P_{cw}$ Profile of Coding Scheme 2 for AWGN Channel

From these two figures, the varying of the code rate can be seen very clearly. In Fig. 3.8, we note that the convolutional code rate increasses when the $E_c/N_0$ level increases. Further, in Fig 3.9, the RS rate may increase with $E_c/N_0$ while the convolutional code rate remains constant. Hence, the composite code rate curve of coding scheme 3 is closer to the cutoff rate $R_0$ curve than that of coding scheme 2.

Table 3.2 summarizes the simulation results of the AWGN channel for the selected code schemes. As mentioned before, the throughput efficiency is obtained by considering a single contact of a total of 30,000 symbols, and the daily coded throughput is then obtained by multiplying the efficiency with the daily theoretical throughput of $T_{R_0}$ = 1.5 Gbits as determined in [2].

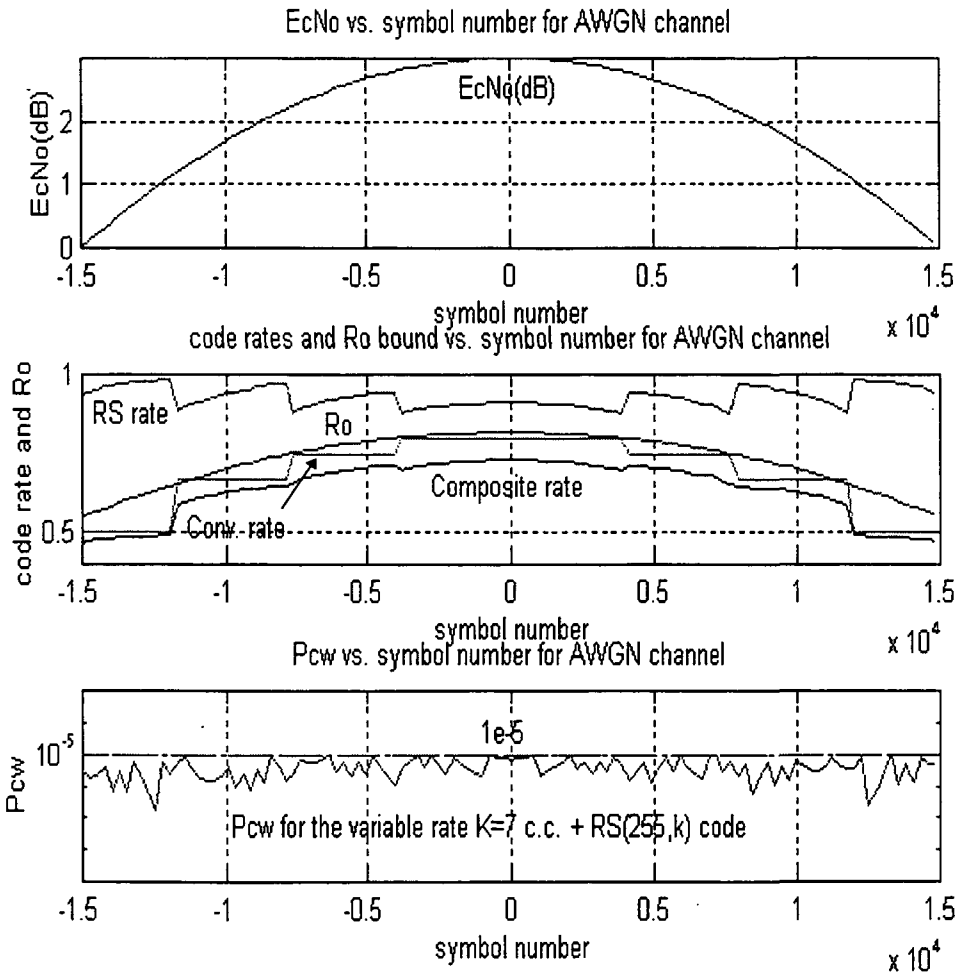


**Fig. 3.9** $P_{cw}$ Profile of Coding Scheme 3 for AWGN Channel

**Table 3.2** Summary of Simulation Results for AWGN Channel

| **Coding Schemes** | **Throughput Efficiencies** | **Daily Coded Throughputs (Daily Theoretical Throughput = 1.5 Gbits)** |
|---|---|---|
| Code 1 | 0.60 | 0.9 Gbits |
| Code 2 | 0.83 | 1.2 Gbits |
| Code 3 | 0.87 | 1.3 Gbits |

# Chapter 4

# EFFECT OF RFI ON THE BEST CODING SCHEME

## 4.1 Simulation Procedure

Code 3, the best coding scheme from the previous chapter, will be examined first for the AWGN-plus-RFI channel to observe how severely the RFI can affect the throughput performance. Based on this result, it will be decided whether the throughput performance for Code 1 and 2 on AWGN-plus-RFI channel needs to be checked.

Weinberg, [10] and [11], derived some formulas for the coded BER performance on the RFI channel, but unfortunately, these formulas are not easier than simulations. Thus we decided to obtain the coded throughput of AWGN-plus-RFI channel via simulations. For the given RFI scenario, several C programs have been written in order to simulate the convolutional coded bit error rate $p$ for rate 1/2 convolutional code and its punctured higher rate cousins (e.g., rate 2/3, 3/4 and 4/5). As mentioned, if the RFI affects more than one channel code bit, the convolutional interleaver will be used against RFI. A brief introduction of this interleaving technique will be given later in this Section.

The throughput simulation procedure for AWGN-plus-RFI channel is much like

that of the AWGN-only channel. The simulation is a combination of the both C programs and Matlab programs. Following is the throughput computation procedure for AWGN-plus-RFI channel. Again, we point out that Code 3 is assumed.

Step 1: Generate a time-varying Gaussian shaped $E_c(s)/N_0$ channel as in Fig. 3.4.

Step 2: For a sufficiently large number of $E_c/N_0$ values (e.g., 21), determine $p$ for the AWGN- plus-RFI scenario of interest via C program simulations, for code rates 1/2, 2/3, 3/4 and 4/5. Then produce $p$ vs. symbol number (time) curves by interpolating these measured point. ( Repeat for other RFI scenarios.)

Step 3: Calculate $P_{cw}$ by (3.3), adjust the convolutional code rate and then the RS code rate so that $P_{cw} \leq 10^{-5}$. Store the composite rate $r(s)$ for later use.

Step 4: Calculate $T_a$ by (3.5), then compare with the theoretical throughput $T_{R_0}$ (e.g., 175,300 bits) to obtain throughput efficiency.

*Convolutional Interleaving Technique*

The idea of interleaving is very simple. Basically, the order of a sequence of symbols is rearranged in a deterministic manner before the sequence is sent to the burst-error channel, and then at the receiver side, the inverse permutation is applied to restore the sequence to its original ordering.

The interleaving structure we shall discuss here was proposed by Ramsey [12]. An example of a convolutional interleaver and its corresponding de-interleaver with $m$ =

4 delay lines and $D$ = 1 symbol delay elements is shown in Fig. 4.1. As we can see, the input symbols are read onto the delay lines in a certain order, and the output of the delay lines is read in the same order.

Because of the separation provided by the delay elements, any burst of $b < m$ errors inserted by the channel results in single errors at the de-interleaver output. For example, in the figure, burst errors hit $x_8$, $x_5$, and $x_2$, but those symbols are separated by three at the output of de-interleaver. This implies that the good long burst-error-correction codes can be obtained from the good short burst-error-correction codes.

With interleaving, extra delay is needed for the decoder. Defining the parameter

$$L = mD \ (= m \text{ in our case in which } D=1)$$

the total end-to-end delay of convolutional interleaving is $m(m\text{-}1)$ symbols. Typically, the interleaver parameters $m$, $D$, and $L$ are selected as follows

$$m > \text{the length of the burst error}$$

$$L > \text{the decoding path length}$$

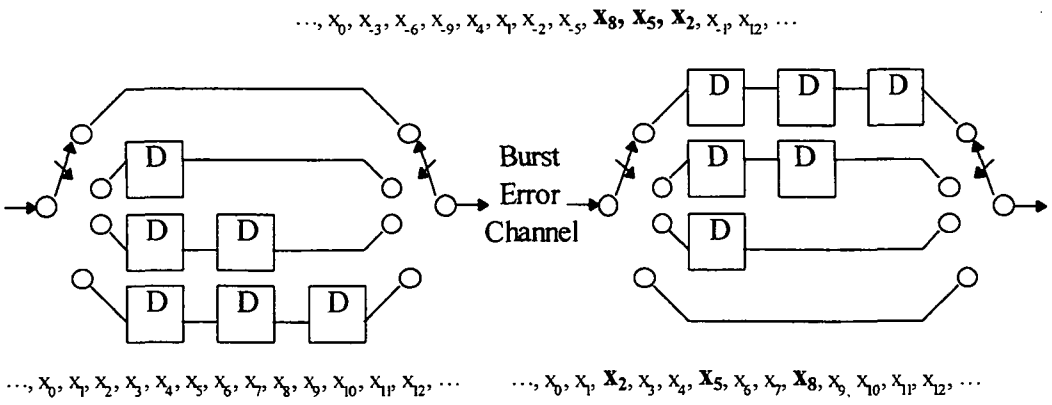


**Fig. 4.1** Convolutional Interleaver and Corresponding De-interleaver

$$D = 1$$

In this thesis, since the decoding path length is 32 for a rate 1/2 convolutional code and 64 for its higher rate (>1/2) punctured code, $L$ has to be chosen larger than 32. Since it is easier to use the same interleaver for all convolutional code rates in the implementation, we set the interleaver parameter $m$ to $m$ = 64 delay lines (although the rate 1/2 code really needs only 32 delay lines).

### 4.2 Simulation Results

In this section, the performance results of Code 3 for the AWGN-plus-RFI channel will be shown. Table 4.1 shows the RFI scenarios we examined in this thesis. In the table, $1/\lambda$ is the mean interarrival time for the Poisson arrival model. We have

**Table 4.1** RFI Scenarios Examined for the Channel

| **RFI ($\alpha$ = 0.1)** | **RFI/code-bit ratio $\beta$** | **arrival model** |
|---|---|---|
| Case 1 | 0.1 | Poisson (arrival rate $\lambda$ = 9.5e-3) |
| Case 2 | 0.1 | periodic (duty = 9.5e-3) |
| Case 3 | 5 | Poisson (arrival rate $\lambda$ = 7.43e-5) |
| Case 4 | 5 | periodic (duty = 7.23e-5) |
| Case 5 | 10 | Poisson (arrival rate $\lambda$ = 4.24e-5) |
| Case 6 | 10 | periodic (duty = 3.9e-5) |
| Case 7 | 20 | Poisson (arrival rate $\lambda$ = 2.23e-5) |
| Case 8 | 20 | periodic (duty = 2.07e-5) |

selected $\lambda$ in each $\beta > 1$ case so that $p_{rate\_2/3}$ (Poisson) $\approx p_{rate\_2/3}$ (Periodic) rather than $\lambda$ = duty cycle.

Fig. 4.2 plots the Case 1 results analogous to Fig. 3.9 except we have included the $p$ vs. symbol number curves based on simulations. The threshold of $p$ (i.e., 2.39e-3) is the highest value allowable to achieve the $P_{cw} \leq 10^{-5}$ requirement for the (255, 223) RS code, the most powerdul RS code. We note that p_4/5 is above this threshold line, so that rate 4/5 code is never selected as seen in the third plot. We also note that the p_1/2 curve becomes flat in the region of high $E_c/N_0$, because, in the high $E_c/N_0$ region, the RFI is more severe than the noise and dominates the BER $p$. Because the RFI/code bit duration ratio $\beta$ is very small (i.e., $\beta = 0.1$), the effect of RFI is limited, and we obtain 86.68 percent of theoretical throughput (i.e., 175,300 bits) for this RFI scenario, compared to the 87 percent on AWGN-only channel.

Fig. 4.3 plots the results for Case 2, which is the Case of periodic arrival RFI with $\beta$ =0.1. All four rate of convolutional codes can be used in this Case, and we obtain 86.13 percent of theoretical throughput this time.

For the $\beta$ = 5 Cases, Fig. 4.4 and Fig. 4.5 plot the results for Case 3 (Poisson arrival RFI) and Case 4 (Periodic arrival RFI), respectively. Only three rates (i.e., rate 1/2, 2/3, 3/4) convolutional codes were simulated since only the two lowest rate codes of them can be used to achieve $P_{cw} \leq 10^{-5}$ (see second and third plots in Fig. 4.4 and Fig. 4.5). Because RFI affects more than one code bit, interleaving is employed. For Case 3, we obtain 74.74 percent of theoretical throughput, and for Case 4, , we obtain

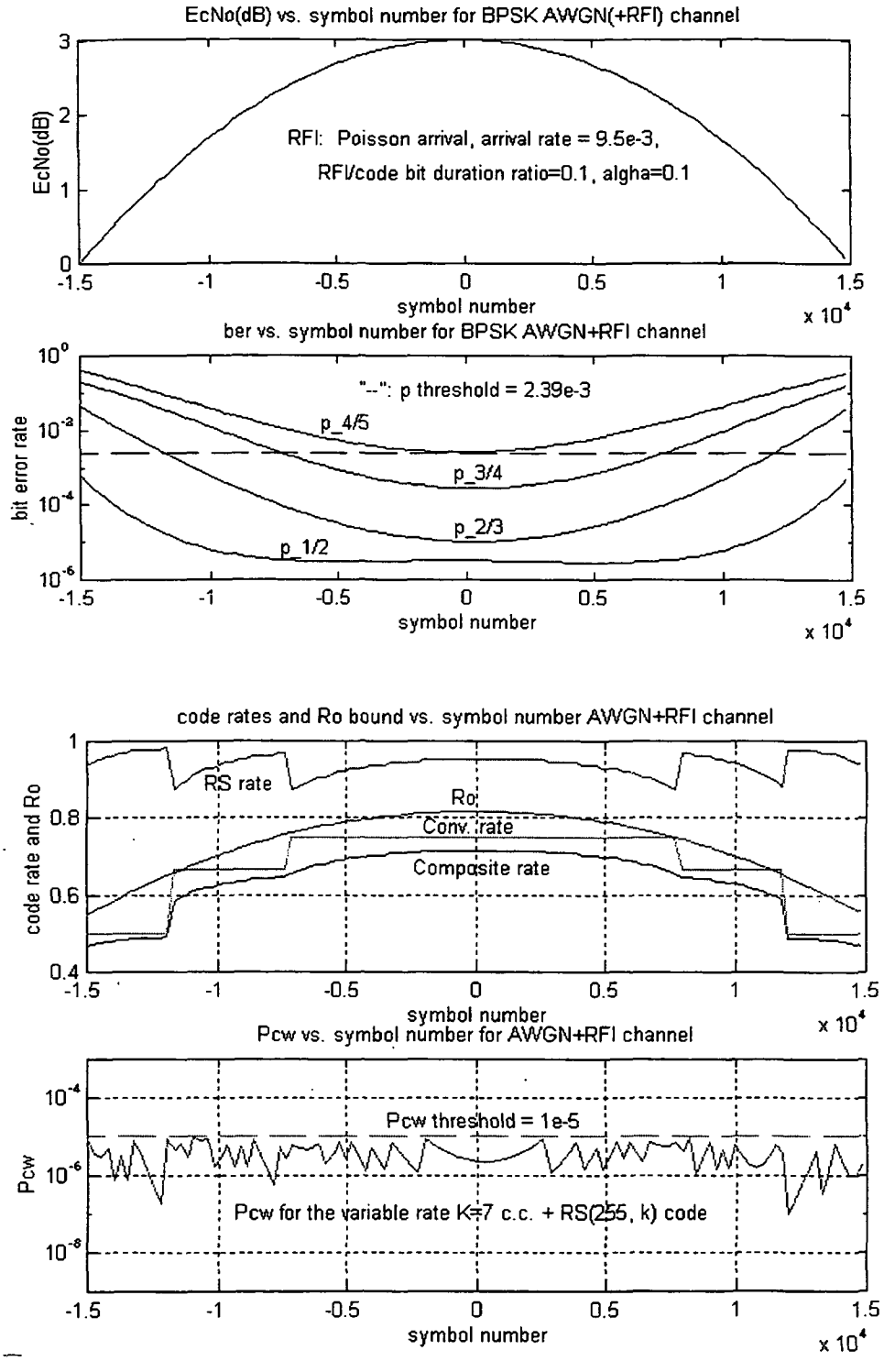


**Fig. 4.2** Results for Case 1

77.48 percent of theoretical throughput.

For the $\beta = 10$ Cases, Fig. 4.6 and Fig. 4.7 plot the results for Case 5 and Case 6, respectively. Again, only rate 1/2 and 2/3 convolutional codes can be used in Case 5 (shown in Fig. 4.6), but rate 3/4 code can be also used in Case 6 (shown in Fig. 4.7). For Case 5, we obtain 73.64 percent of theoretical throughput, and for Case 6, we obtain 74.34 percent of theoretical throughput.

Finally, the Cases of $\beta = 20$ was examined, and Fig. 4.8 and Fig. 4.9 plot the results for Case 7 and Case 8, respectively. Again, only rate 1/2 and 2/3 convolutional codes can be used (shown in Fig. 4.8 and Fig. 4.9). For Case 7, we obtain 74.84 percent of theoretical throughput, and for Case 8, we obtain 74.45 percent of theoretical throughput.

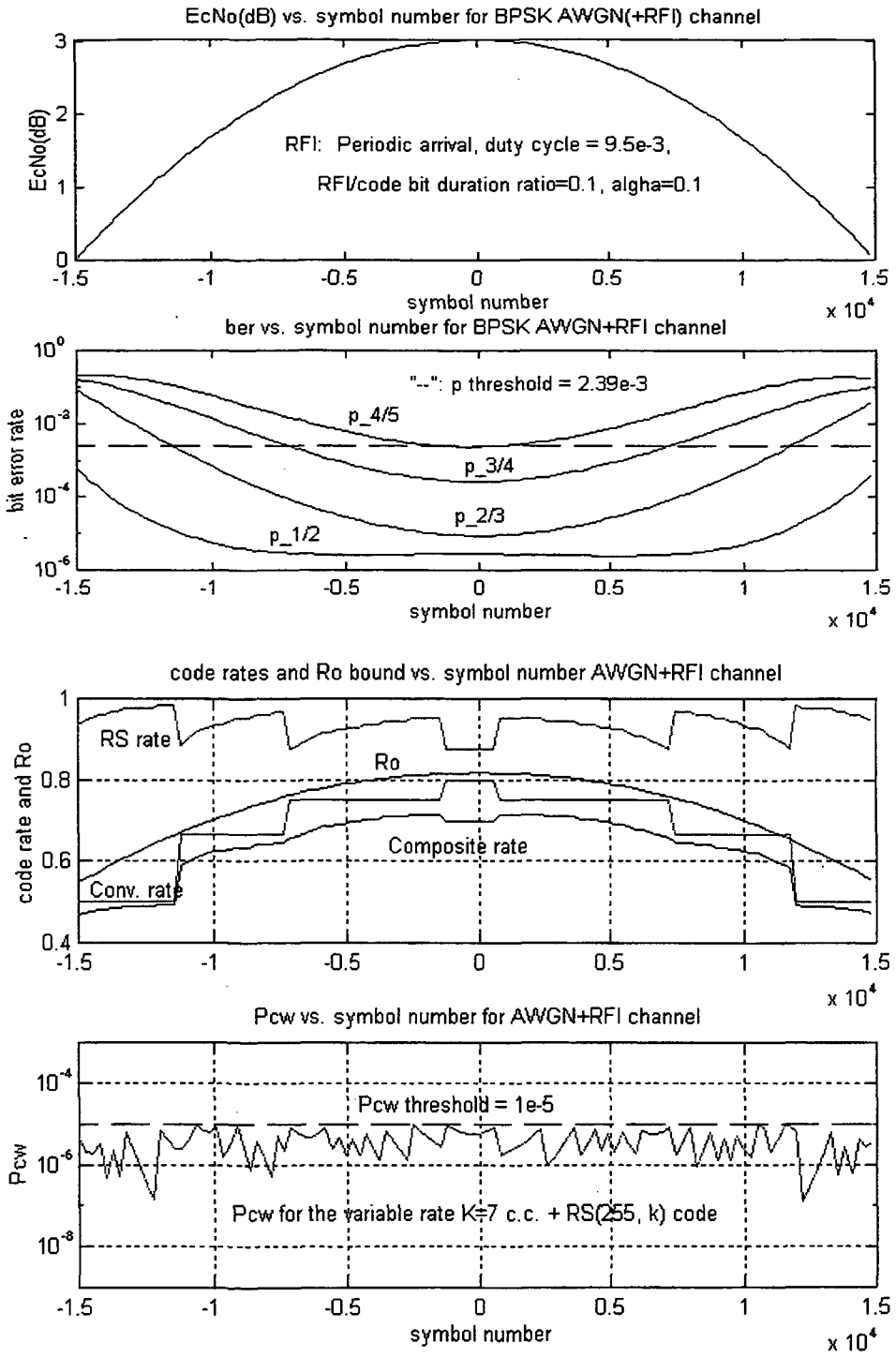


**Fig. 4.3** Results for Case 2

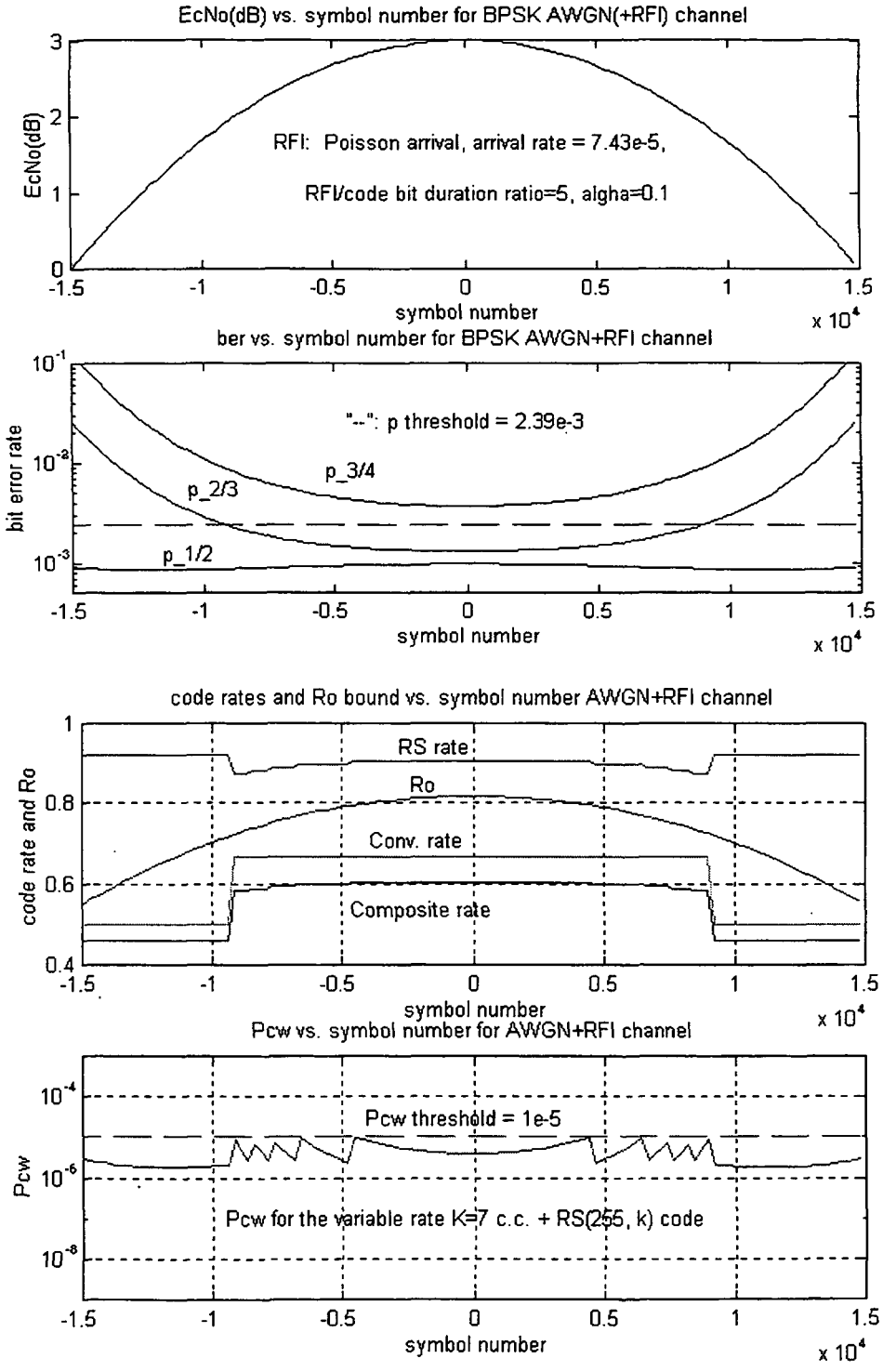


**Fig. 4.4** Results for Case 3

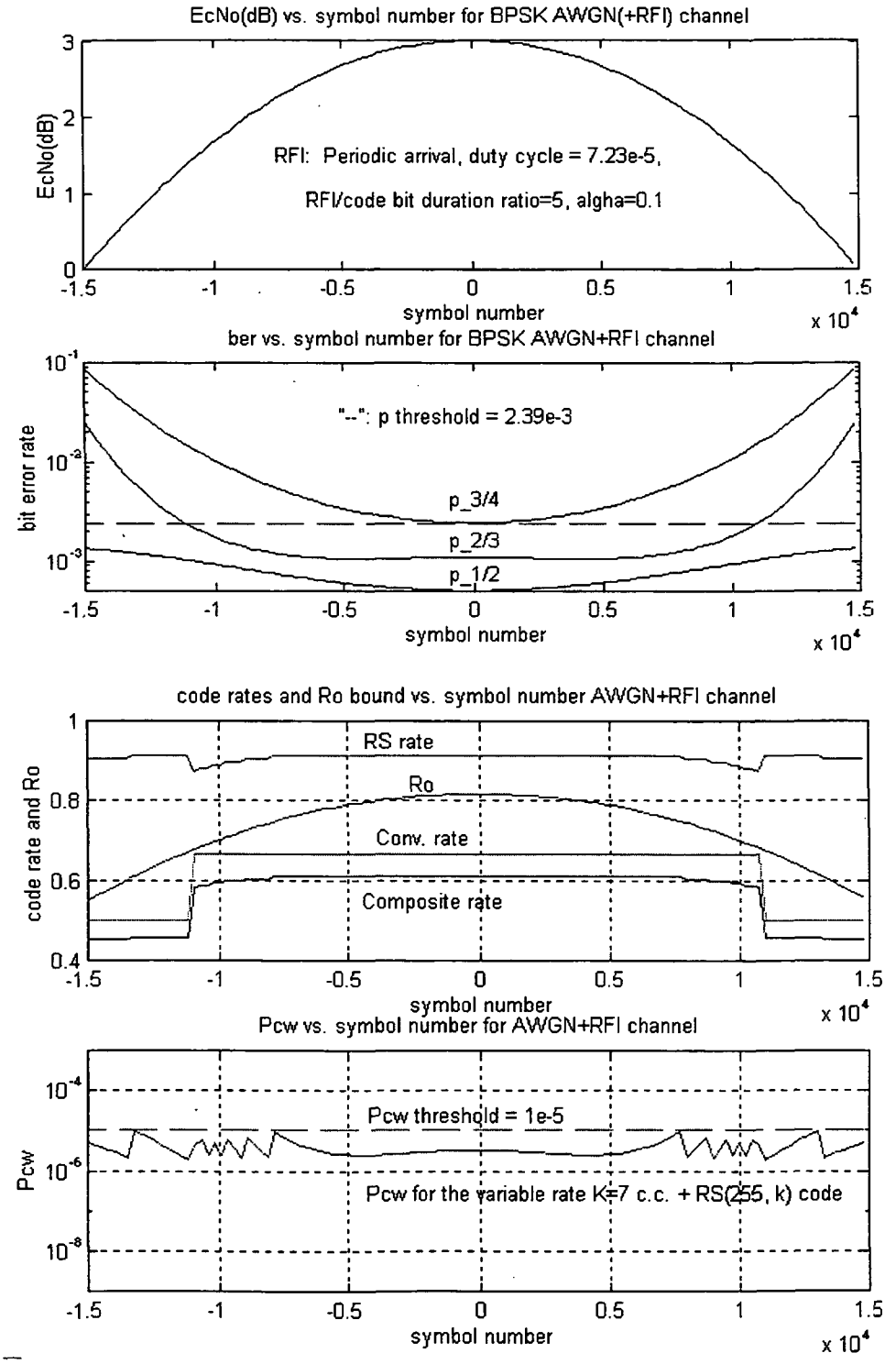


**Fig. 4.5** Results for Case 4

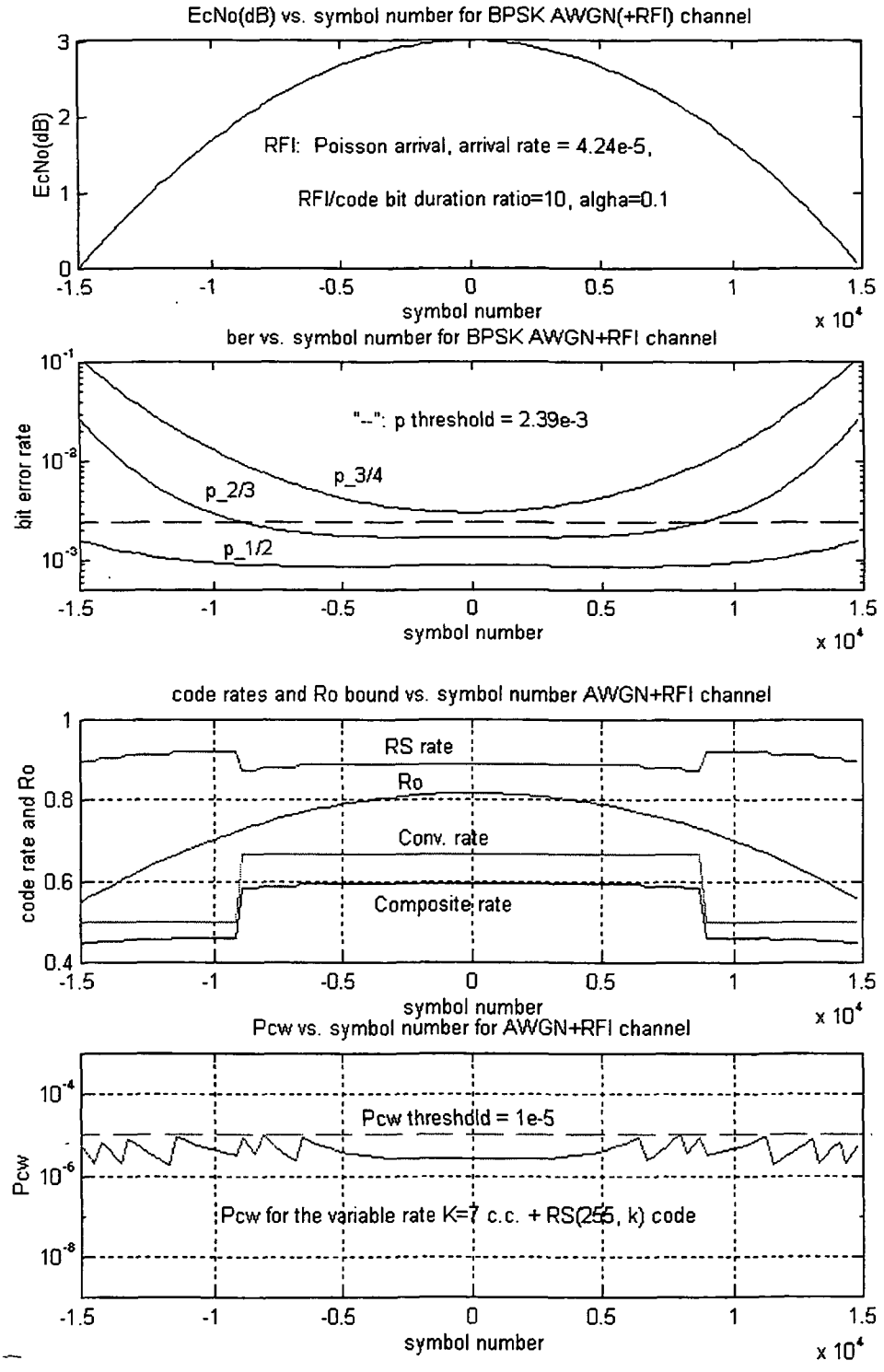


**Fig. 4.6** Results for Case 5

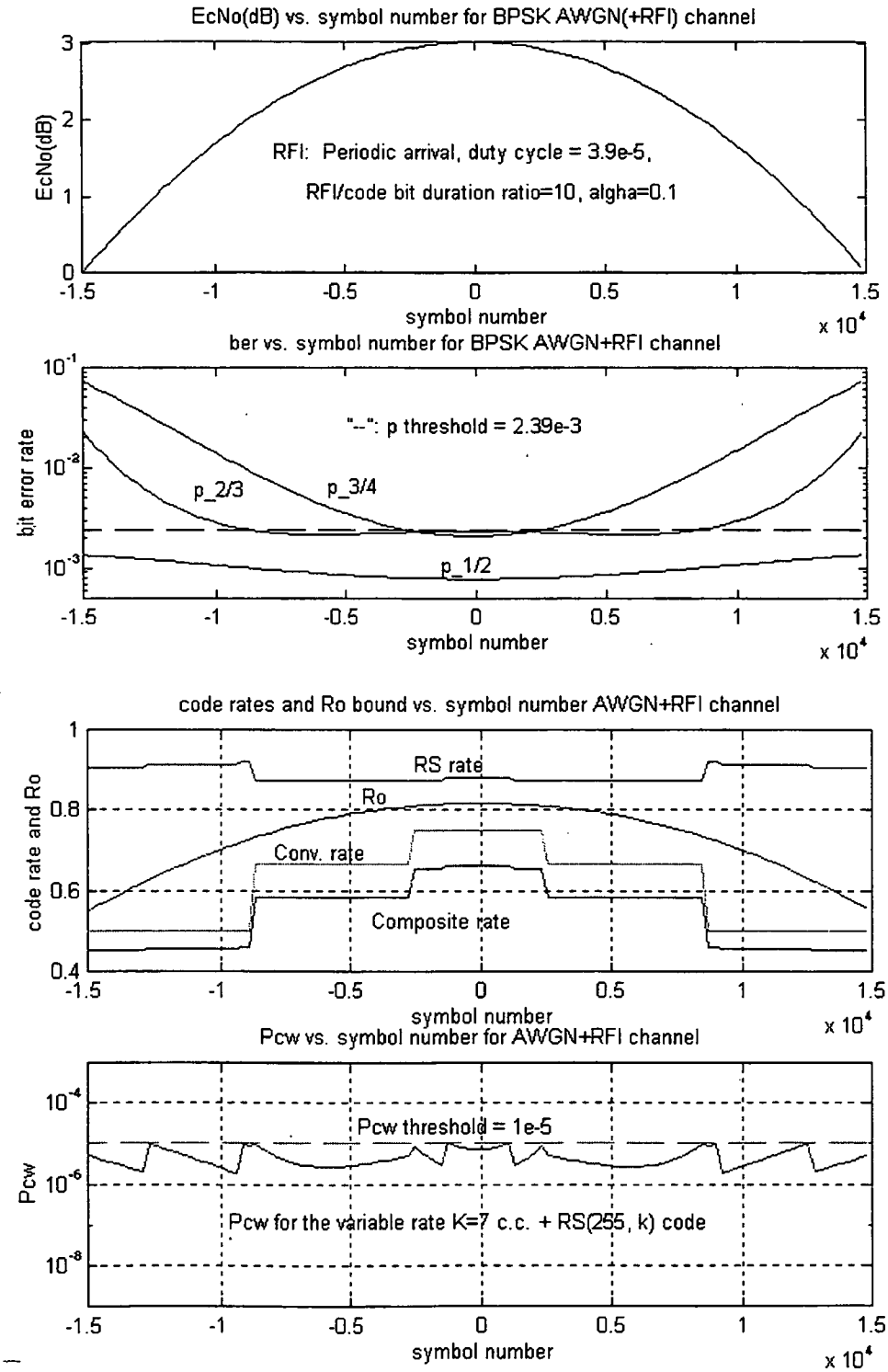


**Fig. 4.7** Results for Case 6

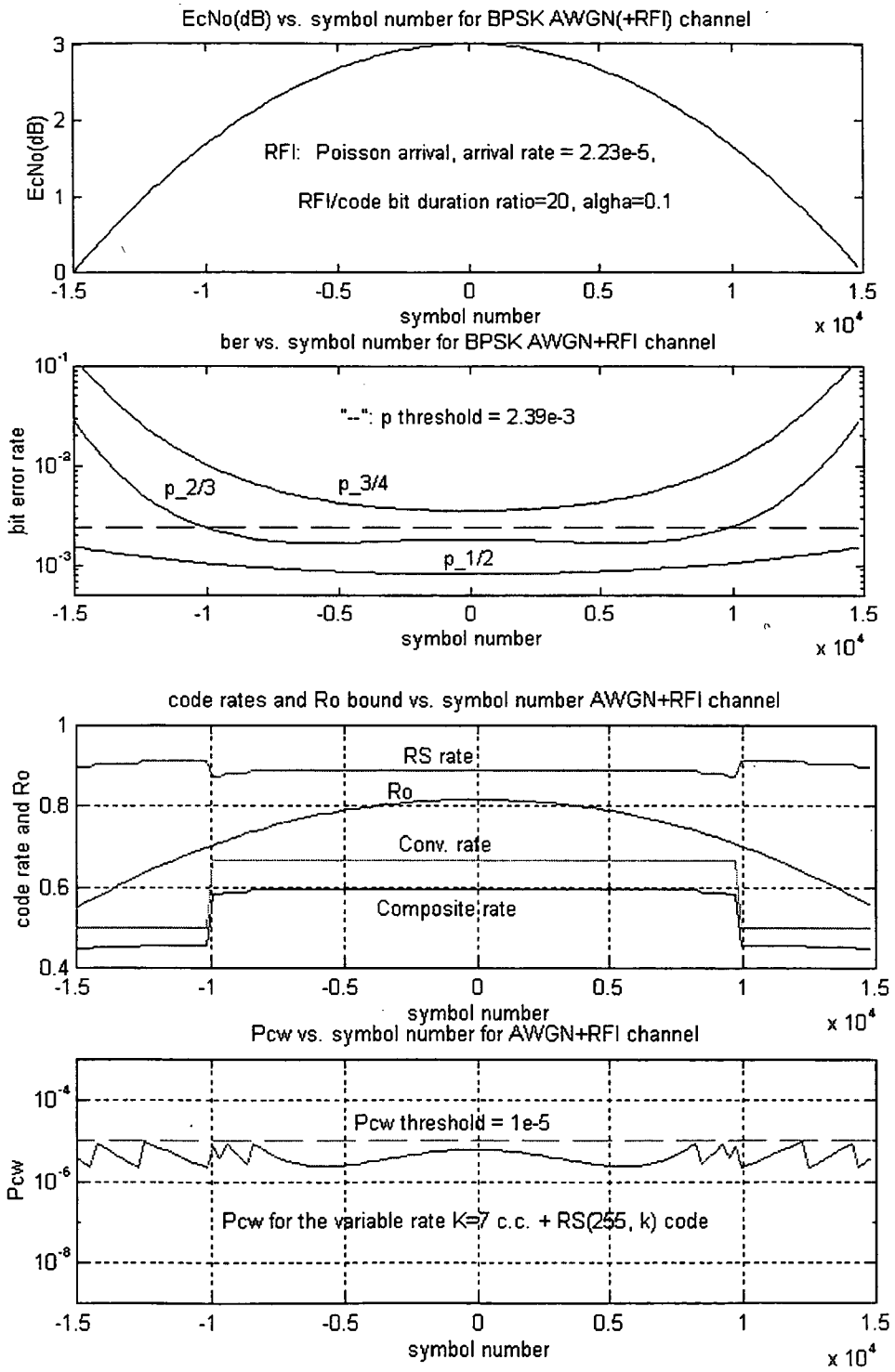


**Fig. 4.8** Results for Case 7

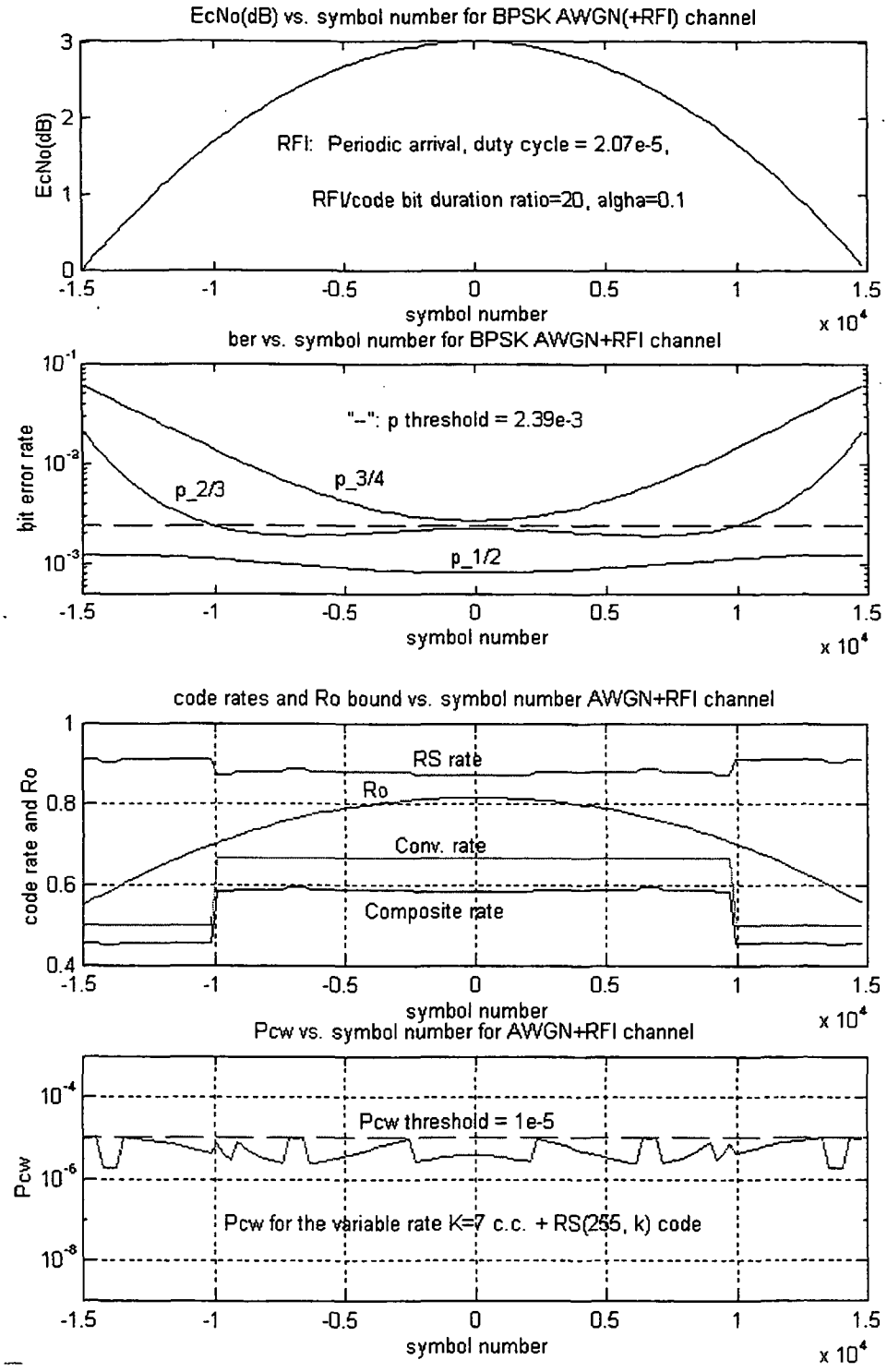


**Fig. 4.9** Results for Case 8

Table 4.2 summarizes all the results discussed above and together with the coded daily throughputs.

**Table 4.2** Coded Throughput Efficiencies and Coded Daily Throughputs

| **RFI Scenarios** | **Coded Efficiencies** | **Daily Coded Throughputs (Daily Theoretical Throughput = 1.5 Gbits)** |
|---|---|---|
| Case 1 | 0.8668 | 1.3 Gbits |
| Case 2 | 0.8613 | 1.29 Gbits |
| Case 3 | 0.7474 | 1.12 Gbits |
| Case 4 | 0.7748 | 1.16 Gbits |
| Case 5 | 0.7364 | 1.1 Gbits |
| Case 6 | 0.7434 | 1.12 Gbits |
| Case 7 | 0.7484 | 1.12 Gbits |
| Case 8 | 0.7445 | 1.12 Gbits |

# Chapter 5

# SUMMARY AND CONCLUSIONS

The design of a reliable satellite communication link involving the data transfer from a small, low-orbit satellite to a ground station, through a geostationary satellite, was examined. In such a scenario, the received signal power to noise density ratio increases as the transmitting low-orbit satellite comes into view, and then decreases as it departs, resulting in a short-duration, time-varying communication link. The optimal values of the small satellite antenna beamwidth, signaling rate, modulation scheme and the theoretical link throughput (in bits per day) from the first sub-project were taken to check the practical coding schemes, both fixed rate and variable rate concatenated FEC coding schemes for the AWGN channel. The effect of RFI on the best coding scheme, which maximized the daily throughput while satisfying a prescribed probability of error requirement, among above schemes was examined.

In Chapter 2, the discrete-time equivalent AWGN-plus-RFI model for simulation was described. In Chapter 3, the throughput efficiencies for the AWGN channel were numerically estimated. Three coding schemes, one fixed rate and two variable rate schemes were examined. It was found that the Code 3, which is the variable rate for both the RS codes and convolutional codes, could achieve 87 percent of the

theoretical throughput, equivalent to 1.3 Gbits/day based on the cutoff rate $R_0$. Then in Chapter 4, the effect of RFI on Code 3 was examined. Eight different RFI scenarios, including both periodic arrival RFI and Poisson arrival RFI, RFI/code bit duration ratio value from 0.1 to 20 were considered. Interleaving technique was used to mitigate degradation due to RFI. It was found that the Code 3 could still achieve 74 percent of the theoretical throughput, equivalent to 1.11 Gbits/day based on the cutoff rate $R_0$, compared to 87 percent for the AWGN-only case.

The final conclusion is easy to be made based on the above results. Code 3 is recommened for this time-varying satellite channel. Actually, the chip set for variable rate convolutional codes are available in the market.

# REFERENCES


[1] W. Ryan and L. Han, " Modulation and Coding for Short-Duration Deterministically Time-Varying Channels," *1995 International Conference on Communications*, Seattle, WA.

[2] P. Quintana, *Optimal Parameters for a Low Orbit to Geostationary Satellite Link,* Technical Report, New Mexico State University, Las Cruces, 1994.

[3] G.C. Clark, J.B. Cain, *Error-correction Coding for Digital Communications,* Plenum Press, New York, 1981.

[4] S. Wilson, *Digital Modulation and Coding,* Prentice-Hall, 1995.

[5] B. Kopp, *An Analysis of Carrier Jitter in an MPSK Receiver Utilizing Map Estimation,* Ph.D. Dissertation, New Maxico State University, Las Cruces, 1994.

[6] S. Lin and D. Costello, *Error Control Coding: Fundamentals and Applications,* Prentice-Hall, 1983.

[7] B. Sklar, *Digital Communications: Fundamentals and Applications,* Prentice Hall, 1988.

[8] M. K. Ochi, *Applied Probability and Stochastic Processes in Engineering and Physical Sciences,* A Wiley-Interscience Publication, 1990.

[9] L. Kleinrock, *Queueing Systems, Volume 1: Theory,* Wiley-Interscience Publication, 1975.

[10] A. Weinberg, " On the Passage of High-Level Pulsed Radio Frequency Interfernce Through a Nonlinear Satellite Transponder, " *IEEE Trans. Commun.,* vol. COM-32, pp.13-24, January 1984.

[11] A. Weinberg, "The Impact of Pulsed RFI on the Coded BER Performance of the Nonlinear Satellite Communication Channel," *IEEE Trans. Commun.,* vol. COM- 29, pp.605-620, May 1981.

[12] J. L. Ramsey, " Realization of Optimum Interleavers , " *IEEE Trans. Info. Theo.*, vol. IT-16, pp.338-345, May 1970.

[13] D. Haccoun and G. Begin, "High-Rate Punctured Convolutional Codes for Viterbi and Sequential Decoding," *IEEE Trans. Commun.* vol. 37, pp 1113-1125, November 1989.

# APPENDIX A

# PUNCTURED CONVOLUTIONAL CODES

In order to transfer more information data in the higher SNR region, higher rate (>1/2) convolutional codes were used in this thesis. But a straightforward application of Viterbi decoding to high-rate codes becomes very rapidly impractical as the coding rate increases. Fortunately, the high-rate codes can be obtained by the periodic elimination (i.e., puncturing) of specific code symbols from the output of a low-rate encoder [13]. The high-rate punctured code depends on both the low-rate code, called the original code, and on both the number and specific positions of the punctured symbols. The pattern of punctured symbols, described in a matrix form, is called the perforation pattern of the punctured code.

In this thesis, the original code is the rate-1/2, constraint length 7 convolutional code, and its punctured higher rates (i.e., rate 2/3, 3/4, 4/5) codes have been used. The perforation pattern matrixes for these codes are given [13] as follows, where 0 indicates the punctured bits,

$$M_{2/3} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$$

$$M_{3/4} = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

$$M_{4/5} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Fig. A.1 and Fig. A.2 show an example of how to obtain a punctured rate-2/3 code with the perforation pattern matrix of $M_{2/3}$ from its original rate-1/2, 4-state encoder. (The constraint length 7 code has too many states, 64, to demonstrate this principle).

Fig. A.1 gives the trellis diagram of a simple four states, rate-1/2 convolutional code where every fourth symbol is punctured (indicated by X). By reading this trellis two branches at a time and redrawing it as in Fig. A.2, we see that it corresponds to a

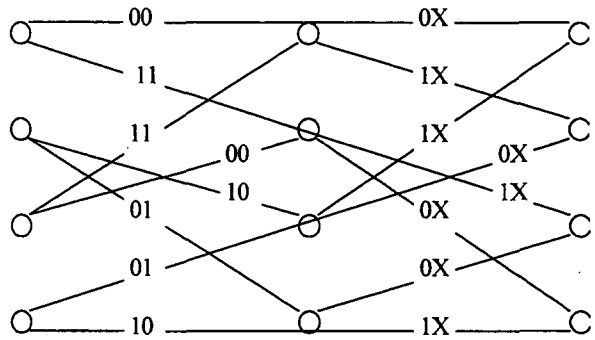


**Fig. A.1** Trellis for Four States, Rate-1/2 Convolutional Code

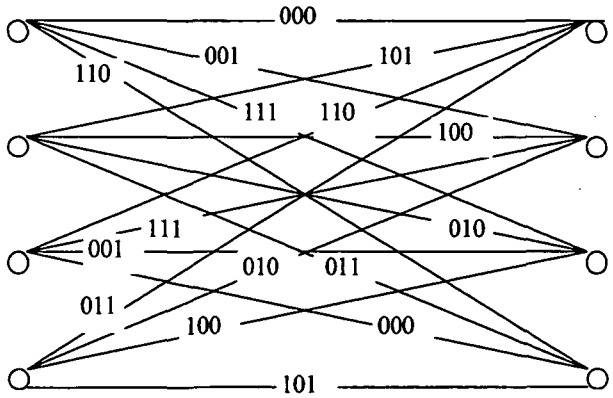


**Fig. A.2** Trellis for Punctured Rate-2/3 Convolutional Code

4-state, punctured rate-2/3 code.

The punctured code can be decoded by adding relatively simple circuitry to the codecs of the original low-rate 1/2 code. The added circuitry is used to insert dummy data (at the decoder) into the positions corresponding to the deleted code symbol based on their perforation pattern matrixes, where these dummy data are discarded by assigning them the same metric value (usually zero) regardless of the code symbol in the decoding process. The advantage of this decoding method is that Viterbi codecs for high-rate punctured codes is that the same rate 1/2 decoder may be used to decode the entire family of codes.

# APPENDIX B

# MATLAB AND C PROGRAMS LIST

In this section, we describe all the programs used in this thesis and then give a sample C simulation program for the AWGN-plus-RFI channel at the end. All the programs listed here can be found from a disk attached at the back of this thesis.

## B.1 Programs List

*Matlab Calculation Programs*

t_cv_rs3.m This program computes the throughput efficiency for a fixed rate 1/2, constraint length 7 convolutional code concatenated with a (255, 223) RS code (i.e., Code 1) on the AWGN-only time-varying channel.

t_vr_cc3.m This program computes the throughput efficiency for the variable rate (1/2, 2/3, 3/4, 4/5), constraint length 7 convolutional codes concatenated with a (255, 223) RS code (i.e., Code 2) on the AWGN-only time-varying channel.

vcr3.m This program computes the throughput efficiency for the variable rate (1/2, 2/3, 3/4, 4/5), constraint length 7 convolutional codes concatenated with a variable rate (255, $k$) RS code (i.e., Code 3) on the AWGN-only time-varying channel.

vcr3_RFI.m This program computes the throughput efficiency for the Code 3 on the AWGN-plus-RFI time-varying channel. This program need to load the probability of error at the output of Viterbi decoder (i.e, $p$) data file generated by pb_vit.m.

pb_vit.m This program produces $p$ vs. symbol number curves by interpolating these measured points, where the $p$ data are generated by the C simulation programs for the variable rate convolutional codes on the AWGN-plus- RFI channel.

binom.m This fuction program calculates the binomial coefficients, "n choose k". It is used in the first four programs discussed above.

mpskroli.m This fuction program calculates the cutoff rate $R_0$ in units of information bits/channel use. The format of this function is :

mpskr0(m, EsNodB). It is used in the first four programs above.

pcw.m This function returns the probability of codeword error for a (n, k) RS code over GF(q) when the channel symbol error probability is p. The format of this function is: Pcw(n, k, p). It is also used in the first four programs discussed above.

*C Simulation Programs*

RFI1.cpp This program simulates the probability of bit error for a rate 1/2, constraint length 7, soft decision viterbi decoder in a RFI environment. The Viterbi decoder uses the best metric with 32-bit memory, and the interleaving is not emploied. The output of this program will be written to a file, which will be then used in vcr3_rfi.m.

RFI2.cpp This program simulates the probability of bit error for a rate 2/3, constraint length 7, soft decision punctured viterbi decoder in a RFI environment. The Viterbi decoder will take two stages from trellis at a time, so that the decoding memory is 64-bit. The interleaving is not invoved. The output of this program will be written to a file too.

RFI3.cpp This program simulates the probability of bit error for a rate 3/4, constraint length 7, soft decision punctured viterbi decoder in a RFI environment. The Viterbi decoder will take two stages from trellis at a time, so that the decoding memory is 64-bit. The interleaving is not invoved. The output of this program will be written to a file too.

RFI4.cpp This program simulates the probability of bit error for a rate 4/5, constraint length 7, soft decision punctured viterbi decoder in a RFI environment. The Viterbi decoder will take two stages from trellis at a

time, so that the decoding memory is 64-bit. The interleaving is not invoved. The output of this program will be written to a file too.

RFI1_int.cpp This program simulates the probability of bit error for a rate 1/2, constraint length 7, soft decision viterbi decoder in a RFI environment. The Viterbi decoder uses the best metric with 32-bit memory, and the interleaving is emplyed. The output of this program will be written to a file, which will be then used in vcr3_rfi.m.

RFI2_int.cpp This program simulates the probability of bit error for a rate 2/3, constraint length 7, soft decision punctured viterbi decoder in a RFI environment. The Viterbi decoder will take two stages from trellis at a time, so that the decoding memory is 64-bit. The interleaving is invoved. The output of this program will be written to a file too.

RFI3_int.cpp This program simulates the probability of bit error for a rate 3/4, constraint length 7, soft decision punctured viterbi decoder in a RFI environment. The Viterbi decoder will take two stages from trellis at a time, so that the decoding memory is 64-bit. The interleaving is invoved. The output of this program will be written to a file too.

### B.2 Sample C Simulation Program

Eor the AWGN-plus-RFI channel, Three different cases were considered during the programming. The first case is the Viterbi decoding without interleaving (RFI

affect less than one code bit) for both rate 1/2, constraint 7 convolutional code and its high-rate punctured codes, the second case is the Viterbi decoding with interleaving (RFI affect more than one channel code bit) for the original rate 1/2 convolutional code, and the last case, and also the most complicated one, is the Viterbi decoding with interleaving for the high-rate punctured codes. Because the first two cases can be considered as the sub-programs of the last case, we just give a sample, Viterbi decoding with interleaving for 3/4 punctured code, C simulation program here, and from this program, the reader may enable to either figure out or easily understand that of the first two cases.

*Sample C Simulation Program*

```
/*********************************************************/
/*  Simulate a rate 3/4 soft decision punctured          */
/*  viterbi decoder reference to [13].                   */
/*  The definitions of parameters used in this           */
/*  program are either listed below or followed by       */
/*  the parameter at the first time being useed.         */
/*                                                       */
/*  state_bk -- shift register of convolutional encoder, */
/*  Ck -------- coded bits,                              */
/* _Rk -------- received bits,                           */
/*  biterrs --- bit error counter,                       */
/*  err_max --- maximum number of error to stop          */
```

```
/*              the simulation,                                  */
/*  opt_n ----- the index of the highest metrix,                 */
/*  opt_met --- optimal (highest) matrix of trells               */
/*              decision equal to 1 when decoder                 */
/*              making error, and 0 otherwise,                   */
/*  Index_DL -- index of delay lines for both                    */
/*              interleaver and de-interleaver,                  */
/*  pattern --- perforation pattern matrix of rate               */
/*              2/3 punctured code,                              */
/*  data_up --- uper 32-bit of information bit stream,           */
/*  data_low -- lower 32-bit of information bit stream,          */
/*  path_up --- uper 32-bit surviving pathes of trellis,        */
/*  path_low -- lower 32-bit surviving pathes,                   */
/*  dt_m ------ extra delay buffer due to interleaver            */
/*              for information bits,                            */
/*  nbits ----- number of information bits have been             */
/*              sent through,                                    */
/*  Fi -------- random phase of RFI,                             */
/*  Tc -------- code bit duration,                               */
/*  beta ------ RFI/Tc ratio,                                    */
/*  delta_W --- deferential frequency of RFI,                    */
/*  Pcc ------- probability of bit error of Viterbi              */
/*              decoder                                          */
/*  mean_cw --- mean of Poisson random variable,                 */
/*  k_cw ------ RFI interarrival time counter                    */
/*                                                               */
/*****************************************************************/
```

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "c:\temp\data.h"  //Contains Ec(s)N0_dB data and 7-bit
                             parity lookup table

#define  POLYA    0133      //First generator polynomal of rate
                             1/2, constraint 7 convolutional
                             encoder
#define  POLYB    0171      //Second generator polynomal
#define  DELAY    64        //Decoding length of Viterbi
#define  TWOPI    6.283185307
#define  M_INF    -1.0e8
#define  D_LINE   64        //Number of delay lines for a
                             convolutional interleaver

long randws(void);          //Random number generator
double expo(double);        //Interarrival time generator
                             for Poisson arrival RFI
void interleaver(unsigned, float *, unsigned);
void awgn(float *, float, float *);   //AWGN generator
void deinterleaver(unsigned, float *, unsigned);
void viterbi(float *, float *, unsigned long *, unsigned long
*);  // Viterbi decoder

float T_int[D_LINE][D_LINE], T_de[D_LINE][D_LINE];
     //Delay lines tables for interleaver and de-interleaver
struct table {
   unsigned ps[128];
   unsigned long in[64];
```

```
   int  out1[128];
   int  out2[128];
} trellis;

main()
{
 unsigned state_bk, n, i, j, k, err_max, opt_n;
 unsigned pattern[6]={1,1,1,0,0,1}, decision, Index_DL;
 unsigned long data_up, data_low, path_up[64], nbits, bk,
              path_low[64], dt_m[95];
 float EbNo_dB, EbNo, EcNo, Ec, No, stddev, m, Pcc[120],
       metric[64], biterrs, opt_met, Tau, Fi, Alfa, beta, Tc,
       low, up, delta_W, Ck[2], Rk[2];
 double k_cw, mean_cw;
 FILE *f_w;

/* generate trellis lookup table */
 for (n=0; n<64; n++) {      //Current state
    trellis.in[n]  = n >> 5;      //Input bit
    for (i=0; i<2; i++ ) {  //Last bit of previous state
       trellis.ps[2*n+i] = ((n&31) << 1) | i; //Previous state
       trellis.out1[2*n+i] = Partab[((n<<1) | I) & POLYA] ?
                                  1 : -1;  //First output symnol
       trellis.out2[2*n+i] = Partab[((n<<1) | i) & POLYB] ?
                                  1 : -1; //Second output symbol
    }
 }

 Tc = 1./4.5e6;  //Because the data rate = 4.5e6 cbps
 Alfa = .1;       // Set Alfa = 0.1
```

```
 delta_W = Alfa*TWOPI/Tc; // Delta omiga of CW RFI
 mean_cw = 23567.;      k_cw = expo(1./mean_cw);
 Tau = 10*Tc;    beta = Tau/Tc;    err_max = 80;

 printf("\n\n Program is running, please wait.....
        \n \nBit Error Rate For Parameter Group A :\n\n");

 for (j=0; j<11; j++) { //11 sample Ec/N0 values in region
                          [-15,000, 0]
/* initialize some parameters */
    state_bk = biterrs = nbits = 0;
    Index_DL = data_up = data_low = 0;
    for (i=0; i<95; i++) dt_m[i] = 0;
                //Need 2*63*3/4 32-bit input data buffer
    for (i=0; i<D_LINE; i++) {
       for (m=0; m<D_LINE; m++) {  //Preset 1
           T_int[i][m] = T_de[i][m] =  1;
       } //For m loop
    }    //For i loop
    m = 0;
/* calculate noise level */
    EcNo = pow(10., EcNo_dB[j]/10.);
    Ec = 1.;
    No = Ec/EcNo;          //AWGN level
    stddev = sqrt(No/2.); //Standard deviation of AWGN

/* initialize metrics*/
    for (n=0; n<64; n++)          metric[n] = M_INF;
    metric[0] = M_INF/100;
```

```
/* calculate BER by doing simulation */
      while(biterrs < err_max)  {

/* Generate random data */
         for (k=0; k<3; k++) {
            nbits ++;     //info. bits counter
            bk = (rand() >> 11) & 1; //Random input(0, 1)
            data_low = (data_low >> 1) | ((data_up & 1) <<
                       // Update lower 32 input bits
31);
            data_up = (data_up >> 1) | ((dt_m[94] & 1) << 31);
                       // Update uper 32 input bits
            for (i=94; i>0; i--) dt_m[i] = (dt_m[i] >> 1) |
((dt_m[i-1] & 1) << 31);
            dt_m[0] = (dt_m[0] >> 1) | (bk << 15);
                       // 64*63*3/4 info-bit delay

/* K=7 rate=1/2 convolutional encoder (133,171) */
            state_bk = (state_bk >> 1) | (bk << 6);
            Ck[0] = Partab[state_bk & POLYA] ? 1. : -1.;
                       // First codebit
            Ck[1] = Partab[state_bk & POLYB] ? 1. : -1.;
                       // Second codebit

/* interleaving the coded bits */
            interleaver(Index_DL, Ck, k);

/* add AWGN to the encoded data */
            awgn(Ck, stddev, Rk);

/* add RFI to the encoded data */
```

```
            if ((k_cw<2) && (k_cw>=1) && (k!=1)) {
                        //The 1st RFI hit Rk[1]
               Fi = TWOPI*(randws()+1)/((float)0x100001);
               low = k_cw*Tc;    //Lower limit of integrator
               if ((k_cw+beta) < (m+1))  {
                  up = (k_cw+beta)*Tc;  //Upper limit
                  Rk[1] += 10/(Tc*delta_W))*(sin(delta_W*up+Fi)
- sin(delta_W*low+Fi));
                  k_cw = expo(1./mean_cw) + up - m*Tc;
                  m = 0; // Reset m=0, and exit the loop
               }  // For if ((k_cw<2) && (k_cw>=1) && (k!=1))
               else {
                  up = (m+1)*Tc;
                  Rk[1] += 10/(Tc*delta_W))*(sin(delta_W*up+Fi)
- sin(delta_W*low+Fi));
                  m++;
               }
               k_cw--;
            }
            else if (k_cw < 1) {
                //The 1st RFI hit Rk[0] or continue from above
               if (m == 0)
                  Fi = TWOPI*(randws()+1)/((float)0x100001);
               low = (k_cw > m) ? k_cw*Tc : m*Tc;
               if (((k_cw+beta)<(m+1)) && (k!=2))  {
                  up =  (k_cw+beta)*Tc;
                  Rk[0] += 10/(Tc*delta_W))*(sin(delta_W*up+Fi)
- sin(delta_W*low+Fi));
                  k_cw = expo(1./mean_cw) + up - m*Tc;
                  m = 0;
```

```
            }
            else if (k!=2)  {
               up = (m+1)*Tc;
               Rk[0]  +=  (10/(Tc*delta_W))*(sin(delta_W*up+Fi)
- sin(delta_W*low+Fi));
               m++;
               low = (k_cw > m) ? k_cw*Tc : m*Tc;
               if (((k_cw+beta)<(m+1)) && (k!=1))  {
                  up =  (k_cw+beta)*Tc;
                  Rk[1] += 10/(Tc*delta_W))*(sin(delta_W*up+Fi)
- sin(delta_W*low+Fi));
                  k_cw = expo(1./mean_cw) + up - m*Tc;
                  m = 0;
               }
               else if (k!=1) {
                  up = (m+1)*Tc;
                  Rk[1] += 10/(Tc*delta_W))*(sin(delta_W*up+Fi)
- sin(delta_W*low+Fi));
                  m++;
               }
            }
         }
         else k_cw -= 2;

/* deinterleaving the received bits */
         deinterleaver(Index_DL, Rk, k);
         Index_DL = k ? (Index_DL+1)%D_LINE :
(Index_DL+2)%D_LINE ;
/* viterbi decoder */
         if (nbits > D_LINE*(D_LINE-1)*3/4)  {
```

```
            Rk[0] *= pattern[2*k]; // Insert dummy data
            Rk[1] *= pattern[2*k+1];
            viterbi(Rk, metric, path_up, path_low);
         }

/* check errors */
         if (nbits >= int(DELAY+D_LINE*(D_LINE-1))) {
            opt_met = M_INF;
            for (n=0; n<64; n++) {
               if (metric[n] > opt_met) {
                  opt_met = metric[n];
                  opt_n = n;
               }
            }
            decision = (path_low[opt_n] ^ data_low) & 1;
            biterrs += decision;
         }
      }
   }
   Pcc[j] = biterrs/(nbits-int(DELAY+D_LINE*(D_LINE-1)*3/4));
   printf("%10.3e,  ", Pcc[j]);

/* open a file for write */
   if((f_w = fopen("p3_c4.dat ","a")) == NULL)  {
      printf("\n The p3_c4.dat file cannot be opened\n");
      return 0;
   }
   fprintf(f_w, "%10.3e,  ", Pcc[j]);
/* close the file  */
   fclose(f_w);
```

```
 }
 printf("\n\n\nThe End!");
 return 0;
}

/* Random # generator described by Widrow and Stearns */
long randws()
{
   static long seed = 78971;  //seed to arbitrary value
   return( seed = (2045*seed+1) & 0xfffff );
}


/* Exponential random variable generator described by ee572
text book */
double expo(double a) //generate the interarrival time of
Poisson arrival RFI
{
   return(-log((randws()+1)/((float)0x100001))/a);
}

/* AWGN generator */
void awgn(float Ck[], float stddev, float Rk[])
{
   float  u1, u2, a, b;
   u1 = (float)(randws()+1)/((float)0x100001);
   u2 = (float)(randws()+1)/((float)0x100001);
   a=TWOPI*u1;
   b=sqrt( -2.0*log(u2) );
   Rk[0] = Ck[0] + stddev*b*cos(a);
```

```
   Rk[1] = Ck[1] + stddev*b*sin(a);
}

/* Viterbi decoder */
void viterbi(float Rk[], float metric[], unsigned long
path_up[], unsigned long path_low[])
{
   unsigned n, i;
   unsigned long path_up_new[64], path_low_new[64];
   float m[64][2];
   for (n=0; n<64; n++) {              // Current state
      for (i=0; i<2; i++)      // Input bit for previous state
         m[n][i] = metric[trellis.ps[2*n+i]] +
Rk[0]*trellis.out1[2*n+i] + Rk[1]*trellis.out2[2*n+i];
         if (m[n][0] > m[n][1]) {
            path_up_new[n] = (path_up[trellis.ps[2*n]] >> 1) |
(trellis.in[n] << 31);
            path_low_new[n] = (path_low[trellis.ps[2*n]] >> 1)
| ((path_up[trellis.ps[2*n]] & 1) << 31);
         }
         else   {
            path_up_new[n] = (path_up[trellis.ps[2*n+1]] >> 1)
| (trellis.in[n] << 31);
            path_low_new[n] = (path_low[trellis.ps[2*n+1]] >>
1) | ((path_up[trellis.ps[2*n+1]] & 1) << 31);
         }
      }
      for (n=0; n<64; n++) {
         metric[n] = (m[n][0] > m[n][1]) ? m[n][0] : m[n][1];
                              // Metric update
```

```
            path_up[n] = path_up_new[n];    // Path_up update
            path_low[n] = path_low_new[n];  // Path_low update
         }
      }
   }

/* interleaver */
void interleaver(unsigned Index, float Ck[], unsigned k)
{
   unsigned n;
   float temp;
   if (k != 2) {
      temp = Ck[0];       // update CK[0]
      if (Index > 0) {    // update delay line of the table
         for (n=Index; n>0; n--)
            T_int[Index][n] = T_int[Index][n-1];
      }
      T_int[Index][0] = temp;
      Ck[0] = T_int[Index][Index];
   }
   if (k != 1) {
      if (k==0) Index = (Index+1)%D_LINE;
                        // (index+1) maybe back to 0
      temp = Ck[1];  // update CK[1]
         if (Index > 0) {
            for (n=Index; n>0; n--)
               T_int[Index][n] = T_int[Index][n-1];
         }
         T_int[Index][0] = temp;
         Ck[1] = T_int[Index][Index];
```

```
    }
}

/* deinterleaver */
void deinterleaver(unsigned Index, float Rk[], unsigned k)
{
   unsigned n;
   float temp;
   if (k != 2) {
      temp = Rk[0];                    // update RK[0]
      if ( (D_LINE-Index-1) > 0 ) {
         for (n=(D_LINE-Index-1); n>0; n--)
            T_de[Index][n] = T_de[Index][n-1];
      }
      T_de[Index][0] = temp;
      Rk[0] = T_de[Index][D_LINE-Index-1];
   }
   if (k != 1) {
      if (k==0) Index = (Index+1)%D_LINE;
                     // (index+1) maybe back to 0
      temp = Rk[1]; // update RK[1]
      if ( (D_LINE-Index-1) > 0 ) {
         for (n=(D_LINE-Index-1); n>0; n--)
            T_de[Index][n] = T_de[Index][n-1];
      }
      T_de[Index][0] = temp;
      Rk[1] = T_de[Index][D_LINE-Index-1];
    }
}

/**************        End of Program        ****************/
```