A Variable-Rate Modulation and Coding Scheme For Low Earth Orbit Satellites

by

Kevin Butchart

Submitted to the Department of Electrical Engineering
in partial fulfilment of the requirements for the degree of

Master of Science in Engineering

at the

UNIVERSITY OF CAPE TOWN

December 1998

Acknowledgements

I would like to thank my supervisor, Dr. Robin Braun for the help and guidance he gave me.

I would also like to thank Gordon Farquharson for always having time to help work through problems I encountered.

Thanks also go to Dan Saban for often giving a new perspective and to Dr Daniel Mashao for the encouragement he gave.

Finally I would like to thank my family for the encouragement and support they have given me.

Synopsis

Low Earth Orbit (LEO) satellites are increasingly being used for a wide variety of communications applications.

These satellites have to operate in widely varying channel conditions. These conditions are often significantly better than the 'worst case' situations that are experienced and thus a single rate transmission scheme is clearly suboptimal.

The objective of the thesis is to suggest and test a method of modulation/coding that can take advantage of better signal strength conditions in order to improve data transmission rates.

In order to provide the goal of approximately 50kbps transmission in a 10 kHz Frequency Division Multiple Access (FDMA) channel it was necessary to consider spectrally efficient, rather than power efficient, modulations.

The proposed modulation scheme makes use of an eight-dimensional trellis coded modulation system. Multiple signal constellation sets are used in conjunction with this coding in order to provide different transmission rates, depending on the signal to noise ratio and the channel state.

To enhance the suitability of the modulation scheme for the channel, it was combined with Reed-Solomon Coding and interleaving in an inner/outer code arrangement.

Various means of determining when to switch between coding rates were discussed briefly, but an in-depth treatment of the subject fell outside of the scope of the thesis.

Various combinations of these codes were tested in gaussian noise conditions and various degrees of Rician and Rayleigh fading. In order to make use of the higher rate QAM constellations, it was necessary to provide the decoder with channel state information.

The tested system achieved its purpose of providing a variable rate coding scheme resulting in

good performance over a range of channel conditions. It is fairly flexible and can be adapted to specific channel requirements.

Contents

List of Figures

Chapter 1

Introduction

Satellites in a low earth orbit (LEO) provide a number of advantages over Geo-stationary satellites. These include the smaller amplifier power requirements, less time delay and lower cost of development and launching.

The low-earth orbit presents a number of difficulties, however. The constantly varying position of the satellite leads to a number of problems, including large doppler frequency shifts and varying signal power. A single LEO satellite is also only visible for short periods of time each orbit. This has lead to many proposals for large 'constellations' of these satellites for communications systems.

In particular, the path loss from the satellite to the receiver is substantially less when the satellite is directly overhead than when the satellite is low on the horizon. This, combined with the greater fading and shadowing experienced when the satellite is close to the horizon, creates a channel environment with significant variation in signal power.

It is proposed that a variable-rate coding/modulation scheme could make use of the extra signal strength, when it is available, to provide higher transmission rates. Such a scheme should provide reliable transmission in poor signal conditions at lower data rates.

1.1 Objectives

The objectives of this thesis are as follows:

- To propose such a variable-rate modulation and coding scheme that has the above described characteristics. This system should be able to adapt to slower rates when the signal strength is weaker.
- To simulate the proposed system performance in typical LEO satellite channel conditions.

The proposed system is designed so that it may be included on a future LEO satellite project.

1.2 Scope of Thesis

The following aspects were covered in this thesis:

- Satellite Channel: The modelling of the satellite channel, its characterisation as a Rayleigh/Rician Fading channel and the variation in received signal power.
- Modulation Scheme: The choice and design of a modulation scheme and the associated coding that will allow different signal rates to be used for different signal conditions.
- Adaptive Rate Code: The factors that affect the decision of changing rates and the switching of the transmission rate accordingly, is discussed.

The scope was limited in the following areas:

- The system was only tested in a single channel environment. The effects of adjacent channel interference resulting from use in a FDMA channel were not assessed.
- As the aim of the thesis is to provide a system that could provide high data rates, spectrally efficient rather than power efficient modulation schemes were considered. This led to the consideration of M-PSK and QAM modulation schemes as opposed to FSK/MSK schemes.
- Only the factors that could be used to change rate were discussed. The actual design of a decision system to switch the data rate when appropriate fell outside the scope of the thesis.

1.3 Structure of Report

Chapter 2 Background and Literature Review.

Chapter 3 Satellite Channel. The sources of degradation in the satellite channel are discussed. The variable signal power caused by the movement of the satellite is considered and a brief development of the Rician/Rayleigh fading channel model that was used is given.

Chapter 4 Modulation Scheme. The process of selection of a modulation scheme is discussed. The proposed 8-dimensional trellis coded modulation scheme is described in detail, and its use together with different signal constellations, is discussed.

Chapter 5 Shows the complete system and discusses the various configurations possible. A few configurations are chosen for the purpose of testing. The simulated system is shown and the various parameters used for the simulation are discussed.

Chapter 6 Adaptive Code Rate. Discusses the factors to be considered when switching code rates and possible network configurations. A simple control system is also briefly described.

Chapter 7 Results and Discussion. The results of the simulations are presented and these results are analysed.

Chapter 8 Conclusions.

Appendix A Diffuse Signal Spectrum. The model of the diffuse signal power, as used in the Rician fading model, is briefly discussed.

Appendix B Matlab and C++ Code used for simulations.

Chapter 2

Background

Although the first satellites deployed were situated in "low earth orbits (LEO)", the Geo-synchrynous, or Clarke, orbit has been more widely used for commercial communications. The stationary position with respect to any point on the ground, and the large footprint of the satellite, offers many advantages over satellites that are placed in other orbits.

Over the last 5-10 years, however, there has been increased interest in LEO satellites. These satellites have the following advantages over the higher Geo-stationary satellites

- Cost. The cost of development and deployment of these satellites is much lower than Geo-stationary satellites.
- Power. The power loss in transmission is relatively low, making mobile communication feasible.
- Tranmission delay. The tranmission delay between the ground station and the transmitter is much lower than that of geo-stationary satellites

Disadvantages include the rapidly changing signal conditions caused by the movement of the satellite, the relatively small footprint of the satellite and the short time duration during which communications can take place for each pass.

This chapter provides a brief overview of research that has been conducted into the use and optimisation of LEO satellite links. Research into different modulation and coding schemes relevant to this study, is also discussed.

2.1 LEO Satellites

Much research of late has gone into optimising the various aspects of the LEO Satellite Link.

Due to the limited availability of an individual LEO satellite, much research is being done into satellite constellations. These constellations consist of a number of LEO satellites that can communicate with each other in order to provide high availability.

There are several "Big LEO" projects that are currently being undertaken by large companies and consortiums. These include the Iridium, Globalstar and Teledesic project.

These are ambitious, large budget, schemes that use between 48 (GlobalStar) and more that 800 (Teledesic) satellites to achieve broad earth coverage. These projects aim to provide large scale real time communication services.

Various smaller LEO satellite constellations have been proposed in [1] and [2]. These "Little LEO" constellations are designed to provide near real-time data transmission at significantly lower cost. These satellite constellations make use of fewer low cost satellites in order to provide services such as regional communications and disaster prediction.

The design of network configurations of these satellites has received a large amount of research. The design and analysis of these systems has been addressed by, amongst others Vatalaro et. al. [3], Glisic et. al [4] and Wood [5],

2.2 Channel Model

The mobile satellite channel has been characterised as a AWGN noise channel that experiences various degrees of short-term fading. This fading has been typically characterised as having a Rician/Rayleigh distribution [6].

For simulation purposes, the Rician distribution may be determined statistically, such as the Clarke model for flat fading. Another approach that is popular especially for mobile channels, is to record the signal fading experienced in a real communications environment and store it for later simulation.

Statistical measurement and analysis of Land-Mobile Satellite channels has been investigated by, amongst others, Vogel [7] and Loo [8]. These studies have considered the effects of

different types of terrain and different elevation angles for mobile satellite links.

2.3 Modulation and Coding

Trellis Coded Modulation

The idea of combining coding and modulation was first formally suggested in 1974 by Massey [9]. Trellis-coded modulation (TCM), described in 1982 by Ungerboeck [10] is perhaps the most significant contribution to this field.

TCM has the advantage of providing better power efficiency, without the bandwidth expansion usually demanded by the coding process.

2.3.1 Multidimensional Trellis Codes

Motivation for the use of multidimensional modulations can be traced back to the work of Shannon. In his analysis of the limitations on performance over a given channel [11], he recognised that performance could be improved by increasing the dimensionality of the signal set. This improvement tends towards the channel capacity as the dimensionality tends towards infinity.

The main benefit of using higher dimensional signal sets with trellis coded modulation is the reduced constellation expansion needed for each signal interval. Another benefit is that well-constructed signal sets are easily made rotationally invariant.

Uncoded four dimensional signal sets have been analysed by Welti and Lee [12], Wilson et al. [13] and Biglieri and Elia [14].

Higher dimensional signal sets have also been considered. Wei [15] has developed a construction that creates a 2N-dimensional alphabet over N adjacent signal intervals. Others have developed higher dimensional signals based on multidimensional lattices such as the Gosset eight-dimensional lattice and the Leech 24-dimensional lattice [16].

Wei construction, in particular, is a particularly practical design as it solves problems such as phase invariance for QAM constellations. It has been used in this thesis and by a number of other authors such as Tretter [17] and Su [18].

Trellis codes are typically designed for use on AWGN channels. This is not optimal for fading channels and modifications to the convolutional code and set design process have been proposed. These are covered in [19] and [20]

2.3.2 Turbo Codes

Turbo codes were introduced in 1993 by Berrou et al.[21]. These codes have received considerable attention due to their very good bit error performance in low SNR conditions.

Binary Turbo Coding makes use of a parallel concatenation of two binary recursive convolutional coders and an interleaver. This has been extended to Trellis Coded Modulation by Robertson and Wörz [22] who made use of two parallel trellis encoders in a scheme called Turbo Trellis Coded Modulation(TTCM).

2.4 Reed-Solomon Code

Reed-Solomon Coding was first introduced in the 1960's by Reed and Solomon [23]. They can be considered as a special case of non-binary BCH cyclic block codes.

Due to the large amount of computation required, they were not prominent until advances in VLSI made the coding/decoding feasible. Reed-Solomon codes are widely used as block error correction codes.

Concatenated coding schemes, introduced by Forney [24] typically use Reed-Solomon coding as an outer code. This is used to mop up any residual inner code errors, since these errors are by nature bursty.

2.5 Adaptive Power Modulation

An alternative means of compensating for fading is to make use of information about the fading in the received signal to adjust the transmitter power to compensate. This technique is much more power efficient than nonadaptive modulation.

This subject has been investigated in [25] and [26].

Chapter 3

Satellite Channel

A satellite in a circular low earth orbit typically has a period of $1\frac{1}{2}$ to 2 hours. This means that an individual satellite is visible for approximately 10 to 20 minutes at a time. The channel conditions vary substantially during this time of availability.

Multipath fading and shadowing are the most important sources of degradation while the satellite is low on the horizon. When the satellite is high above the horizon, the sources of degradation are diminished. In addition, the satellite is physically closer to the receiver and hence the signal is stronger.

The various factors affecting the signal, and the modelling of these factors, are discussed below.

3.1 Digital Satellite System

The digital satellite system is shown in figure 3.1. The signal to be transmitted is coded and modulated before being transmitted to the satellite. Onboard the satellite, the signal is demodulated and decoded.

Depending on the system, the signal may either be retransmitted immediately, stored for later transmission, or forwarded to another satellite in a multiple satellite system.

For the downlink, the signal is once again coded and modulated before transmission to the ground station.

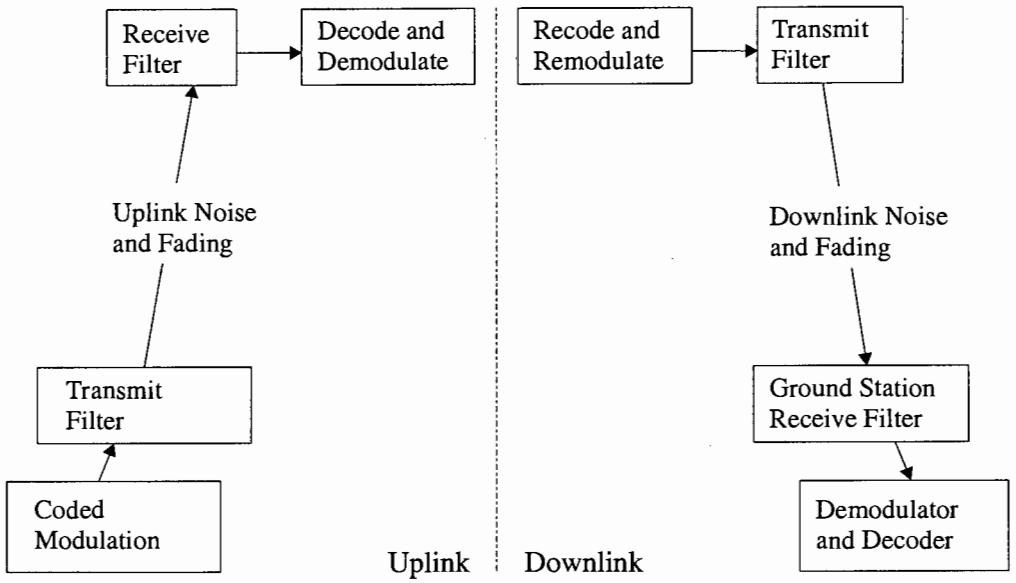

Figure 3.1: Digital Satellite System

3.2 Channel Signal Degradation

The main factors affecting the transmission channel are:

- Fading and Shadowing. The transmitted signal often reaches the receiver along multiple paths due to reflections by obstacles. These signals, having differing phases and amplitudes, can reinforce or cancel each other. Shadowing is the result of the obstruction of radio waves by obstacles such as buildings, trees etc.
- Variation in Distance from receiver. In a near circular LEO orbit, the distance from the satellite to the receiver varies depending on the elevation of the satellite above the horizon. This causes a variation of up to 14dB at typical LEO satellite altitudes.
- Doppler frequency effects. The velocity of the LEO satellite gives rise to doppler frequency shifts of between 2 and 10kHz.
- Interleaving Depth. Interleaving is used to break up error bursts caused by amplitude fades. In order to achieve the best performance, the depth of interleaving should be greater than the maximum anticipated fade duration. If this is not possible, a loss in performance will result.

- Channel Nonlinearities. Since the satellite environment is power limited, it is desirable to run the satellite amplifier as close to saturation as possible. In order to achieve linear operation, it is necessary to reduce the output power.
- Adjacent Channel Interference. Channels are tightly spaced due to bandwidth limitations and, thus, introduce interference in adjacent channels.

3.3 Variation in Satellite Altitude

The signal strength from a satellite in a low earth orbit varies according to the altitude and type of orbit. For the purposes of the discussion below a near circular orbit is assumed, and altitudes of between 550km and 1300km are considered.

The free space path loss is given as a function of the distance between the satellite and the receiver in equation 3.1.

$$P(d) = \frac{P_t}{4\pi d^2} Watts/m^2 \tag{3.1}$$

From the above equation, it is seen that the received power decreases proportionally to the square of the distance between the transmitter and receiver. The distance of the satellite from the receiver depends on the elevation of the satellite above the horizon and is calculated as follows:

Denoting c as the distance from the centre of the earth to the satellite and using the cosine rule

$$c^2 = r^2 + d^2 - 2rd\cos\hat{c}$$

Now $c = r + s$ and $\hat{c} = 90^o$ + satellite elevation θ so by rearranging

$$d^2 - 2rd(1 + cos\theta + r^2 - (r + s)^2 = 0$$

This gives distances of 2200 km and 3186 km for satellite altitudes of 550 km and 1000 km respectively when the satellite is at 5 degrees above the horizon.

For the 550 km orbit, the power loss from the satellite to the receiver is given as

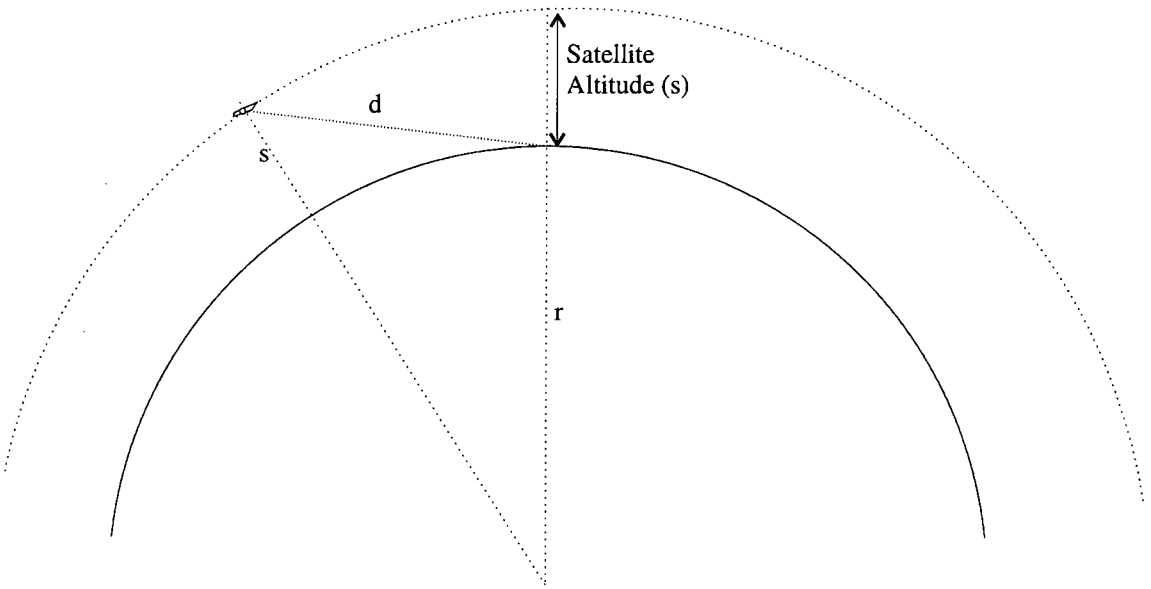

Figure 3.2: Distance of Satellite from Receiver

$$P(5^o) = \frac{P_t}{4\pi \cdot 2\,200 \text{ km}^2} W/m^2$$

The power loss when the satellite is directly overhead is

$$P(90^o) = \frac{P_t}{4\pi \cdot 550 \text{ km}^2} W/m^2$$

This gives the increase in power available when the satellite is directly overhead as:

$$\begin{aligned} \frac{P(5^o)}{P(90^o)} &= \frac{1/(0.55 \times 10^6)^2}{1/(2.20 \times 10^6)^2} \\ &= 16.003 \\ &= 12.04dB \end{aligned}$$

At 1000km this figure drops to approximately 10dB. The effects of the atmosphere have been ignored so far.

At lower elevations, the increased distance through the atmosphere will increase the power loss. This will accentuate these figures. Assuming a receiver signal to noise ratio of 8 db at 5 degrees elevation, this will give a SNR range of approx. 10 - 22 dB.

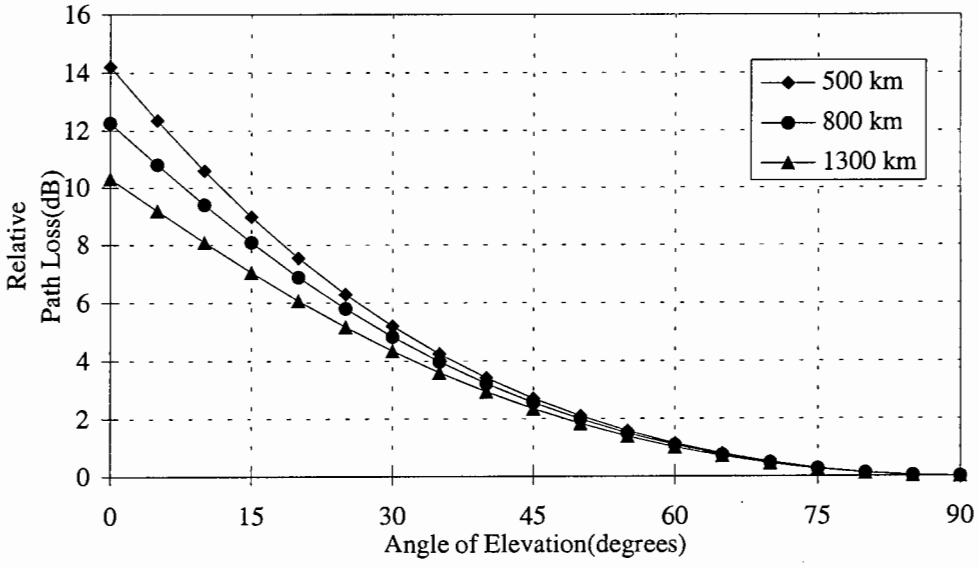

Figure 3.3: Relative Path Losses for Different Satellite Altitudes

3.4 Doppler Shift

The doppler shift is found by comparing the wavelength of the signal with the speed of the satellite relative to the receiver. The doppler shift associated with a LEO satellite at an elevation of 800km with a carrier frequency of 400MHz, is shown in figure 3.4

This doppler shift must be compensated for and tracked in order to provide reliable communications from and to the satellite. This topic, however, does not fall within the scope of this thesis and hence in not covered in any further detail. A detailed model for predicting the doppler shift is given in [27].

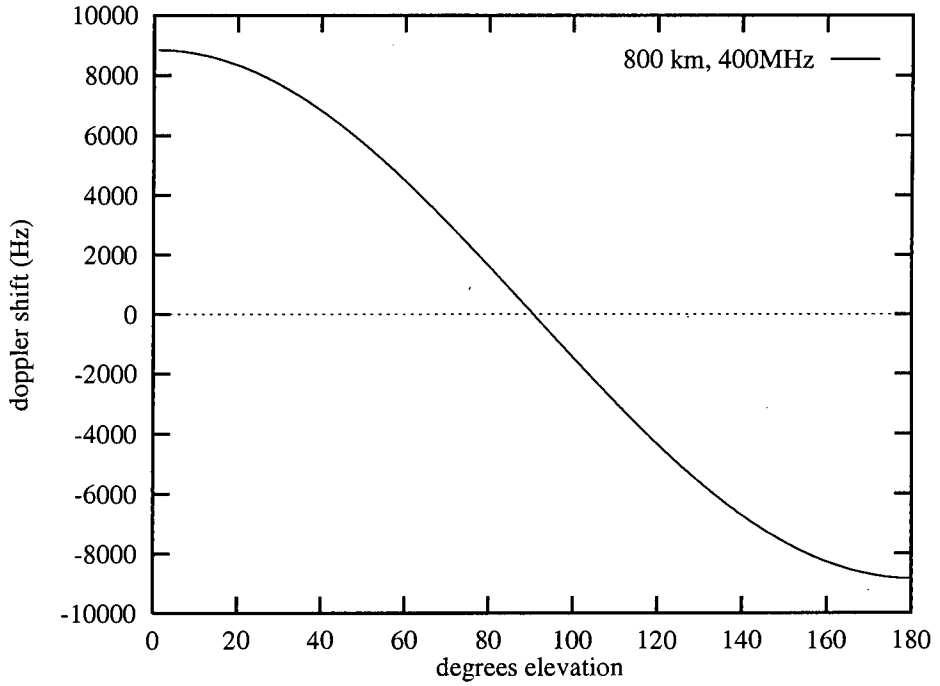

Figure 3.4: Doppler Shift During Overhead pass

3.5 Rician Fading Channel

The Rician channel model considers the received signal to consist of a line-of-sight component, a dominant specular reflection and a background diffuse reflection component. The degree of fading (Rician K parameter) is expressed as the ratio of the power in the Line-of-Sight and specular components to the diffuse power.

In the (non-mobile) LEO satellite situation, when the satellite is directly overhead, the line-of-sight/specular component will dominate, with, as a result, little multipath fading.

When the satellite is close to the horizon, however, the line-of-sight component may be completely blocked, resulting in the "worst case" situation described by Rayleigh fading.

A slow fading channel has been assumed since the doppler spread bandwidth will generally be much lower than the transmission bandwidth for stationary stations and low speed mobile stations.

It will be assumed that the phase is adequately tracked by either a phase-locked loop or pilot tone calibration techniques and thus, only the amplitude fading caused by the channel will be

considered.

By using interleaving of the encoded sequence, it is possible to make the channel approach the ideal memoryless channel. In addition, the interleaving provides a measure of protection from burst errors.

3.5.1 Development of Rician Channel Model

The following development of the Rician Channel is taken from [28].

Three components of the received wave can be identified: A direct line-of-sight component, a specular reflection and a diffuse component. The direct (line-of-sight) and specular components are referred to as the coherent received component and the diffuse signal as the non-coherent component.

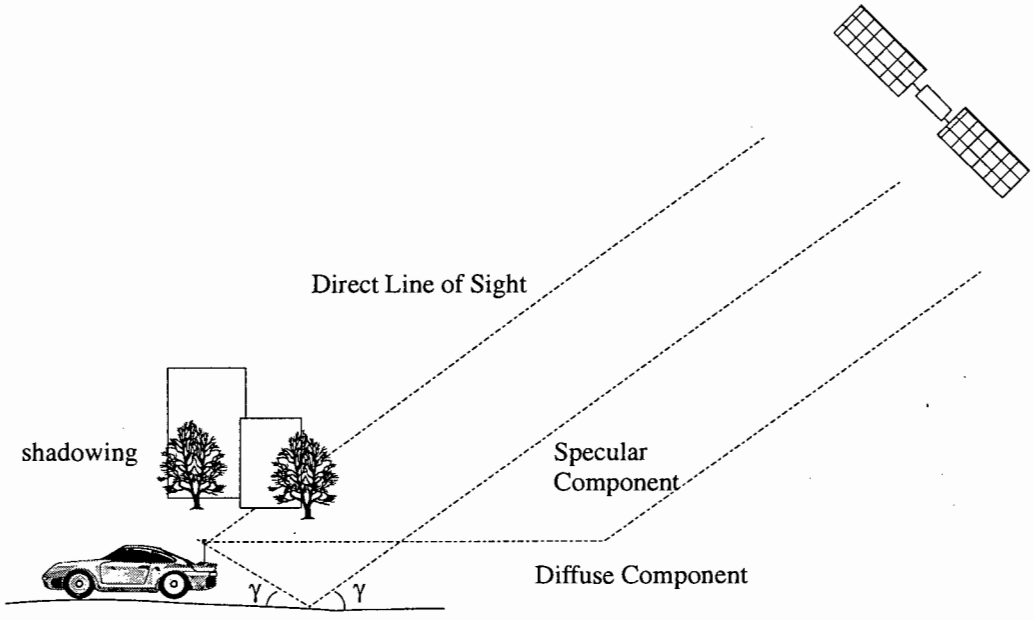

Figure 3.5: Fading Environment

The reflected diffuse component seen by the receiver is given as:

$$e(t) = \sum_{n=1}^{N} e_n \cos\left[(\omega_c - \omega_n)t + \phi_n\right] \tag{3.2}$$

and

$$\omega_n = \frac{2\pi}{\lambda} v \cos \alpha_n \tag{3.3}$$

In equations 3.2 and 3.3 ω_c is the angular carrier frequency , e_n is the amplitude of the nth wave, λ the carrier wavelength and v the velocity of the mobile vehicle. ϕ_n is a uniformly distributed random variable representing the phase of the signal.

The maximum doppler frequency is given as

$$f_d = \frac{v}{\lambda} = \frac{v f_c}{c} \tag{3.4}$$

By the central limit theorem, the combined field components tend towards gaussian processes as N becomes large. For times much shorter than the variations of the signal, the processes are wide-sense stationary and, since they are Gaussian, they are stationary.

We can write

$$e(t) = N_c(t) \cos \omega_c t - N_s(t) \sin \omega_c t \tag{3.5}$$

where $N_c(t)$ and $N_s(t)$ are stationary independent gaussian noise processes. Each process has zero mean and variance σ^2.

Expressing the line-of-sight component as $A \cos [(\omega_c + \omega_d)t]$ and the specular component as $B \cos [(\omega_c + \omega_d)t + \phi_0]$, where ϕ_0 is some arbitary phase.

Defining

$$\mu_c = A \cos \omega_d t + B \cos(\omega_d t + \phi_0) \tag{3.6}$$

and

$$\mu_s = A \sin \omega_d t + B \sin(\omega_d t + \phi_0) \tag{3.7}$$

The Rician parameter K is defined as the ratio of the power of the direct and specular components to the power of the diffuse component.

$$K = \frac{\mu_c^2 + \mu_s^2}{2\sigma^2} \tag{3.8}$$

Fixing the time t we let

$$x = \mu_c + N_c(t) \tag{3.9}$$

and

$$y = \mu_s + N_s(t) \tag{3.10}$$

The amplitude and the phase of the fading process are given as

$$\rho(t) = \sqrt{x^2 + y^2} \tag{3.11}$$

$$\theta(t) = \arctan \frac{y}{x} \tag{3.12}$$

The pdf's of $\rho(t)$ and $\theta(t)$ can be shown to be

$$p(\rho) = 2(K+1)\rho e^{-(K+1)\rho^2 - K} I_0\left(2\rho\sqrt{K(1+K)}\right), \qquad \rho \geq 0 \tag{3.13}$$

$$p(\theta) = \frac{1}{2\pi}e^{-K} + \frac{1}{2}\sqrt{\frac{K}{\pi}}\cos\theta e^{-K\sin^2\theta}\left[1 + erf\left(\sqrt{K}\cos\theta\right)\right], \qquad |\theta| \leq \pi \tag{3.14}$$

where $I_0(\cdot)$ is the zeroth order modified Bessel function of the first kind.

The complex baseband form of the channel is shown in figure 3.6. $F(t)$ is the fading term and $N(t)$ is additive white gaussian noise.

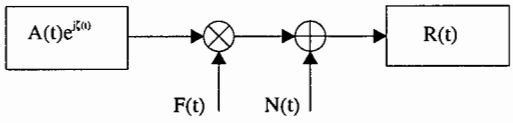

Figure 3.6: Complex Baseband Fading Channel Model

The fading term is given as

$$\begin{aligned} F(t) &= m(t)e^{j\omega_d t} + n_d(t)e^{j\psi t} \\ &= \rho(t)e^{j\theta t} \\ &= [m(t)\mu_c + N_c(t)] + j[m(t)\mu_s + N_s(t)] \end{aligned} \tag{3.15}$$

$N_c(t)$ and $N_s(t)$ are independent coloured gaussian noise processes. In order to generate the fading signal, it is neccessary to generate the power spectrum of the signal. This is determined

from the power spectrum of the diffuse signal $N_c(t) + N_s(t)$.

A short discussion of the calculation of the power spectral density of the diffuse signal is given in appendix A. Equation 3.16 gives an approximation, assuming an omni-directional antenna.

$$S(f) = \frac{1}{\pi\sqrt{1-(f/f_d)^2}} \qquad -f_d \leq f \leq f_d \tag{3.16}$$

To generate the diffuse signal, complex white gaussian noise can be passed through a filter with the transfer function

$$H(f) = \sqrt{S(f)} \tag{3.17}$$

The resultant signal is Rayleigh distributed. A Rician distribution is generated by adding a constant to the gaussian noise signal ie.

$$A + N_c(t) + jN_s(t)$$

3.5.2 Fade Duration

The average duration of fading is an important statistic. This can be determined from the PDF of the Rician/Rayleigh fading process.

The Level-Crossing rate (LCR) is defined as the rate at which the Rayleigh fading envelope, crosses a specific level in a positive going direction. The number of crossings per second (N_R) is given as:

$$N_R = \int_0^\infty \dot{r}p(R,\dot{r})d\dot{r} = \sqrt{2\pi}f_m\rho e^{-\rho^2} \tag{3.18}$$

where f_m is the maximum doppler frequency spread, $\dot{r}(t)$ is the derivative with respect to time of the fading envelope (r(t)), $\rho(R,\dot{r})$ is the joint density function of r and $\dot{r}$ at $r = R.\rho = R/R_{RMS}$ is the specified level R of the fading envelope, normalised to the RMS value of the envelope.

The average fade duration is the average amount of time that a signal drops below a specified level R. For the Rayleigh channel this is given as

$$\text{fade duration} = \frac{1}{N_R} P_r[r(t) < R] \tag{3.19}$$

The probability of r(t) being less than R, $P_r[r(t) < R]$ is found from the fading distribution $p(r)$. For the Rayleigh case this is given as

$$P_r[r(t) < R] = \int_0^R p(r)dr = 1 - e^{-\rho^2} \tag{3.20}$$

The average fade duration for a Rayleigh distribution is thus given as

$$\text{fade duration} = \frac{e^{\rho^2} - 1}{\rho f_m \sqrt{2\pi}} \tag{3.21}$$

The Rician case can be derived in the same way, with the substitution of the Rician fading density.

The maximum doppler spread will depend mostly on the speed of the mobile station. The doppler shift introduced by the movement of the satellite itself is not considered since the doppler shift described in section 3.4 does not cause the multipath effects described in A.

Assuming a maximum mobile speed of 60km/h and a carrier frequency of 400MHz, this gives a maximum doppler frequency of 22 Hz. In Rayleigh conditions, the average duration that the signal would be below 1/10 the RMS ($\rho = 0.1$) level is given as

$$\begin{aligned} \text{fade duration} &= \frac{e^{.1^2} - 1}{(.1)20\sqrt{2\pi}} \\ &= 1.82ms \\ &\approx 18\text{symbol intervals} \end{aligned}$$

This gives an indication of the average number of symbols likely to be affected by a particular fade.

Chapter 4

Selection and Design of Modulation Scheme

The channel model discussed in the previous chapter shows large variations in conditions and signal to noise ratios. An optimal modulation scheme would be able to take advantage of the increased signal to noise ratios, when available, while maintaining current transmission rates when signal conditions deteriorate.

4.1 Design Constraints and Considerations

The main requirement of the system is that is should provide acceptable communication rates during poor signal conditions, but be able to take advantage of the additional signal power available when the satellite is at high elevations.

Other design considerations include:

- Bandwidth and power efficiency. The system needs to operate in a channelised environment and hence is strictly bandlimited. The satellite amplifier is power limited.
- Robustness in fading and shadowing conditions. In order to operate to provide maximum availability, it is necessary to be able to operate at low elevations. At these low elevations, fading and shadowing are particularly important and must be dealt with.

- Relatively low hardware complexity. The modulator/demodulator and the coding used needs to be implemented on board the satellite, and thus it is desirable to keep the hardware complexity down.

4.2 Selection of modulation scheme

The channel considered is a 9.6kHz channel. It is both power limited and bandwidth limited and thus both factors must be taken into consideration.

Popular choices for this type of channel are Continuous Phase Frequency Shift Keying (CPFSK) modulations such as Minimum Shift Keying(MSK). MSK, in particular, has very low out-of-band power. In addition, the ability to use them in non-linear systems makes them attractive in power limited cases, where it is desirable to operate the amplifiers close to saturation.

In order to increase the transmission rate to rates of above 3 bps/Hz, however, it is necessary to consider more spectrally efficient modulation schemes such as Quadrature Amplitude Modulation (QAM) schemes. These modulation schemes use both the phase and the amplitude of the signal to convey information, and thus need a linear channel. This results in requiring a larger output power backoff at the amplifier.

The system is required to be able to change transmission rates as the SNR conditions change. In a system that combines coding and modulation, this can be accomplished by altering, either, or both these factors.

4.3 Coding Scheme

Trellis Coded Modulation allows for substantial coding gains in an additive white gaussian noise environment. The trellis code, by transmitting only a subset of the total constellation in each symbol interval, increases the minimum distance between signal points.

The cost of this gain is an increase in the signal constellation size. The constellation expansion ratio(CER) is typically about double (i.e. for an equivalent rate to 64-QAM the 128-CROSS constellation could be used) although it is possible to obtain smaller CERs by using higher dimensional spaces.

Trellis coding, on its own, is not ideally suited to the channel. The convolutional coding is very susceptible to burst errors and the usual AWGN design of the trellis code is not optimal in fading conditions. These factors lead to its combination with other codes in a concatenated coding scheme.

Although the trellis coding is based on convolutional codes, it is difficult to vary the rate of transmission by changing the rate of the convolutional code due to the tight linkage between the constellation and the coding inherent in TCM. For this reason it was decided to rather make use of different constellation sets for the purpose of changing coding rates.

4.3.1 8-Dimensional Trellis Code

Trellis coded modulation expands the constellation over what is needed for uncoded transmission, in order to achieve coding gain. A typical constellation expansion ratio(CER) for two dimensional constellations is a factor of 2. In order to transmit at a maximum rate of 48kbps, uncoded transmission would require a 32 point constellation such as the CROSS-32 constellation. A trellis coded scheme, based on a two dimensional constellation, would require a constellation expansion ratio of approximately double the size. We will choose an 8 dimensional constellation mapping, in order to reduce this constellation expansion ratio. In addition, it is simpler to make the constellation rotationally invariant to 90^o phase shifts, which is very desirable for recovery purposes. Based on the Wei construction [15], we will proceed as follows. It is desired to transmit 48kbps at 9600 baud. This is a rate of 5 bits per baud. An 8 dimensional constellation gives N = 4 baud intervals over which to send the constellation. The number of bits will be $5 \times 4 + 1 = 21$ bits. Hence the 8 dimensional constellation will include 2^{21} signals.

Each two dimensional constellations will include

$$(1 + \frac{1}{4})2^5 = 40$$

signals, instead of the 64 needed if an extra bit per each two dimensional constellation was added. The 40 point constellation is shown in figure 4.1. It has 32 bits, arranged as per the CROSS-32 constellation, and 8 points in an outer group. The 8 dimensional constellation is formed by concatenating four of the 40 point, two dimensional constellations and excluding

those signals that include more than one point in the outer group.

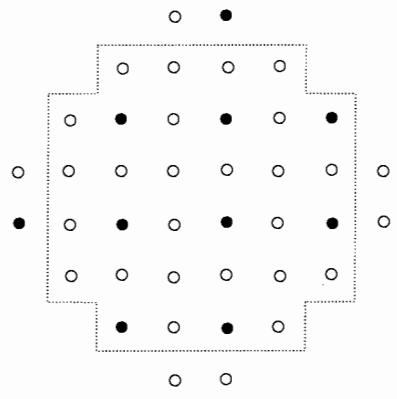

Figure 4.1: Expanded 40 point constellation

The choice of a 3/4 rate convolutional encoder is made, requiring the 8 dimensional constellation to be partitioned into $2^4 = 16$ subconstellations. This partitioning, taken from [28] as per the guidelines in [15], is done as follows:

1. A four way partition of the original two dimensional constellation, is performed. Each 2D subconstellation has 10 signals, 8 in the inner group and two in the outer group. The minimum squared distance(MSD) between the constellation points is $4d^2$, where d^2 is the free distance of the original 40 point constellation. The four subconstellations are labelled A,B,C,D. Each subconstellation can be obtained by a multiple-ninety degree rotation of the others.

2. 16 four-dimensional 'types' are defined by concatenating pairs of two dimensional subconstellations, (A,A) (A,B) ... (D,D). The MSD of each type remains $4d^2$ and each 'type' contains 10x10 = 100 signals.

3. Pairs of the 16 types are grouped in order to form a new set of 8 four dimensional 'types'. The pairing is done such that the MSD of $4d^2$ is maintained. The new constellations have 200 signals and are chosen as

   1. $(A, A) \bigcup (B, B)$
   2. $(C, C) \bigcup (D, D)$
   3. $(A, B) \bigcup (B, A)$
   4. $(C, D) \bigcup (D, C)$

5. $(A, C) \bigcup (B, D)$
6. $(C, B) \bigcup (D, A)$
7. $(A, D) \bigcup (B, C)$
8. $(C, A) \bigcup (D, B)$

4. 64 Eight-dimensional 'types' are formed by concatenating pairs of these subconstellations: (1,1),(1,1) .. (8,8). Each type has 200^2 signals.

5. Finally the 64 eight dimensional types are grouped into 16 eight dimensional subconstellations with minimum distance $4d^2$ Each subconstellation will have 640×10^3 signals. This is shown in table 4.1

Subconstellation	Convolutional Code Output	Eight-Dimensional Types
1	0 0 0 0	(1,1,(2,2),(3,3),(4,4)
2	0 0 0 1	(1,2),(2,1),(3,4),(4,3)
3	0 0 1 0	(1,3),(2,4),(3,1),(4,2)
4	0 0 1 1	(1,4),(2,3),(3,2),(4,1)
5	0 1 0 0	(5,5),(6,6),(7,7),(8,8)
6	0 1 0 1	(5,6),(6,5),(7,8),(8,7)
7	0 1 1 0	(5,7),(6,8),(7,5),(8,6)
8	0 1 1 1	(5,8),(6,7),(7,6),(8,5)
9	1 0 0 0	(1,5),(2,6),(3,7),(4,8)
10	1 0 0 1	(1,6),(2,5),(3,8),(4,7)
11	1 0 1 0	(1,7),(2,8),(3,5),(4,6)
12	1 0 1 1	(1,8),(2,7),(3,6),(4,5)
13	1 1 0 0	(5,1),(6,2),(6,3),(8,4)
14	1 1 1 0	(5,2),(6,1),(6,4),(8,3)
15	1 1 1 0	(5,3),(6,4),(6,1),(8,2)
16	1 1 1 1	(5,4),(6,3),(6,2),(8,1)

Table 4.1: The 16 subconstellations

Encoder Design

The convolutional encoder selects, at each output step, one of the 16 eight-dimensional types. For each of these types, it is possible to transmit one of 16 different 8 dimensional signals.

Four of the remaining bits are used to select one of these signals in such a way that the design remains rotationally invariant to 90 degrees.

These 16 signals have been designed so that each contains four unique signals and 90,180 and 270 degree rotations of each of these signals. The first of the 16 types is shown in table 4.2.

	0^o		90^o		180^o		270^o
1	AAAA	$\Rightarrow$	CCCC	$\Rightarrow$	BBBB	$\Rightarrow$	DDDD
2	AABB	$\Rightarrow$	CCDD	$\Rightarrow$	BBAA	$\Rightarrow$	DDCC
3	ABAB	$\Rightarrow$	CDCD	$\Rightarrow$	BABA	$\Rightarrow$	DCDC
4	ABBA	$\Rightarrow$	CDDC	$\Rightarrow$	BAAB	$\Rightarrow$	DCCD

Table 4.2: Signals in first 8 dimensional type

From, this the procedure is seen to be to select one of the 4 unique signals with two of the bits. The last two bits are used to determine which rotation of the signal to use. Since it is wished to keep the modulation rotationally invariant, these two bits are differentially encoded so that only the relative rotation, and not the absolute rotation, is important.

This leaves 13 bits to be transmitted over the 10 signal points in each of the 4 2-D subconstellations. In order to perform this mapping, an 8 dimensional block encoder is used.

The block encoder is used to select the signal within the chosen TCM set. It is designed in such a way that only one of the four timeslots will contain an outer signal point. In this way, on average, only one in eight symbols is an outer constellation point.

The block encoder taken from [17], is shown in table 4.3.

			n			n+1			n+2			n+3		
I_1	I_2	I_3	O_3	O_2	O_1	O_3	O_2	O_1	O_3	O_2	O_1	O_3	O_2	O_1
0	X	X	0	I_2	I_3	0	I_4	I_5	0	I_6	I_7	0	I_8	I_9
1	0	0	1	0	0	0	I_4	I_5	0	I_6	I_7	0	I_8	I_9
1	0	1	0	I_4	I_5	1	0	0	0	I_6	I_7	0	I_8	I_9
1	1	0	0	I_4	I_5	0	I_6	I_7	1	0	0	0	I_8	I_9
1	1	1	0	I_4	I_5	0	I_6	I_7	0	I_8	I_9	1	0	0

Table 4.3: Block coder for 20 point constellation

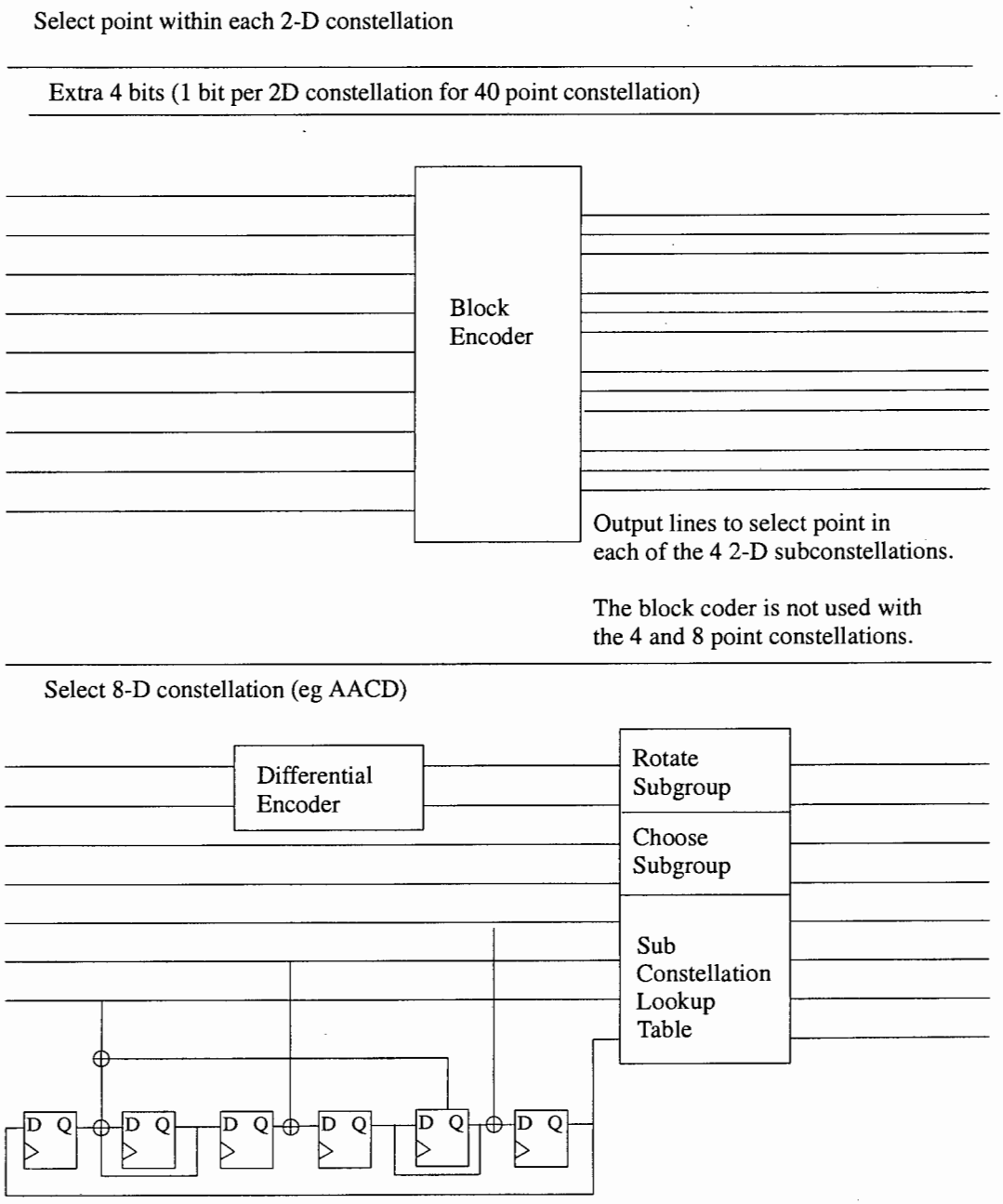

Figure 4.2: Diagram of Coder

Decoder Design

The I and Q channels are quantised at the receiver, and these values are used to calculate the metrics for the viterbi decoder. The signal points in the constellation are set up in a ROM. The four subsets A, B, C and D are 90^o rotations of each other, thus they can either be stored separately, or, as in figure 4.3, derived from each other.

A change in constellation set can be made by changing the address accessed in the ROM.

The decoder needs to calculate a metric to each of the 16 eight-dimensional 'types'. This is done sequentially, as per the design process.

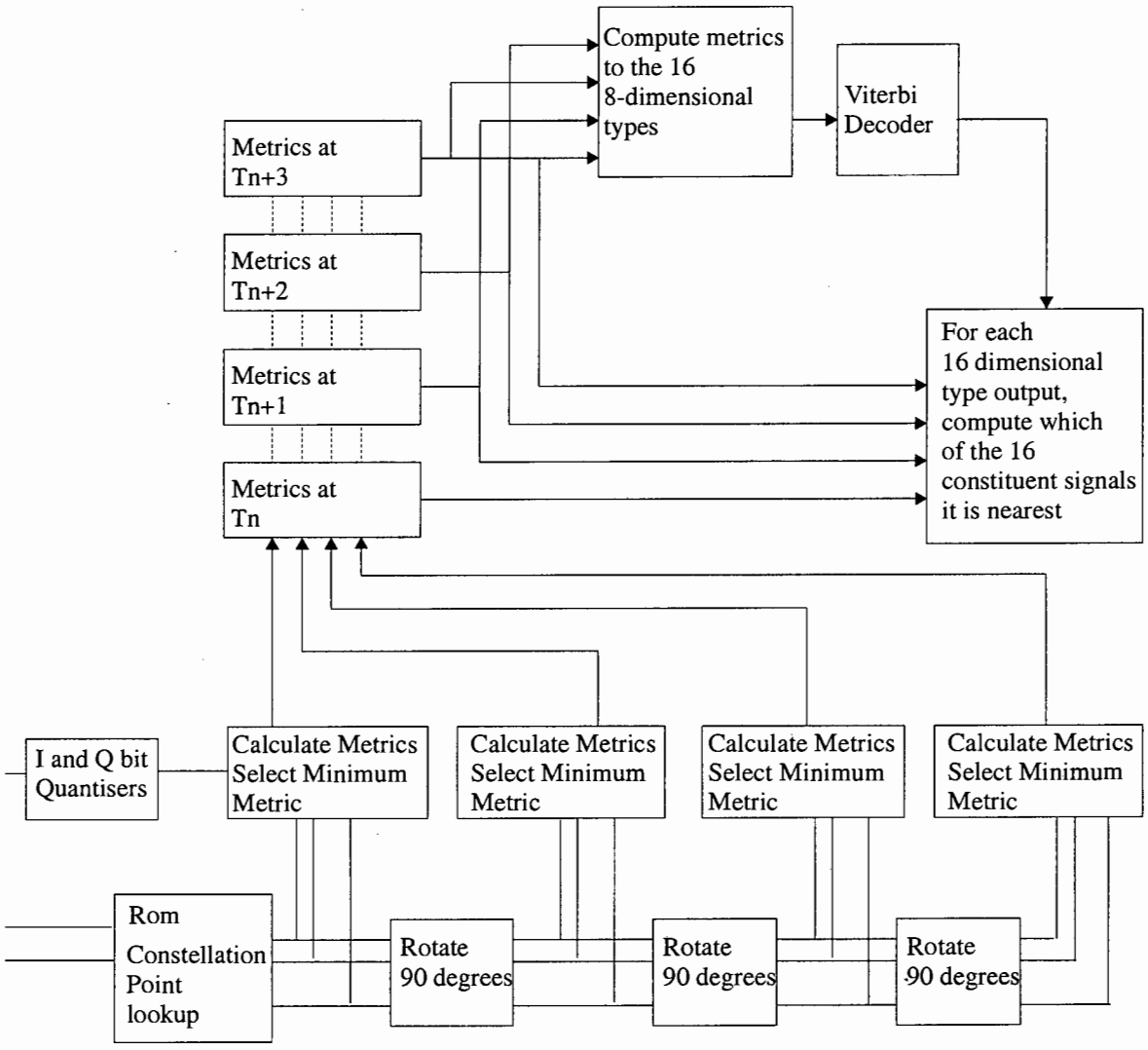

Figure 4.3: Diagram of Decoder

1. A metric to each constellation signal point is calculated in each of the 4 two-dimensional constellations. A minimum distance to each subconstellation (A,B,C or D) is obtained by taking the minimum distance

2. A metric to each of the 16 four-dimensional types is calculated by adding the squared metrics of the constituent pairs ie. (A,A), (A,B), (D,D). This is then reduced to the 8 four-dimensional types.

3. A metric to each of the 64 eight-dimensional types is calculated. This list of metrics is then reduced to the 16 eight-dimensional types and the metrics are ready for the viterbi decoding process.

A soft-decision viterbi decoder is used for decoding the convolutional code. The path depth, after which a decision will be made, can be limited to 15 - 20 eight-dimensional symbols. Almost all of the coding gain is attained by this choice [17].

The output of the Viterbi decoder gives the 8 dimensional trellis symbol eg. A B C A. The closest point in each of these four 2-D symbols to the received point, is chosen as the received signal. The rest of the bits are then extracted.

Coding gain

Using a 64 state 3/4 rate convolutional code gives a minimum error path length greater than the minimum distance between signal points in the type (parallel transitions in the state diagram).

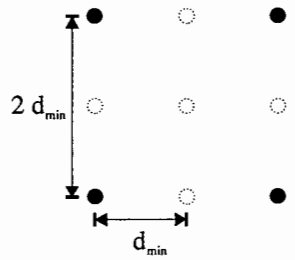

Figure 4.4: Minimum distance

The distance between parallel transitions is twice that of the uncoded case. The asymptotic

coding gain is thus given as

$$\begin{aligned}
\text{coding gain} &= 10\log\left[\frac{\text{coded free distance}^2}{\text{uncoded free distance}^2}\right] dB \\
&= 10\log\left[\frac{(2d_{min})^2/E}{(d_{min})^2/E'}\right] \text{dB} \\
&= 10\log\left(\frac{4E}{E'}\right) \text{dB} \\
&= 6 - 10\log\left(\frac{E'}{E}\right) \text{dB}
\end{aligned}$$

where E is the original average energy and E' is the average energy of the expanded constellation. The penalty for the expanded constellation is approximately .6 to 1 dB, depending on the constellation.

The asymptotic coding gain is approached at bit error rates of approximately 10^{-6}. At the tested bit error rate of 10^{-3}, the coding gain was found to be approximately 2 to 2.5 dB.

4.3.2 Adaptive Rate Transmission

Although the proposed trellis coded modulation scheme was discussed with a particular constellation set in mind, it is possible to use any 2-D constellation that can be divided into 4 sets with it.

This is the basis of the proposed mechanism for switching coding rates. Some of the constellations that have been used with this trellis code are shown in figure 4.5. These constellations range from the basic QPSK constellation to the modification of the CROSS-32 constellation discussed earlier, providing a transmission rate range between 1.5 and 5 bits per symbol.

This scheme allows constellation sets with greater free distance to be used in low SNR and fading conditions.

As mentioned earlier, the additional constellations can be implemented with minimal additional transmitter/ receiver complexity, as they rely on the same 3/4 rate convolutional encoder/decoder.

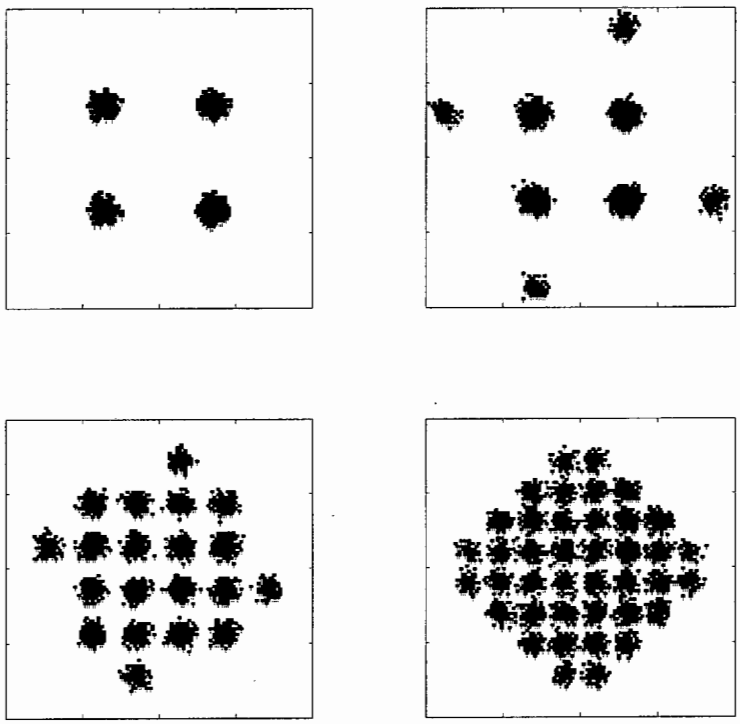

Figure 4.5: Constellation Sets

4.4 Protection against Fading

As described, the proposed modulation scheme is, in itself, not well suited to the satellite channel. In particular, it is very sensitive to burst errors and is not designed optimally to cope with fading.

The following strategies are considered in order to make the overall system more robust in these channel conditions:

- Interleaving
- Reed-Solomon Outer Code
- Channel State Information

4.4.1 Interleaving

Interleaving is a general means of providing a memoryless channel. The output of the modulator is interleaved before transmission to break up regions of fading. If the interleaving depth is kept above the maximum expected fade duration, interleaving provides a good approximation of the perfect memoryless channel.

Errors in the output of the Viterbi will be bursty due to the nature of convolutional decoding. Interleaving is thus employed on the output of the Viterbi Decoder in order to break up the errors for the block decoder.

For simplicity, only block interleaving has been considered: However, convolutional and other types of interleaving are also applicable.

4.4.2 Reed-Solomon Coding

Reed Solomon Codes are maximal free distance cyclic block codes that are particularly suited for burst error correction.

A (n, k) Reed-Solomon code is a cyclic block code that codes k bits in a n bit block. The code is able to correct $n - k + 1/2$ symbol errors.

For example, a (63,31) Reed-Solomon code is able to correct 15 consecutive symbols, while providing a code rate of approximately $1/2$. A (63,47) Code can correct 8 symbols, while providing a code rate of $3/4$.

In the case of a stationary receiver, the amount of fading and shadowing will be largely determined by the elevation of the satellite above the horizon. When the elevation is high, fading will be minimal and it may be decided not to make used of the additional Reed-Solomon coding.

For a mobile receiver, however, the amount of fading and shadowing will be more dependant on the speed of the mobile and the terrain it is moving through.

4.5 Channel State Information

Channel State information provides a measure of the amount of fading experienced in the channel. This is particularly important for multilevel constellations such as the QAM constellations in fading conditions.

4.5.1 Modification of Viterbi Algorithm

The Viterbi Decoder is an implementation of the maximum likelihood decoder.

Suppose a sequence of M-ary signals are transmitted, say:

$x = (x_0, x_1, x_2, ..., x_{K-1})$

These symbols are used to modulate a signal, $s(t)$. The resulting transmitted signal can be represented as

$$y(t) = \sum_{k=0}^{K-1} x_k s(t - kT) + n(t) \tag{4.1}$$

where T is the symbol period and n(t) is an additive white gaussian noise signal. The demodulator processes the received signal, in order to produce an estimate of the transmitted vector $\hat{x}$

$\hat{x} = (\hat{x}_0, \hat{x}_1, ..., \hat{x}_{K-1})$

The maximum likelihood decoder will choose $\hat{x}$ if

$$P\left[y(t) \mid \hat{x}\right] = \max_{x} P\left[y(t) \mid x\right]$$

For the additive white gaussian noise channel, we can express the output y(t) as a sum of time samples.

$$y(t) = \sum_{k=0}^{K-1} y(t - kT)$$

The probabilities are then

$\log P[y(t) \mid x] = \sum_{k=0}^{K-1} \log P[Y(t - kT)] \mid x_i] = C - \sum_{k=0}^{K-1} \|Y(t - kT) - x_k s(t - kT)\|^2$

where $\|f(t)\|^2$is defined as the Euclidean distance between the signals. The constant can be disregarded in the maximisation, so the metric can be expressed as

$$m[y(t), x] = \sum_{k=0}^{K-1} \|Y(t - kT) - x_k s(t - kT)\|^2 \tag{4.2}$$

The maximum likelihood rule is thus the minimisation, with respect to all possible sequences of $x$of the metric.

For an amplitude fading channel, equation 4.1 can be expressed as

$$y_k = \rho_k x_k + n_k \tag{4.3}$$

where p_k is the amplitude of the fading process at time k.

An assumption of ideal channel state information is equivalent to assuming that the channel state information is exactly the amplitude of the fading process. This can be achieved by altering the metric in equation 4.2 to

$$m[y, x, \rho] = \sum_{k=0}^{K-1} \|y_k - x_k \rho_k\|^2 \tag{4.4}$$

4.5.2 Obtaining the Channel State

For simple constellations such as QPSK and the 8 point constellation, the estimate of the channel state may be derived from the amplitude of each individual symbol.(In the case of the 8 point constellation, the one symbol interval that is allowed to extend beyond the basic 4 point constellation is ignored.)

For larger constellations (ie. the constellations that rely on amplitude information in addition to phase information), it is necessary to make use of other methods to derive CSI.

The insertion of a pilot tone is a common technique to accomplish this. The pilot tone can be used to accomplish phase synchronisation as well as to give an estimate of the channel state.

Chapter 5

System Overview

In Chapter 4 the various components of the proposed system were discussed. The integration of these components, and the various design choices involved for a practical system, are discussed further below.

5.1 Complete System

The complete coded modulation scheme is shown in figure 5.1.

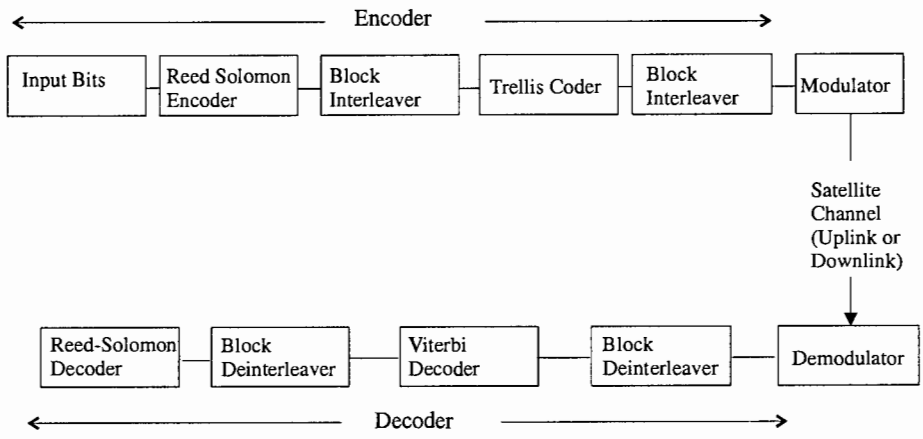

Figure 5.1: Complete Coded Modulation Scheme

The signal is first Reed-Solomon encoded, before it is sent to an optional interleaving stage. The signal is then fed to the Trellis coder.

The trellis code maps the signal onto the chosen constellation before modulation.

Onboard the satellite, the signal is demultiplexed and deinterleaved before it is sent to the Viterbi decoder. The second deinterleaver is then used, followed by the Reed-Solomon decoder.

The binary signal may then be stored for later retransmission, forwarded to another satellite, or retransmitted immediately, depending on the usage of the satellite or network of satellites,

The downlink from the satellite is identical in design.

5.2 Design Choices

The above design leaves several parameters of the system to be designed, according to the particular requirements of the implementation.

The following design choices need to be made:

- Block Size and rate of Reed-Solomon Coding: The choice of coding will affect the rate and reliability of the overall transmission. It may be decided not to use the Reed-Solomon coding in times of high signal to noise ratios, and not much fading.
- Depth of Interleavers: The size of the interleavers needs to be chosen.
- Constellation Choice: The 2-D constellations chosen to be used, should be chosen with the specific target bit rates in mind. While several constellations have been suggested, as mentioned before, it is possible to use any constellation that has a multiple of 4 signal points.
- Channel State Information: A choice of what, if any, channel state information is used, needs to be made. This will improve performance in multipath fading conditions, but adds some additional design complexity.

5.3 Simulated System

In order to simulate the system, the following design choices were made:

1. Reed-Solomon Code: (63,31) and (63,47) Reed-Solomon codes were used. These provide correction of 15 and 7 errors, respectively, with moderate complexity.

2. Constellations: The 4 constellation sets shown in figure 4.5 were used.

3. Channel State Information: A single bit CSI was used. This is derived directly from the automatic gain control of the receiver.

With these choices, it is possible to use many different variations to achieve certain data rates. The combinations of Reed-Solomon coding and constellation sets that were chosen for simulation are shown in 5.1.

Constellation	RS Coding	Bits Per Symbol
4 point	(63,31)	0.875
4 point	none	1.875
8 point	(63,31)	1bps
8 point	none	2 bps
20 point	(63,47)	3 bps
20 point	none	4bps
40 point	none	6bps

Table 5.1: Simulated Data Rates

5.4 Simulation Model

In order to test the proposed system, a baseband simulation was used. The simulation tests the error rate of the bandlimited system, in Rician fading and AWGN conditions.

5.5 Simulation Parameters

The simulation model is shown in figure 5.2.

The following parameters were used for simulation:

- Simulation Resolution: The timestep used was $11.57\mu s$, corresponding to a maximum Nyquist frequency of 43200kHz.

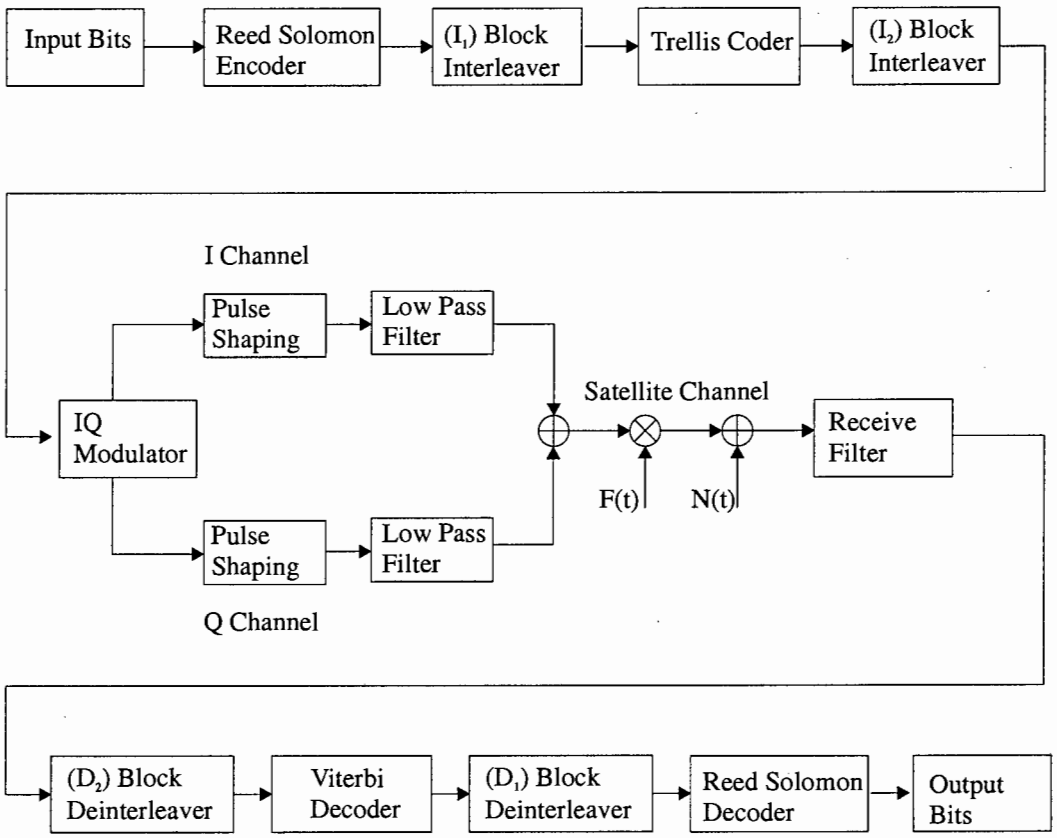

Figure 5.2: Simulation Model

- Pulse Shaping Filter: The pulse shaping filter is a Raised-Cosine Filter with a roll-off of 0.5.
- Low pass filter: A fourth order butterworth filter was used. The cutoff frequency is 9600kHz.
- Doppler Filter: The simplified model for an omnidirectional antenna given in Appendix A was used. The maximum doppler frequency used was 3kHz.
- Viterbi Decoder: The 64 state Viterbi Decoder suggested by WEI [15] was used in the system. The truncated trellis depth used to make decisions was 20 8-D symbols based on the investigation by Tretter[17].

Interleaver I_2 did not contribute to significant performance improvement, over a range of different interleaving depths and different average fade durations.

A performance improvement was, however, seen over the same channel when using a hard-decision block coding scheme. Further investigation is needed to determine whether the the method is inappropriate (interleaving has traditionally been employed for hard decision channels, although it has been suggested that it is just as applicable for soft-decisions [29].) or whether the discrepancy is a result of inadequacies, or errors, in the simulation.

The simulation was performed largely in Matlab. It was necessary to implement the Viterbi decoder since the design of trellis coding closely ties the constellation with the coder. The Viterbi decoder, and the associated mapping of metrics from the 2-D constellations, were implemented in C++ due to speed considerations.

Chapter 6

Adaptive Transmission Rate

The mechanism of changing data rates by selection of different constellations, has been discussed in previous chapters. The decision of when to make these changes has not yet been addressed.

The specific configuration of the system must be considered when making these decisions.

6.1 System Configuration Considerations

For a variable rate to be used, both the transmitter and receiver must agree upon a fixed data rate.

All stations that listen to the satellite's retransmission will also need to agree on a common rate. This essentially limits the system to a two-user system with a single receiving station (per channel).

Two way communication between the transmitter and receiver is necessary in order to provide feedback about the current signal conditions.

6.2 Transmission rate switching

In order to change the data rate, the receiving station must make use of several parameters to decide the most suitable modulation scheme for the current channel conditions.

Some of the factors that may be used are:

- The absolute distance of the satellite to the ground station
- The elevation of the satellite above the horizon
- Channel State Information, derived from the automatic gain control
- Feedback from the Viterbi Decoder

A combination of these factors allows a good assesment of whether it will be possible for a particular combination of coding and modulation to provide transmission at the required bit error rate.

6.2.1 Distance of Satellite from Ground Station

The distance between the satellite and the ground station provides an upper bound of the available power that reaches the receiver (section 3.3). This can be used as an overall guide as to what range of transmission rates are possible.

6.2.2 Elevation of satellite above horizon

When the satellite is low on the horizon, the transmission will be more affected by multipath fading and shadowing. This gives a second estimate based purely on the position of the satellite.

6.2.3 Channel State Information

Use of feedback from the automatic gain control allows an estimate of the degree of multi-path fading that the tranmission is experiencing. In the case of a mobile receiver, this is a particularly useful estimate.

6.2.4 Path metrics of Viterbi Decoder

As the metric of the decision path of the Viterbi Decoder tends towards that of other paths, the likelihood of errors increases. A measurement of the average metric between the chosen path and the next best path, would alert the decoder of deterioriation in the channel. The rate could be dynamically adjusted accordingly.

Specifically, in figure 6.1, path A is chosen as the received path. If the next best path to that node is path B, the difference between these paths will give an indication of how reliable the decisions are.

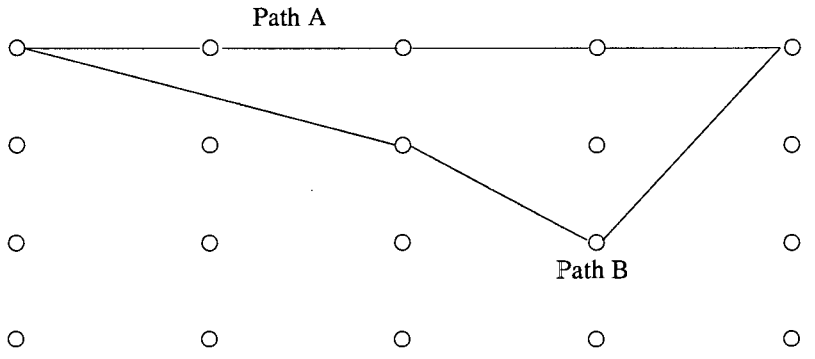

Figure 6.1: Viterbi Metrics

These differences may be averaged over a time period in order to make a decision on whether or not to change rates.

Figure 6.2 shows the results of a simulated test with the 20 point constellation in AWGN conditions. This suggests that it is indeed possible to use this metric as a guide of when to change rates. More experimentation is required to determine the reliability of this metric under different conditions.

6.3 Adaptive Rate Controller

The design of a controller that makes use of the above stated factors in order to make decisions about when to change rates, is a complex task and is not fully addressed here.

The first two factors can be predetermined from the orbit of the satellite and these give an achieveable range of signal strengths. The last two factors make use of dynamic information

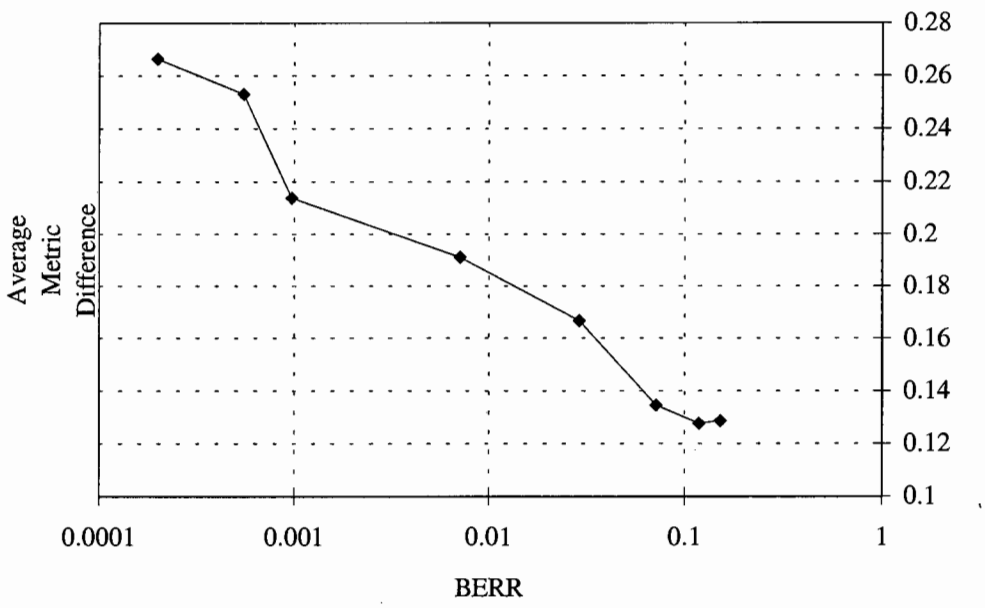

Figure 6.2: Average Difference in Metrics versus bit error rate

about the channel.

A simple Fuzzy controller is suggested for the system, due to the simplicity of design. Such a controller will produce smoothly interpolated decisions and is easy to fine tune in an intuitive way, depending on the precise requirements of the channel. No guarantee of optimality is, however, possible.

Chapter 7

Simulation and Discussion

The coding and modulation system that was detailed in chapter 4 was tested by means of computer simulation, according to the model described in chapter 5. The focus of the testing was bit error performance of the various configurations of the proposed system under AWGN and Rician/Rayleigh fading conditions.

7.1 Bandwidth utilisation

Due to the need to operate in a channelised FDMA environment, each channel must be strictly bandlimited.

Figure 7.1 shows the transmission spectrum, while utilising the 40 point constellation. The transmission was prefiltered with a raised cosine filter with roll-off of .5, before being passed through a eight-order butterworth filter.

The filtered signal has low out-of-band power and thus will not cause much interchannel interference. This is important for a channelised environment and the system was thus tested with the above band-limiting filters. The effect of interchannel interference was, however, not tested.

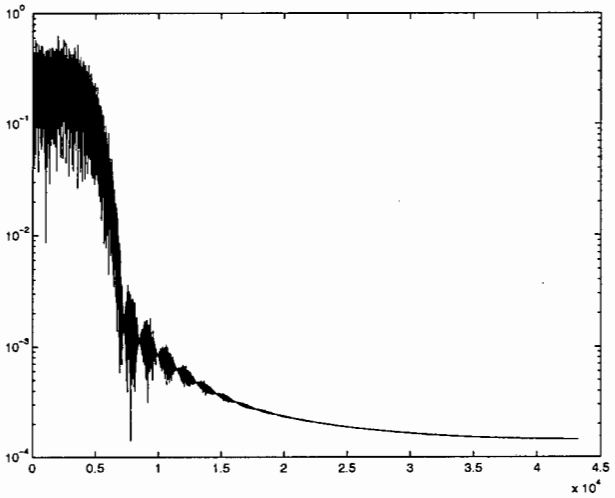

Figure 7.1: Transmission bandwidth (40 point constellation)

7.2 Performance under AWGN Conditions

While the satellite is at high elevations, fading and shadowing will be at a minimum and the channel will tend towards AWGN conditions.

Figure 7.2 shows which constellation sets can be used at different signal to noise ratios, assuming a bit-error rate of 10^-3 is required. No additional coding or interleaving was used.

This provides 1 3/4 bits/symbol transmission at signal to noise ratios as low as 7dB up to 5 bits/symbol at 18dB.

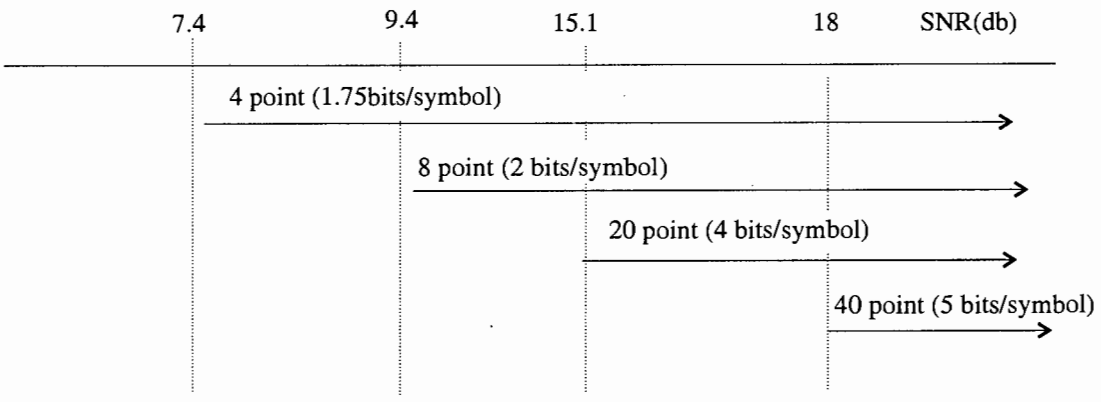

Figure 7.2: Performance under AWGN conditions

Figure 7.3 shows the incorporation of Reed-Solomon coding into the scheme. Incorporating

Reed-Solomon coding leads to lower required SNR's for each constellation set, and better protection against burst errors. An 8 8-D symbol interleaver was used between the Reed-Solomon coder and the trellis coder.

As expected, the results shown in figure 7.3 show an improvement over those in figure 7.2 at the expense of a slightly lower bit rate. The use of high rate Reed-Solomon codes provides good error performance, without sacrificing the data rate too much.

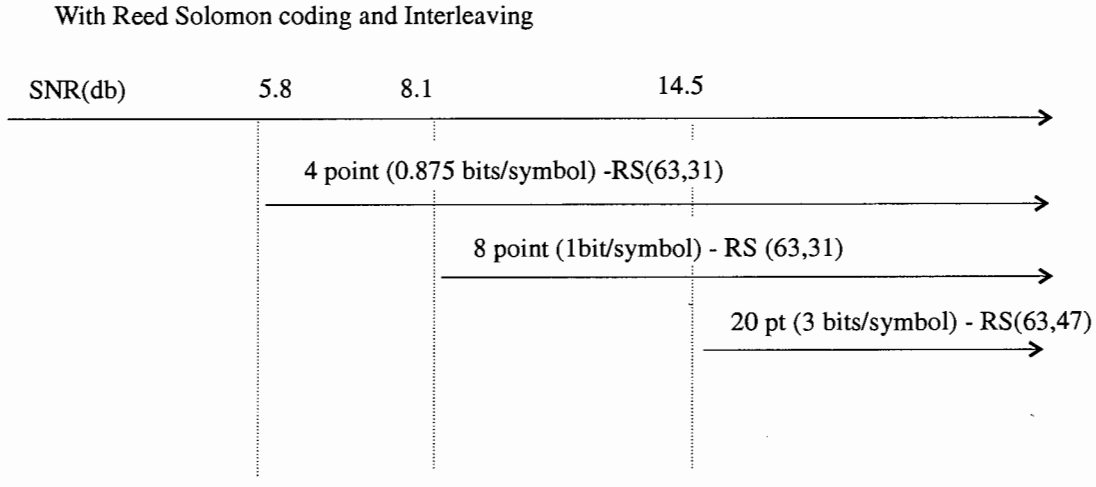

Figure 7.3: With Reed Solomon Encoding and Interleaving

7.3 Performance under Rician Fading Conditions

In a non-mobile environment, Rician and Rayleigh fading will occur primarily when the satellite is low on the horizon, where the signal strength is weak as well.

The 4 point (QPSK) and 8 point constellations were thus tested most for sensitivity to fading.

Figure 7.4 shows the performance of the 4 point constellation without additional Reed-Solomon encoding, under different degrees of Rician and Rayleigh fading.

Figure 7.5 shows the same system used in conjunction with a high rate (63,47) Reed Solomon encoder. Interleaving was used on both the output of the convolutional encoder and between the convolutional encoder and the Reed-Solomon encoder.

The performance improvement is most noticeable under Rayleigh fading, where an improvement of almost 2 dB is seen at the nominal bit error rate of 10^{-3}. Under Rician fading

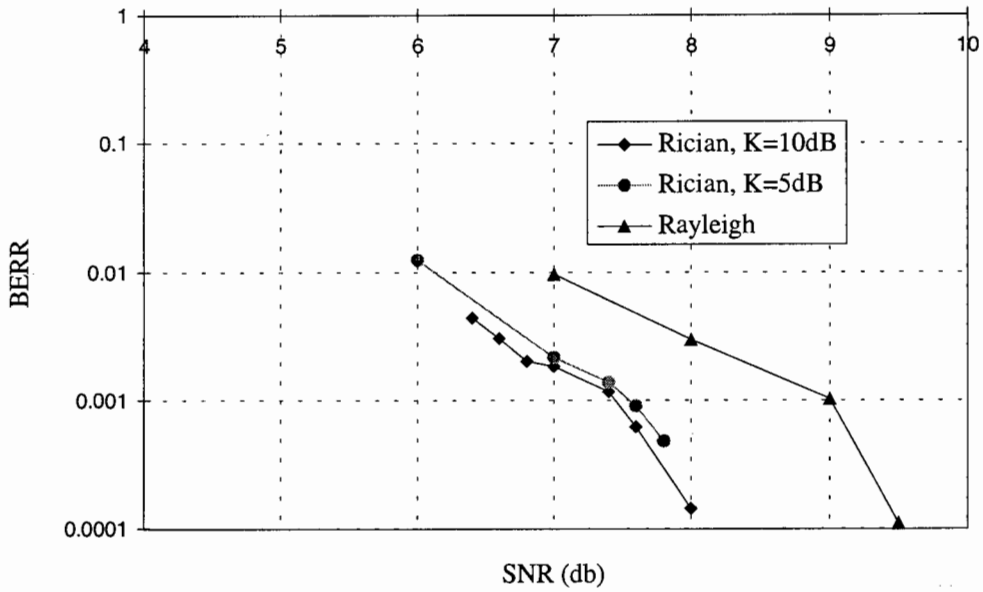

Figure 7.4: 4 Point constellation, no Reed Solomon coding

(K=10dB), the performance improvement was approximately 1dB.

Figure 7.6 shows the effect of using a (63,31) Reed-Solomon encoder. At a bit error rate of 10^{-3}, a further 1 dB is gained, at the price of a lower effective transmission rate.

The higher rate QAM-based constellations were also tested in Rician and Rayleigh fading conditions. Since these constellations make use of multiple amplitude levels, the use of channel state information was essential.

Figure 7.7 shows the performance of the 8 point constellation, with (63,31) Reed-Solomon encoding and interleaving between the trellis code and the Reed-Solomon encoding. The figure shows that the performance is acceptable in light Rician fading conditions without the use of CSI. With the Rician parameter K =5dB,

Under Rayleigh fading, the degradation caused by the uncompensated channel is severe and the use of Channel State Information is important. Figure 7.8 shows the increase in performance of the system under Rayleigh fading with the use of ideal channel state information.

Figure 7.9 and figure 7.10 show the bit error performance using the 20 point and 40 point constellations in Rician and Rayleigh fading. The same Reed-Solomon coding and interleaver settings were used. Once again, ideal channel state information was assumed.

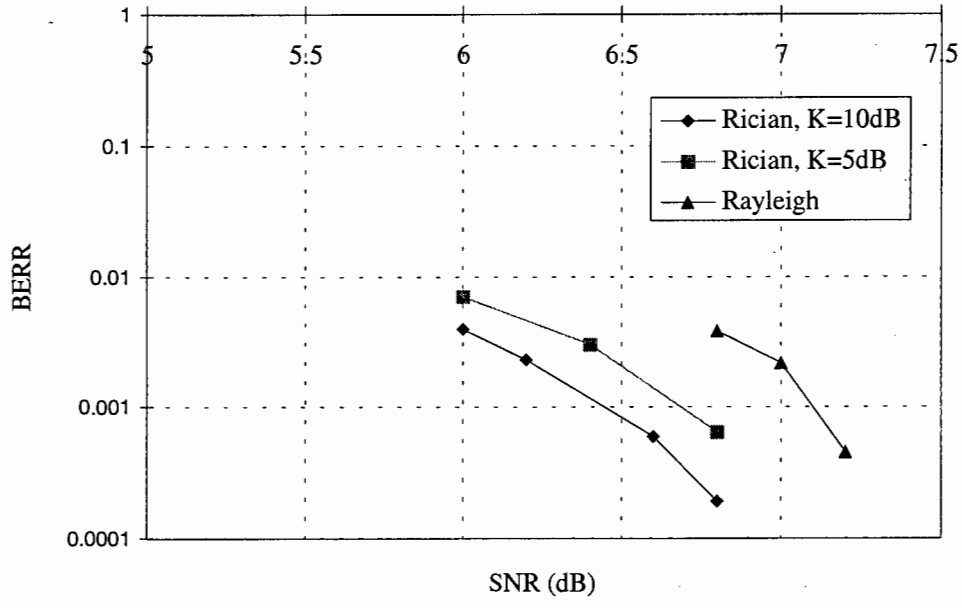

Figure 7.5: With (63,47) Reed-Solomon Encoding

The assumption of ideal channel state information will lead to the results being a bit optimistic. The insertion of some form of pilot tone in order to provide the CSI will also lead to additional degradation not shown here.

7.4 Summary

The simulations show the wide range of channel conditions over which good performance may be achieved using the proposed coding and modulation system.

The actual effectiveness of such a system will depend on the design parameters of the specific implementation. For example, suppose a system is to be designed to operate from a minimum elevation angle of 0 degrees above the horizon. If the SNR when the satellite is on the horizon is, say, 8 dB, the power during an overhead pass will be almost 20dB (depending on the altitude over the satellite). This would allow transmission at 8400 (.875 bps/Hz) or 9600 in the Rayleigh fading conditions at low elevations and up to 48kHz in the AWGN conditions when the satellite is directly overhead.

If, however, the maximum angle of elevation of the satellite is low, the difference in the

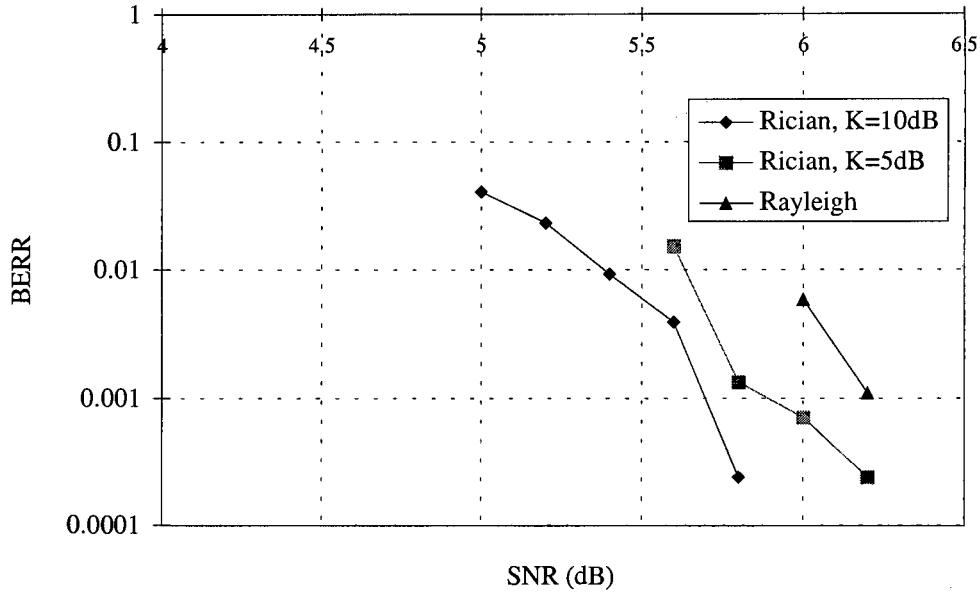

Figure 7.6: With (63,31) Reed-Solomon Encoding

received power is much lower (see section 3.3). This would limit the ability to use higher data rates.

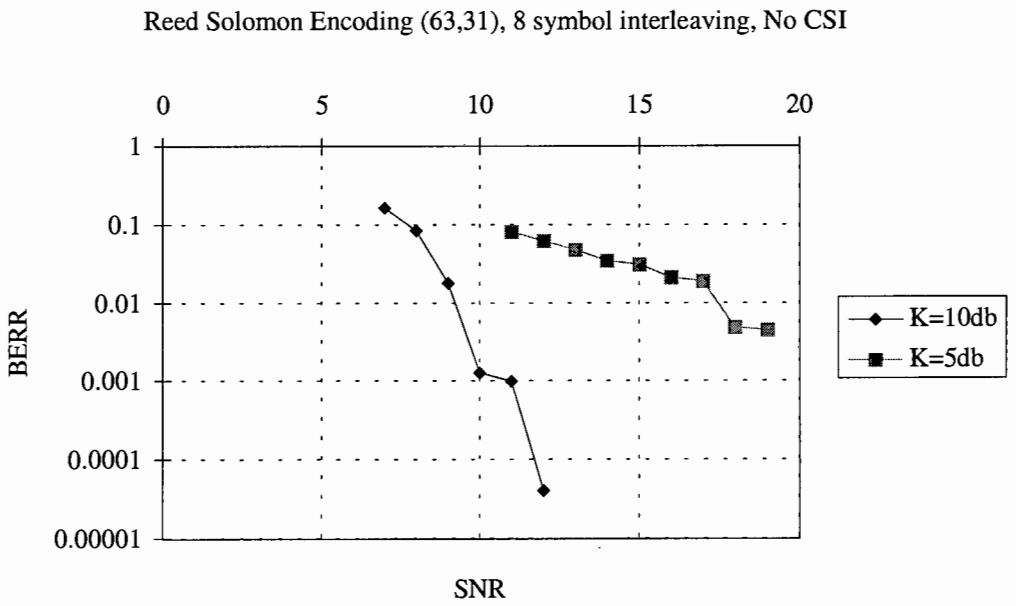

Figure 7.7: 8 point constellation in Rician fading

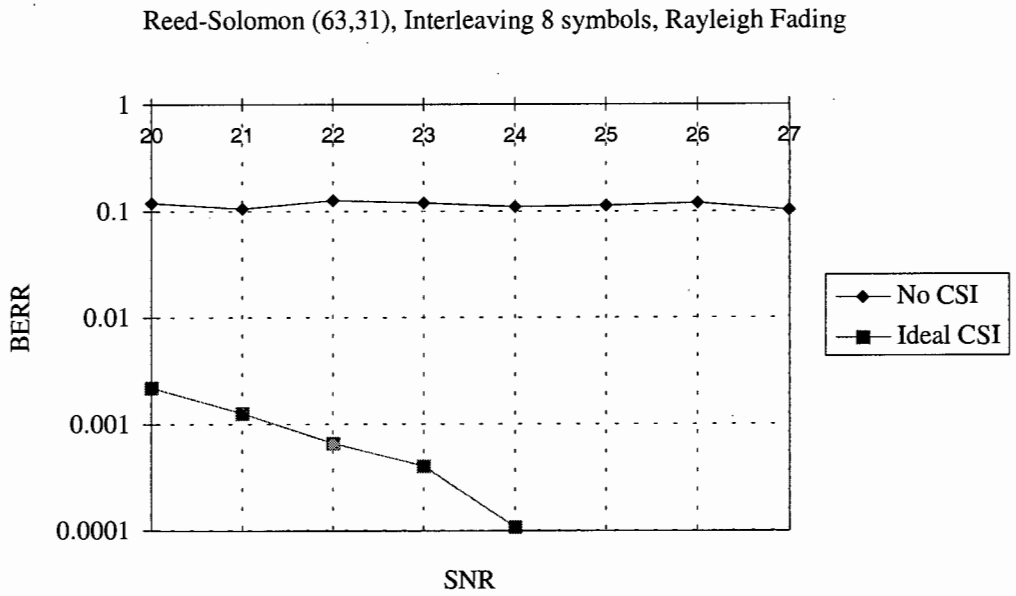

Figure 7.8: 8 point constellation with, and without, CSI

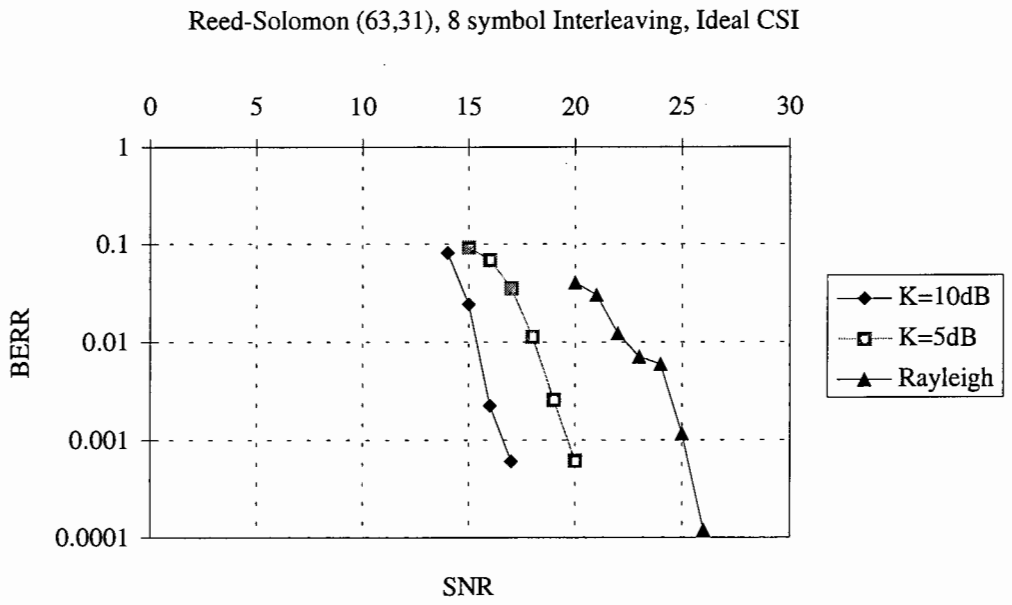

Figure 7.9: 20 point constellation in Rician and Rayleigh Fading

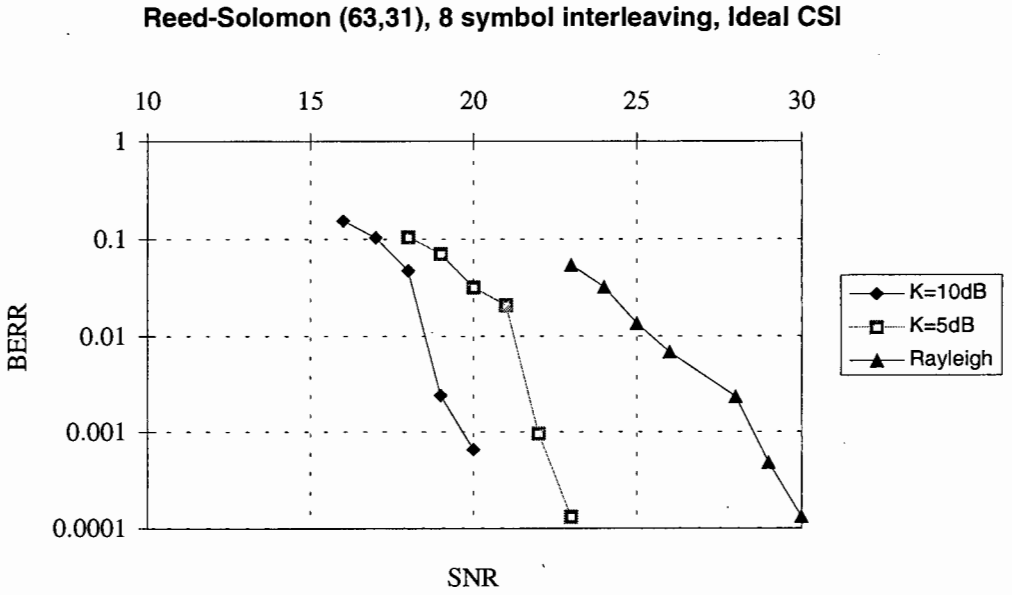

Figure 7.10: 40 point constellation in Rician and Rayleigh Fading

Chapter 8

Conclusions

A system was proposed that is able to provide high data rates when channel conditions allow, while providing reliable data transmission in poor conditions.

The following design objectives were achieved:

- The system provides a wide range of modes of transmission that give different data rates. The coding/modulation modes best suited to a particular implementation of the system may be chosen, resulting in a fairly flexible system.
- The complexity of the system was kept relatively low by utilising the same convolutional coder / viterbi decoder for all constellation sets used.

A number of drawbacks and, or, limitations are inherent in the system. These include:

- Synchronisation of transmitter/receiver data rate: In order to make use of the higher data rates, the transmitter and receiver must agree on a common data rate. This makes it very difficult to broadcast information, since all the receiving stations would need to negotiate a common data rate.
- Linear Mode Operation: The system as proposed needs to be operated in a linear mode. While the solid state amplifiers typically used in this type of satellite do have a linear mode (unlike the Travelling Wave Tube (TWT) Amplifiers), this requires some backoff of the satellite amplifier output power.

8.1 Further Research

Possible areas of future research are:

- Design of Trellis Codes for Fading Conditions: The trellis codes used were not specifically designed for fading conditions. Overall performance might be increased by using codes designed specifically for fading as opposed to the AWGN codes used. This would be especially important for mobile communications systems.
- Impact of interchannel Interference: The impact of interchannel interference was not explicity tested. This factor will lead to some additional performance degradation.
- Turbo Trellis Codes: Trellis Turbo Codes [22] offer an alternative to the single convolutional coder / Viterbi decoder used in thesis thesis. These codes could probably be used in place of the suggested system should the coding gain and complexity be favourable.
- Use near amplifier saturation: The system as discussed needs to be used in a linear mode of the Solid State Power Amplifiers. Compensation and Equalisation of the nonlinear channel arising from operating these amplifiers near saturation, may allow for more power efficient operation.

Topics not included int this thesis, such as the design of the decision feedback system to modify the code rate, also need further research.

Bibliography

[1] Dr W. Sun, M.N. Sweeting, and A. da Silva Curiel, "Leo satellite constellations for regional communications," *Surrey Satellite Technology Ltd, Centre for Satellite Engineering Research, University of Surrey, Guildford, Surrey, GU2 5XH, United Kingdom*, 1996.

[2] M. Sweeting, "Network of low cost small satellites for monitoring and mitigation of natural disasters," *Surrey Satellite Technology Ltd, Centre for Satellite Engineering Research, University of Surrey, Guildford, Surrey, GU2 5XH, United Kingdom*, 1996.

[3] F. Vatalaro, G. E. Corazza, C. Caini, and C. Ferrarelli, "Analysis of leo, meo, and geo global mobile satellite systems in the presence of interference and fading," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 291–300, Feb. 1995.

[4] S. G. Glisic, J. J. Talvitie, T. Kumpumaki, M. Latva-aho, J. H. Iinatti, and T. J. Poutanen, "Design study for a CDMA-based LEO satellite network: Downlink system level parameters," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 1796–1807, Dec. 1996.

[5] L. Wood, "Network performance of non-geostationary constellations equipped with intersatellite links," Master's thesis, University of Surrey, Sept. 1995.

[6] Dr W. Sun, M.N. Sweeting, and M.S.Hodgart, "Investigation of MSK and FSK modulation schemes for single channel communication system using DSP techniques on-board low earth orbit microsatellite.," *Surrey Satellite Technology Ltd, Centre for Satellite Engineering Research, University of Surrey, Guildford, Surrey, GU2 5XH, United Kingdom*, 1995.

[7] W.J.Vogel and J. Goldhirsh, "Multipath fading at L band for low elevation angle, land mobile satellite scenarios," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 197–204, Feb. 1995.

[8] C. Loo, "Measurements and models of a land mobile satellite channel and their applictaions to msk signals," *IEEE Transactions on Vehicular Technology*, vol. VT-35, pp. 114–121, Aug. 1987.

[9] J. Massey, "Coding and modulation in digital communications," *1974 International Zurich Seminar on Digital Communciations*, Mar. 1974.

[10] G. Ungerboeck, "A leech lattice modem," *IEEE Transactions on Information Theory*, vol. IT-28, pp. 56–67, Jan. 1982.

[11] C.E.Shannon, "Communication in the presence of noise," *Proc. IRE*, vol. 37, pp. 10–21, Jan. 1949.

[12] G. Welti and S. Lee, "Digital transmission with coherent four-dimensional modulation," *IEEE Transactions on Information Theory*, vol. IT-20, pp. 397–402, July 1949.

[13] S.G.Wilson, H.A.Sleeper, and N.K.Srinath, "Four dimensional modulation and coding: An alternative to frequency reuse," *Proc. ICC'84*, pp. 919–923, May 1984.

[14] E. Biglieri and M. Elia, "Multidimensional modulation and coding for band-limited digital channels," *IEEE Transactions on Information Theory*, vol. IT-34, pp. 397–402, July 1988.

[15] L. F. Wei, "Trellis-coded modulation with multidimensional constellations," *IEEE Transactions on Information Theory*, vol. IT-33, pp. 483–501, July 1987.

[16] G. R. Lang and F. M. Longstaff, "A leech lattice modem," *IEEE Journal on Selected Areas in Communications*, vol. 7, pp. 968–972, Aug. 1989.

[17] S. A. Tretter, "An eight dimensional 64-state trellis code for transmitting 4 bits per 2-d symbol," *IEEE Journal on selected Areas in Communications*, vol. 7, pp. 1392–1394, Dec. 1989.

[18] S.-L. Su and J.-M. Wu, "Combination of BCM and TCM with 90^o, 180^o and 270^o phase rotational invariance," *IEEE Transactions on Communications*, vol. 45, pp. 800–808, July 1997.

[19] E. Biglieri, *Introduction to Trellis-Coded Modulation with Applications*, chapter 10. New York: Macmillan Publishing Company, 1991.

[20] Y.-M. Gu and C. W. Lee, "Set partitioning for multilevel codes on a rayleigh-fading channel," *IEEE Transactions on Communications*, vol. 46, pp. 161–163, Feb. 1998.

[21] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes," *Proceedings ICC'93*, pp. 1064–1070, May 1993.

[22] P. Robertson and T. Wörz, "Bandwidth-efficient turbo trellis-coded modulation using punctured component codes," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 206–218, Feb. 1998.

[23] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. SIAM*, vol. 8, pp. 300–304, 1960.

[24] G. D. Forney, *Concatenated Codes*. MIT Press,Cambridge MA, 1966.

[25] S. A. Goldsmith, "Adaptive coded modulation for fading channels," *IEEE Transactions on Communications*, vol. 46, pp. 595–602, May 1998.

[26] S. A. Goldsmith, "Variable-rate, variable-power MQAM for fading channels," *IEEE Transactions on Communications*, vol. 45, pp. 595–602, Oct. 1997.

[27] I. Ali, N. Al-Dhahir, and J. E. Hershey, "Doppler characterisation for leo satellites," *IEEE Transactions on Communications*, vol. 46, pp. 309–313, Mar. 1998.

[28] E. Biglieri, *Introduction to Trellis-Coded Modulation with Applications*. New York: Macmillan Publishing Company, 1991.

[29] S. G. Wilson, *Digital Modulation and Coding*. New Jersey: Prentice Hall, Inc., 1996.

Appendix A

Diffuse Fading Spectrum

The Rician Fading model, described in section 3.5 assumes a certain power spectral density for the diffuse signal. The derivation of the diffuse signal's power spectral density, taken from [28] is as follows.

We assume that, for simplicity, all the diffuse waves are travelling in the horizontal plane.

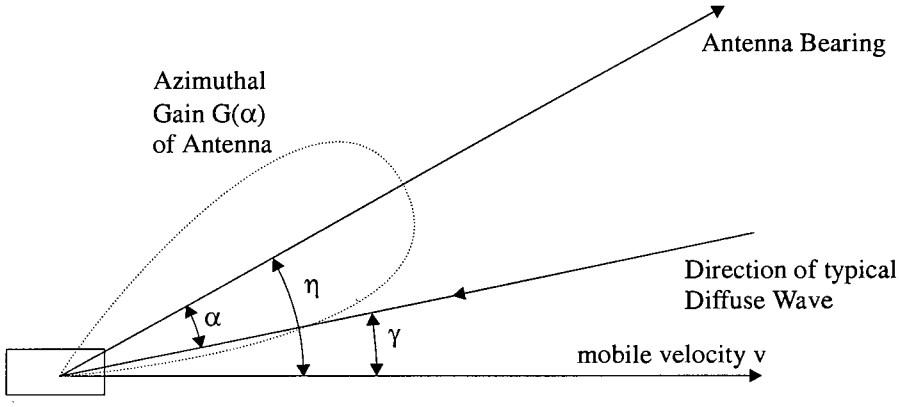

Figure A.1: Overhead View of Mobile

The diffuse signal arrives at the antenna at an angle γ to the motion of the vehicle. The doppler shift of the signal is given as

$$f_d = \frac{v}{\lambda}\cos(\gamma) \tag{A.1}$$

The power contribution with an angle $d\gamma$ is proportional to

$$p(\gamma)G(\gamma - \eta)d\gamma \tag{A.2}$$

where $p(\gamma)$ is the angular probability density function of the diffuse waves and G is the

angular gain of the antenna.

If $S(f)$ is the power spectral density of the diffuse signal at S(f), the power contribution at frequency f is

$$S(f)\,|df| \tag{A.3}$$

The doppler frequency is given as

$$f = f_d \cos\gamma \tag{A.4}$$

where f_d is the maximum doppler shift.

For any f_d, the angles γ and $-\gamma$ can result in the same frequency. The contribution of power over differential frequency $d\gamma$ that results in doppler frequency f is proportional to

$$k\,[p(\gamma)G(\gamma-\eta)+p(-\gamma)G(-\gamma-\eta)]\,d\gamma \tag{A.5}$$

Equating equations A.3 and A.5 gives

$$S(f) = \frac{k\,[p(\gamma)G(\gamma-\eta)+p(-\gamma)G(-\gamma-\eta)]\,d\gamma}{|df|} \tag{A.6}$$

where K is a constant. Now,

$$\begin{aligned} |df| &= f_d|-\sin\gamma|d\gamma && \text{(A.7)}\\ &= f_d\sqrt{1-\left(\frac{f}{f_d}\right)^2}\,d\gamma && \text{(A.8)}\\ &= \sqrt{f_d^2-f^2}d\gamma && \text{(A.9)}\\ & && \text{(A.10)} \end{aligned}$$

where

$$\begin{aligned} \sin\gamma &= \sqrt{1-\cos^2\gamma}\\ &= \sqrt{1-\left(\frac{f}{f_d}\right)} \end{aligned}$$

Thus, $S(f)$ becomes

$$S(f) = \frac{k\,[p(\gamma)G(\gamma-\eta)+p(-\gamma)G(-\gamma-\eta)]}{\sqrt{f_d^2-f^2}} \tag{A.11}$$

The angular density function $p(\gamma)$ is approximated as being uniform

$$p(\gamma) = 1/2\pi \tag{A.12}$$

The power spectral density is then given as

$$S(f) = \frac{kG(\gamma - \eta)}{\pi\sqrt{f_d^2 - f^2}} \tag{A.13}$$

For the simple case of an omni-directional antenna, this power spectral density will reduce to

$$S(f) = \frac{1}{\pi\sqrt{1 - (f/f_d)^2}}, \qquad -f_d \leq f \leq f_d \tag{A.14}$$

This is shown in figure A.2.

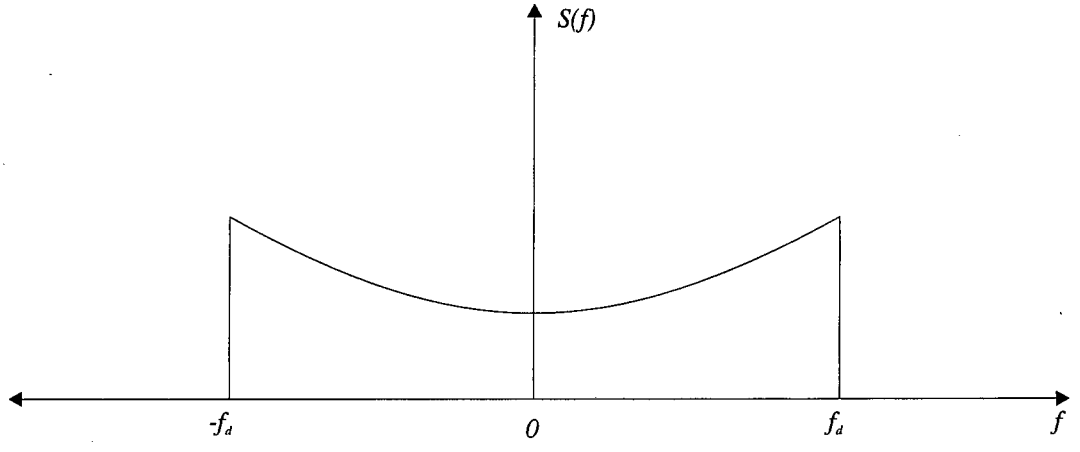

Figure A.2: Power Spectrum of Doppler Filter

Appendix B

Simulation Code

The code for the trellis coding and the simulation was written in a mixture of Matlab and C++ (for the Viterbi Decoder). Due to the close relationship between the constellation set and the soft decision Viterbi decoding it was necessary to use a customised Viterbi Decoder.

B.1 Main Modulation and Simulation Routine

The routine below codes and modulates the input data stream before adding the simulated channel.

The received signal is then demodulated, deinterleaved and decoded. Finally a bit error rate is calculated to assess performance.

```
simspeed = menu('Choose Coding method','2 bits per symbol',
                '4 bits per symbol','6 bits per symbol');
simtype = menu('Choose Simulation','with Reed Solomon Coding',
                'no additional coding');
intltype = menu('Choose Interleaving Strategy','After RS Code',
               'After Trellis Code','Both','None');

if (simspeed==1)
   constl = c8pt;
   numpts = 16;
elseif simspeed==2
   constl = c20pt;
   numpts = 36;
elseif simspeed==3
   constl = c40pt;
   numpts = 64;
end
```

```
b = genbin(84000-2);% odd number of symbols due to certain initial
                    % constraints due to needing to be multiples of
                    % certain values(interleaving size, code rate)
                    % These have largely been corrected so that they
                    % are automatically dealt with, but the odd sizes
                    % live on.
bb = [ 0 0 b]; % prefix the two bytes that will be lost due
% to initial phase uncertainty

% Now add the optional reed-solomon encoding
if (simtype == 1)
   c = encode(bb,15,7,'rs')';
else
   c = bb;
end

%size(c)

% first interleaver
if (intltype==1 | intltype==3)
   [d xtra] = bintlve(c,1,100);
else
   d = c;
   xtra = 0;
end

% 8 dimensional trellis coding
%keyboard
d = [d zeros(1,840)];
e = enc8dim(d,simspeed-1,6);
% second interleaver
if (intltype==2 | intltype==3)
   [f xtra2] = bintlve(e',1,100);
   f = f';
else
   f = e;
   xtra2 = 0;
end
% transmit data
g = qamconst(numpts,f);
gref = qamconst(numpts,const1);
[g nv] = normtotalpower(g,1,0);
gref = normtotalpower(gref,nv,1);

% Add Rician Channel
g = satchan(g,snr,K);

% deinterleave data
if (intltype==2 | intltype==3)
```

```
   h = bdeintlve(g,1,100);
   h = h(1:length(h)-xtra2)';
else
   h = g;
end
% decode
i = dec8dim(h,gref,simspeed-1,6);
i = i(1:length(c));


%deinterleave before RS
if (intltype==1 | intltype==3)
   j = bdeintlve(i,1,100);
   j = j(1:length(j)-xtra);
else
   j = i;
end

% Reed Solomon Decoding
if (simtype == 1)
   out = decode(j,15,7,'rs')';
else
   out = j;
end
out = out(3:length(out));
% Print Bit error rate
brr = berr(out,b);
disp(brr);


function [y] = satchan(input,snr,K)
% simulates satellite channel
% input = input modulated points
% snr = ratio of signal to required additional gaussian
% noise(dB)
% K = Rician fading parameter - ratio of LOS component
%     to specular component(dB)
% K. Butchart 1998

global sampler;
global samplerate;
global fading;
global doppler;

I = rcflt(real(input),.5); % raised cosine .5 rolloff
Q = rcflt(imag(input),.5);

trans = I+i*Q;              % ready to transmit
clear I;
clear Q;
[fb fa] = butter(2,2/samplerate); % sixth order butterworth
```

```
                                        filter
trans = filtfilt(fb,fa,trans);  % tranmission filter
trans = normtotalpower(trans,1,0);
trans = rician(trans,K,doppler); % simulate rician channel
trans = awgn(trans,snr);  % add gaussian noise
trans = filtfilt(fb,fa,trans);
sampler = [samplerate 1];
fading = sample(fading);
y = sample(trans);
```

B.2 Viterbi Decoder

The following code was written in C++ because of speed considerations. The memory handling is fairly messy since, due to the large amount of iterations, it was attempted to allocate all the memory needed by the routines before the actual algorithm got started. This translated into more difficult to read code.

```
#ifndef __VITERBI_H__
#define __VITERBI_H__
#include "cmatrix.h"
#include "mat.h"
#include <iostream.h>
#ifndef NO_MEX
#include <mex.h>
#endif
class viterbi {
public:
  viterbi(){StateList=0;UncodedBits=0;}
  ~viterbi();

  matrix& encode(matrix&unencoded,int startstate);

private:
  matrix* tmpmat;
  matrix* stateseq;

  matrix* mt4;
  matrix* mt4r;
  matrix* mt64;
  matrix* mt256;
  matrix* ovec;

  static int tt[16][32];
  // helper function
  // must allocate the space in d2 before starting
```

```
  void calculateDists(cmatrix&f,cmatrix&g,matrix&output);
  void calctypes(matrix&metrics,matrix&output,matrix*csi);
  void preparetypemem();
  void cleanuptypemem();

public:
  matrix &softdecode(cmatrix& dists,cmatrix& sets,int startstate,
                     int trellisdepth,matrix* csi);
  matrix *getstateseq() {return stateseq;}

  // get free distance - calculates the free distance of an en-
coder in bits
  int getfreedistance();

  matrix& getTransmittedSymbols(int startstate);
  // functions to take generator polynomials and produce convolu-
tional encoders
  int GetOutputSymbol(matrix& state,submatrix&input,matrix&gen);
  int GetNextState(matrix& state,submatrix&input);
  void GenCoder(int mem,int inbits,matrix& gen);
  void GenParallel(int numpar);


  // read and write viterbi decoder information
  friend ostream& operator<<(ostream&os,viterbi& v);
  friend istream& operator>>(istream&is,viterbi& v);

  // debugging information

  void disp();
  matrix& GetStateList() {if (!StateList)
                    throw MatrixException(MatrixException::MATRIX_EMPTY);
                  else
                    return *StateList;}

  void SetInputSymbols(matrix& insymbols) {
    inputsymbols = insymbols;
  }

  void SetOutputSymbols(matrix& outsymbols) {
    outputsymbols = outsymbols;
  }

  void SetStateSeq(matrix&sseq) {
    StateSeq = sseq;
  }

  void SetStateOutput(matrix&sout) {
    StateOutput = sout;
  }
```

```
  void SetMemSize(int ms) {
    memsize = ms;
  }

  void SetNumStates(int ns) {
    NumStates = ns;
  }

  void SetInBits(int inbits) {
    InBits = inbits;
  }

  void SetOutBits(int outbits) {
    OutBits = outbits;
  }


private:
  matrix StateSeq,StateOutput;
  int InBits,OutBits,NumStates;
  int memsize;
  char out[300];
  int minrdist;

  matrix inputsymbols,outputsymbols;
  int CurrentState;

  matrix* StateList;
  matrix* UncodedBits;

  matrix StateDistances;
  matrix PreviousState;

  int softsymb(matrix& dists,int startstate,int numsteps,int firsttime);
  int gout(int i,int j,int numpar,int InBits,int OutBits);
  int gstate(int i,int j,int numpar,int InBits,int OutBits);
};

#endif


#include "viterbi.h"
#include <complex.h>
#include <iostream.h>

#define SQR(x) ((x)*(x))

// definitions of types for easier reading in later procedures
#define _AA 1
#define _AC 2
#define _AB 3
#define _AD 4
```

```
#define _CA 5
#define _CC 6
#define _CB 7
#define _CD 8
#define _BA 9
#define _BC 10
#define _BB 11
#define _BD 12
#define _DA 13
#define _DC 14
#define _DB 15
#define _DD 16

#define MIN(x,y) ((x<y)? x : y)
#define PAIR(x,y) ((x-1)*8+y)


// map of which combinations of sub-types finally constitute
// the 16 types ie. _AA_CC or _BB_DD are part of the first type
// as well as the rest of the pairs of the first row of the
// array tt
int viterbi::tt[16][32]={
{1,1,6,6,11,11,16,16,1,11,6,16,11,1,16,6,3,3,
  8,8,9,9,14,14,3,9,8,14,9,3,14,8},
{1,6,6,11,11,16,16,1,1,16,6,1,11,6,16,11,3,8,
  8,9,9,14,14,3,3,14,8,3,9,8,14,9},
{1,3,6,8,11,9,16,14,1,9,6,14,11,3,16,8,3,1,8,
  6,9,11,14,16,3,11,8,16,9,1,14,6},
{1,8,6,9,11,14,16,3,1,14,6,3,11,8,16,9,3,6,8,
  11,9,16,14,1,3,16,8,1,9,6,14,11},
{2,2,7,7,12,12,13,13,2,12,7,13,12,2,13,7,4,4,
  5,5,10,10,15,15,4,10,5,15,10,4,15,5},
{2,7,7,12,12,13,13,2,2,13,7,2,12,7,13,12,4,5,
  5,10,10,15,15,4,4,15,5,4,10,5,15,10},
{2,4,7,5,12,10,13,15,2,10,7,15,12,4,13,5,4,2,
  5,7,10,12,15,13,4,12,5,13,10,2,15,7},
{2,5,7,10,12,15,13,4,2,15,7,4,12,5,13,10,4,7,
  5,12,10,13,15,2,4,13,5,2,10,7,15,12},
{1,2,6,7,11,12,16,13,1,12,6,13,11,2,16,7,3,4,
  8,5,9,10,14,15,3,10,8,15,9,4,14,5},
{1,7,6,12,11,13,16,2,1,13,6,2,11,7,16,12,3,5,
  8,10,9,15,14,4,3,15,8,4,9,5,14,10},
{1,4,6,5,11,10,16,15,1,10,6,15,11,4,16,5,3,2,
  8,7,9,12,14,13,3,12,8,13,9,2,14,7},
{1,5,6,10,11,15,16,4,1,15,6,4,11,5,16,10,3,7,
  8,12,9,13,14,2,3,13,8,2,9,7,14,12},
{2,1,7,6,12,11,13,16,2,11,7,16,12,1,13,6,4,3,
  5,8,10,9,15,14,4,9,5,14,10,3,15,8},
{2,6,7,11,12,16,13,1,2,16,7,1,12,6,13,11,4,8,
  5,9,10,14,15,3,4,14,5,3,10,8,15,9},
{2,3,7,8,12,9,13,14,2,9,7,14,12,3,13,8,4,1,5,
```

```
  6,10,11,15,16,4,11,5,16,10,1,15,6},
{2,8,7,9,12,14,13,3,2,14,7,3,12,8,13,9,4,6,5,
  11,10,16,15,1,4,16,5,1,10,6,15,11}};

// ------------------------------------------------------
// IO routines - used mainly with some matlab routines for
// verification purposes
// write our viterbi decoder to stream
ostream& operator<<(ostream&os,viterbi&v) {
  os << v.InBits << endl << v.OutBits << endl;
  os << v.StateSeq << v.StateOutput
    << v.inputsymbols << v.outputsymbols;

  return os;
}
// read our viterbi decoder from a stream
istream& operator>>(istream&is,viterbi&v) {
  is >> v.InBits >> v.OutBits;
  is >> v.StateSeq >> v.StateOutput
    >> v.inputsymbols >> v.outputsymbols;
  v.NumStates = v.StateSeq.GetRows();

  return is;
}
// ------------------------------------------------------
  viterbi::~viterbi() {
   if (StateList)
       delete StateList;
    if (UncodedBits)
       delete UncodedBits;
    StateSeq.dispose();
    StateOutput.dispose();
  }

  matrix& viterbi::encode(matrix&unencoded,int startstate) {
    int sz;
    int i,j,z;

    matrix* OutStream;
    sz = unencoded.GetLength();
    OutStream = new matrix(1,sz*OutBits/InBits);
    CurrentState = startstate;

    int outindex = 1;
    matrix insymb(1,InBits);

    for (i = 1;i<=sz;i+=InBits,outindex+=OutBits) {
      // read first symbol
      for (j=0;j<InBits;j++)
        insymb[ind(1,j+1)] = unencoded[ind(1,i+j)];
        // convert it into a symbol index
```

```
    int InSymbol = insymb.getdecval()+1;
    // Output the output symbol corresponding to current
                                          state and input symbol
    int osr = (int)StateOutput[ind(CurrentState,InSymbol)];
    for (j=0;j<OutBits;j++)
      (*OutStream)[ind(1,outindex+j)] = outputsymbols[ind(osr,j+1)];
    //Change to the next state
    CurrentState = (int)StateSeq[ind(CurrentState,InSymbol)];
  }
  return *OutStream;
}
// ---------------------------------------------------------------------
// memory allocation - to prevent memory from being allocated and
//                  deallocated every step of the routines  although
//                  the code does read better the other way
// ---------------------------------------------------------------------
void viterbi::preparetypemem() {
  mt4 = new matrix(16,2);
  mt4r = new matrix(8,2);
  mt64 = new matrix(64,1);
  mt256 = new matrix(16,16);
}

void viterbi::cleanuptypemem() {
  mt4->dispose();
  mt4r->dispose();
  mt64->dispose();
  mt256->dispose();
}
// ---------------------------------------------------------------------
// CalculateDists
// calculates the euclidean distance from each received point to the
// 2D constellation points
// ---------------------------------------------------------------------
void viterbi::calculateDists(cmatrix&f,cmatrix&g,matrix&output) {
  int i,j,k,l,m;
  double dst;
  matrix& dtmp = *tmpmat;
  for (i=1;i<=f.GetRows();i++) {  // for each input array point
    complex c = f[i];
    for (j = 1;j<=g.GetRows();j++)
      for (k = 1;k<=g.GetColumns();k++) {
        dst = SQR(real(c) - real(g[ind(j,k)])) +
          SQR(imag(c) - imag(g[ind(j,k)]));
        dtmp[ind(j,k)] = dst;
      }
    for (l = 1;l<=g.GetRows();l++) {
      output[ind(i,l)] = 1e10;
      for (m=1;m<=g.GetColumns();m++)
        if (dtmp[ind(l,m)]<output[ind(i,l)]) {
          output[ind(i,l)] = dtmp[ind(l,m)];
```

```
          output[ind(i,1+g.GetRows())] = m;
        }
    }

  }
}

// ------------------------------------------------------------------
// calctypes
// uses the distances calculated by calc dists and calculates the new
// distances to each of the 16 8-dimensional types
// ------------------------------------------------------------------
void viterbi::calctypes(matrix &metrics,matrix&outvec,matrix*csi) {
  matrix& met4 = *mt4;
  matrix& met4r = *mt4r;
  matrix& met64 = *mt64;
  matrix& met256 = *mt256;
  int num8dsyms = metrics.GetRows()/4;
  int i,j,k,m;

  for (k = 1;k<=num8dsyms;k++) {
    for (i=1;i<=4;i++) {
      for (j=1;j<=4;j++) {
        if (csi) {
          matrix&c = *csi;
          met4[ind((i-1)*4+j,1)] = metrics[ind(1+(k-1)*4,i)]*
                                   (c[ind((k-1)*4+1,1)])+
                                   metrics[ind(2+(k-1)*4,j)]*
                                   (c[ind((k-1)*4+2,1)]);
          met4[ind((i-1)*4+j,2)] = metrics[ind(3+(k-1)*4,i)]*
                                   (c[ind((k-1)*4+3,1)])+
                                   metrics[ind(4+(k-1)*4,j)]*
                                   (c[ind((k-1)*4+4,1)]);
        } else {
          met4[ind((i-1)*4+j,1)] = metrics[ind(1+(k-1)*4,i)]+
                                   metrics[ind(2+(k-1)*4,j)];
          met4[ind((i-1)*4+j,2)] = metrics[ind(3+(k-1)*4,i)]+
                                   metrics[ind(4+(k-1)*4,j)];
        }
      }
    }
    for (i=1;i<=16;i++)
      for (j=1,m=1;m<=16;j+=2,m++) {
        met256[ind(i,m)] = met4[ind(tt[i-1][j-1] ,1)] +
                           met4[ind(tt[i-1][j],2)];
      }


    double mval,mpos;
    for (i=1;i<=16;i++) {
      mval = 1e16;
```

```
  for (j=1;j<=16;j++) {
    if (met256[ind(i,j)] < mval) {
      mval = met256[ind(i,j)];
      mpos = j;
    }
  }
outvec[ind(k,i+16)] = mpos;
}


 for (i=1;i<=2;i++) {
  met4r[ind(1,i)] = MIN( met4[ind(_AA,i)] , met4[ind(_BB,i)] );
  met4r[ind(2,i)] = MIN( met4[ind(_CC,i)] , met4[ind(_DD,i)] );
  met4r[ind(3,i)] = MIN( met4[ind(_AB,i)] , met4[ind(_BA,i)] );
  met4r[ind(4,i)] = MIN( met4[ind(_CD,i)] , met4[ind(_DC,i)] );
  met4r[ind(5,i)] = MIN( met4[ind(_AC,i)] , met4[ind(_BD,i)] );
  met4r[ind(6,i)] = MIN( met4[ind(_CB,i)] , met4[ind(_DA,i)] );
  met4r[ind(7,i)] = MIN( met4[ind(_AD,i)] , met4[ind(_BC,i)] );
  met4r[ind(8,i)] = MIN( met4[ind(_CA,i)] , met4[ind(_DB,i)] );
}

 for (i=1;i<=8;i++)
  for (j=1;j<=8;j++)
   met64[ind(PAIR(i,j),1)] = met4r[ind(i,1)] + met4r[ind(j,2)];


 outvec[ind(k,1)] = MIN( MIN(met64[PAIR(1,1)],met64[PAIR(2,2)] ) ,
      MIN(met64[PAIR(3,3)],met64[PAIR(4,4)] ) );
 outvec[ind(k,2)] = MIN( MIN(met64[PAIR(1,2)],met64[PAIR(2,1)] ) ,
      MIN(met64[PAIR(3,4)],met64[PAIR(4,3)] ) );
 outvec[ind(k,3)] = MIN( MIN(met64[PAIR(1,3)],met64[PAIR(2,4)] ) ,
      MIN(met64[PAIR(3,1)],met64[PAIR(4,2)] ) );
 outvec[ind(k,4)] = MIN( MIN(met64[PAIR(1,4)],met64[PAIR(2,3)] ) ,
      MIN(met64[PAIR(3,2)],met64[PAIR(4,1)] ) );
 outvec[ind(k,5)] = MIN( MIN(met64[PAIR(5,5)],met64[PAIR(6,6)] ) ,
      MIN(met64[PAIR(7,7)],met64[PAIR(8,8)] ) );
 outvec[ind(k,6)] = MIN( MIN(met64[PAIR(5,6)],met64[PAIR(6,5)] ) ,
      MIN(met64[PAIR(7,8)],met64[PAIR(8,7)] ) );
 outvec[ind(k,7)] = MIN( MIN(met64[PAIR(5,7)],met64[PAIR(6,8)] ) ,
      MIN(met64[PAIR(7,5)],met64[PAIR(8,6)] ) );
 outvec[ind(k,8)] = MIN( MIN(met64[PAIR(5,8)],met64[PAIR(6,7)] ) ,
      MIN(met64[PAIR(7,6)],met64[PAIR(8,5)] ) );
 outvec[ind(k,9)] = MIN( MIN(met64[PAIR(1,5)],met64[PAIR(2,6)] ) ,
      MIN(met64[PAIR(3,7)],met64[PAIR(4,8)] ) );
 outvec[ind(k,10)] = MIN( MIN(met64[PAIR(1,6)],met64[PAIR(2,5)] ) ,
      MIN(met64[PAIR(3,8)],met64[PAIR(4,7)] ) );
 outvec[ind(k,11)] = MIN( MIN(met64[PAIR(1,7)],met64[PAIR(2,8)] ) ,
      MIN(met64[PAIR(3,5)],met64[PAIR(4,6)] ) );
 outvec[ind(k,12)] = MIN( MIN(met64[PAIR(1,8)],met64[PAIR(2,7)] ) ,
      MIN(met64[PAIR(3,6)],met64[PAIR(4,5)] ) );
 outvec[ind(k,13)] = MIN( MIN(met64[PAIR(5,1)],met64[PAIR(6,2)] ) ,
```

```
          MIN(met64[PAIR(7,3)],met64[PAIR(8,4)] ) );
      outvec[ind(k,14)] = MIN( MIN(met64[PAIR(5,2)],met64[PAIR(6,1)] ) ,
          MIN(met64[PAIR(7,4)],met64[PAIR(8,3)] ) );
      outvec[ind(k,15)] = MIN( MIN(met64[PAIR(5,3)],met64[PAIR(6,4)] ) ,
          MIN(met64[PAIR(7,1)],met64[PAIR(8,2)] ) );
      outvec[ind(k,16)] = MIN( MIN(met64[PAIR(5,4)],met64[PAIR(6,3)] ) ,
          MIN(met64[PAIR(7,2)],met64[PAIR(8,1)] ) );

     // finally preserve which point was closest in each subset
     for (i=1;i<=4;i++) {
       for (j=1;j<=4;j++) {
         outvec[ind(k,32 + (i-1)*4 + j)] = metrics[ind(i+(k-1)*4,4+j)];
       }
     }
    }

}

// ------------------------------------------------------------------
// softdecode
// main entry routine for viterbi decoder
// The routine calls the functions calcdists and calctypes to work out
// the metrics in the 8-dimensional trellis. If another trellis is to
// be used, these must be modified
// cmatrix dists - complex matrix of distances to each constellation
//                 point
// cmatrix sets - details the sets into which the 2D constellation has
//                 been divided. In this case the matrix will have four
//                 rows and will
// trellis depth - number of (in this case) 8-dimensional symbols to
//                 truncate the viterbi algorithm at
// csi           - channel state information

matrix &viterbi::softdecode(cmatrix& dists,cmatrix& sets,
                            int startstate,int trellisdepth,
                                           matrix * csi) {
  CurrentState = startstate; // temporary copy of the current state
  matrix * csis;
  int NumSteps = trellisdepth;
  int NumSyms = dists.GetRows()/4;
  matrix &detectedbits =*new matrix(1,(NumSyms-NumSteps)*InBits);
  StateList = new matrix(NumSyms-NumSteps,5);
  int i,j,k,ii,jj;
  int nstate = CurrentState;
  int pnt;
  int firsttime;

  StateDistances = *new matrix(NumSteps+1,NumStates);
  PreviousState = *new matrix(NumSteps,NumStates);

  preparetypemem();
```

```
int pow2inbits = (int)pow(2,InBits);
subcmatrix sbc(dists,ind(1,1),NumSteps*4,1);
matrix * csisub;
if (csi) {
  csisub = new matrix(NumSteps*4,1);
  for (i=1;i<=NumSteps*4;i++)
    (*csisub)[ind(i,1)] = (*csi)[ind(i,1)];
} else
  csisub = 0;


matrix& distances=*new matrix(NumSteps*4,sets.GetRows()*2);
matrix& ovec = *new matrix(NumSteps,48);
matrix& dis = *new matrix(4,sets.GetRows()*2);
if (csi)
  csis = new matrix(4,1);
else
  csis = 0;
matrix& ty = *new matrix(1,48);
tmpmat = new matrix(sets.GetRows(),sets.GetColumns());
calculateDists(sbc,sets,distances);
calctypes(distances,ovec,csisub);


int si = 16;
cmatrix& tmpc = *new cmatrix(4,1);
firsttime = 1;

distances.dispose();
int outind;
for (i=1,outind=1;i<=(NumSyms-NumSteps)*4;i+=4,outind++) {
  int insym,outsym;
  // save the current state
  // use the viterbi decoder to get the next state number
  nstate = softsymb(ovec,CurrentState,NumSteps,firsttime);

  firsttime=0;

  // determine which input symbol gave rise to this state transition
  for (j = 1;j<=pow2inbits;j++) {
    if (nstate == StateSeq[ind(CurrentState,j)]) {
      insym = j;
      outsym = StateOutput[ind(CurrentState,insym)];
    }
  }

  pnt = ovec[ind(1,si+outsym)];
  int pt1 = tt[outsym-1][(pnt-1)*2];
  int pt2 = tt[outsym-1][(pnt-1)*2+1];
  int p1,p2,p3,p4;
```

```
      p1 = floor((pt1-1)/4)+1;
      p2 = ((pt1-1)%4)+1;
      p3 = floor((pt2-1)/4)+1;
      p4 = ((pt2-1)%4)+1;
      (*StateList)[ind(outind,1)] = pnt;

      (*StateList)[ind(outind,2)] = ovec[ind(1,32+p1)];
      (*StateList)[ind(outind,3)] = ovec[ind(1,36+p2)];
      (*StateList)[ind(outind,4)] = ovec[ind(1,40+p3)];
      (*StateList)[ind(outind,5)] = ovec[ind(1,44+p4)];

      // output the input symbol
      for (k=1;k<=InBits;k++)
        detectedbits[ind(1,(outind-1)*InBits+k)] =
                                          inputsymbols[ind(insym,k)];

      // move to the next state and remove bits we have just processed
      CurrentState = nstate;

      // copy new values across
      tmpc[ind(1,1)] = dists[ind((NumSteps)*4+i,1)];
      tmpc[ind(2,1)] = dists[ind((NumSteps)*4+i+1,1)];
      tmpc[ind(3,1)] = dists[ind((NumSteps)*4+i+2,1)];
      tmpc[ind(4,1)] = dists[ind((NumSteps)*4+i+3,1)];

      calculateDists(tmpc,sets,dis);
      // roll up all the type-distances up one
      for (ii=1;ii<NumSteps;ii++)
        for (jj=1;jj<=ovec.GetColumns();jj++)
          ovec[ind(ii,jj)] = ovec[ind(ii+1,jj)];

      (*csis)[ind(1,1)] = (*csi)[ind(NumSteps*4+i,1)];
      (*csis)[ind(2,1)] = (*csi)[ind(NumSteps*4+i+1,1)];
      (*csis)[ind(3,1)] = (*csi)[ind(NumSteps*4+i+2,1)];
      (*csis)[ind(4,1)] = (*csi)[ind(NumSteps*4+i+3,1)];

      calctypes(dis,ty,csis);
      for (ii=1;ii<=ty.GetColumns();ii++)
        ovec[ind(ovec.GetRows(),ii)] =ty[ii];

      // now move the channel state information on
      // (if, of course, we are using it)
    }

  //release all the memory we have used
  StateDistances.dispose();
  PreviousState.dispose();
  dis.dispose();
  ty.dispose();
  csis->dispose();
  tmpc.dispose();
```

```
ovec.dispose();
tmpmat->dispose();
distances.dispose();
delete csisub;

cleanuptypemem();
// return the result
return detectedbits;
}

// ---------------------------------------------------------------------
// softsymb
// main decoder algorithm for Viterbi decoder
// This routine performs each step of the viterbi algorithm to the
// truncation depth. It then produces an output and the next call to
// this algorithm is stepped on one symbol.
// To save time, the distances already calculated on the previous call
// are preserved on subsequent calls, so only new symbols need to be
// considered

int viterbi::softsymb(matrix& dists,int startstate,int numsteps,
                                                    int firsttime) {
  int currstateno = startstate;

  // Only the inital state is valid so we assign it a distance of
  // 0 and the other states -1
  int i,j,k,l;
  int pow2inbits = (int)pow(2,InBits);
  //double rj=1e15;

  if (firsttime) {
    for (i=1;i<=StateDistances.GetRows();i++)
      for (j=1;j<=StateDistances.GetColumns();j++)
        StateDistances[ind(i,j)] = -1;

    StateDistances[ind(1,currstateno)] = 0;
  } else {
    for (j=1;j<=StateDistances.GetColumns();j++) {
      StateDistances[ind(1,j)] = StateDistances[ind(2,j)];
    }

    for (i=2;i<=StateDistances.GetRows();i++)
      for (j=1;j<=StateDistances.GetColumns();j++)
        StateDistances[ind(i,j)] = -1;
  }


  for (i = 1;i<=numsteps;i++) {
    // read next symbol

    // go through each state
```

```
  for (j = 1;j<=NumStates;j++) {
    // get the distance to this state
    double CurrentStateMetric = StateDistances[ind(i,j)];
    // if we haven't got to it yet
    if (CurrentStateMetric == -1){
      // do nothing
    }else {
      // for each input symbol
      for (k = 1;k<=pow2inbits;k++) {
        // get the state we would move to
        int NextState = (int)StateSeq[ind(j,k)];
        double Metric = dists[ind(i,(int)StateOutput[ind(j,k)])];
        Metric += CurrentStateMetric;  // add it to distance sofar

        if (StateDistances[ind(i+1,NextState)]== -1) {
        // if this is first visit to state, assign the distance
          StateDistances[ind(i+1,NextState)] = Metric;
          PreviousState[ind(i,NextState)] = j;
        } else {
          // otherwise assign it if it is shorter than any other
          // path to this state
          if (Metric < StateDistances[ind(i+1,NextState)]) {
            StateDistances[ind(i+1,NextState)] = Metric;
            PreviousState[ind(i,NextState)] = j;
          }
        }
      }
    }
  }
}

// now that we have repeated the sequence for the required number
// of steps, we are ready to decide on a minimum path and output
// the first state we move to
// get minimum distance


double mind = 100000;
int bestfinalstate;

for (l = 1;l<=StateDistances.GetColumns();l++) {
  if (StateDistances[ind(numsteps+1,l)]<mind) {
    mind = StateDistances[ind(numsteps+1,l)];
    bestfinalstate=l;
  }
}

// back track to the original state using the information stored
// in PreviousState
for (l = numsteps;l>1;l--) {
  bestfinalstate= (int)PreviousState[ind(l,bestfinalstate)];
```

```
    }
    return bestfinalstate;
  }
  // ---------------------------------------------------------------------

  matrix& viterbi::getTransmittedSymbols(int startstate) {
    if (!StateList)
      throw MatrixException(MatrixException::MATRIX_EMPTY);

    return *StateList;
  }

  // function to take generator polynomial and produce convolu-
tional encoder

  int viterbi::GetOutputSymbol(matrix& state,submatrix&input,matrix&gen) {
    int i,j;
    int symbol = 0;
    // create one big matrix
    int numOut = gen.GetRows();
    matrix& inst = input.mergeright(state);
    matrix instate(numOut,gen.GetColumns(),(double)0);

    // copy the right bits
    for (i=1;i<=gen.GetRows();i++)
      for (j=1;j<=gen.GetColumns();j++)
        if (gen[ind(i,j)])
          instate[ind(i,j)] = inst[j];


    matrix& sum = instate.sumRows();
    sum%=2;
    return sum.getdecval()+1;
  }

  int viterbi::GetNextState(matrix& state,submatrix&input) {
    int memsize = state.GetColumns();
    int insize = input.GetColumns();
    int keep = memsize-insize;
    submatrix kp(state,1,keep);
    matrix tmp(1,memsize,(double)0);
    submatrix tmpnew(tmp,1,insize);
    submatrix tmpkp(tmp,insize+1,keep);
    tmpnew = input;
    tmpkp = kp;

    return tmp.getdecval()+1;
  }


  void viterbi::GenCoder(int mem,int inbits,matrix& gen) {
```

```
  int i,j;
  memsize = mem;
  OutBits = gen.GetRows();
  InBits=inbits;
  matrix state(1,memsize,(double)0);
  StateOutput = *new matrix((int)pow(2,memsize),(int)pow(2,InBits));
  StateSeq = *new matrix((int)pow(2,memsize),(int)pow(2,InBits));

  for (i = 1;i<=pow(2,memsize);i++) {
    state.makebinary(i-1);
    for (j=1;j<=pow(2,InBits);j++) {
      submatrix insymbol(inputsymbols,ind(j,1),1,InBits);
      StateOutput[ind(i,j)] = GetOutputSymbol(state,insymbol,gen);
      StateSeq[ind(i,j)] = GetNextState(state,insymbol);
    }

  }

  NumStates = (int)pow(2,memsize);
}



void viterbi::GenParallel(int numpar) {
  int i,j;
  NumStates = StateSeq.GetRows();
  NumStates = (int)pow(NumStates,numpar);
  int numinsymbols = (int)pow(2,InBits*numpar);
  int numoutsymbols = (int)pow(2,OutBits*numpar);
  matrix tmpseq(NumStates,numinsymbols);
  matrix tmpout(NumStates,numinsymbols);

  for (i=1;i<=NumStates;i++)
    for (j=1;j<=numinsymbols;j++) {
      int nxtstate = gstate(i,j,numpar,InBits,OutBits);
      int out = gout(i,j,numpar,InBits,OutBits);
      tmpseq[ind(i,j)] = nxtstate;
      tmpout[ind(i,j)] = out;
    }

  StateSeq = tmpseq;
  StateOutput = tmpout;
  tmpseq.dispose();
  tmpout.dispose();
  InBits  = InBits*numpar;
  OutBits = OutBits*numpar;
  inputsymbols = *new matrix(numinsymbols,InBits);
  outputsymbols = *new matrix(numoutsymbols,OutBits);
  matrix tmp(range(0,numinsymbols-1));
  inputsymbols.makebinary(tmp);
```

```
  matrix tmp2(range(0,numoutsymbols-1));
  outputsymbols.makebinary(tmp2);
}



// debugging information

void viterbi::disp() {
  cout << "Inbits: " << InBits << "OutBits: " << OutBits << endl;
  cout << "StateSeq" << endl;
  StateSeq.disp();
  cout << "StateOutput" << endl;
  StateOutput.disp();
  cout << "inputsymbols" << endl;
  inputsymbols.disp();
  cout << "outputsymbols" << endl;
  outputsymbols.disp();
}


int viterbi::gout(int i,int j,int numpar,int InBits,int OutBits) {
  matrix state(1,numpar);
  matrix in(1,numpar);
  int x;

  int pow2in = (int)pow(2,InBits);
  int pow2out = (int)pow(2,OutBits);
  int ns = StateSeq.GetRows();

  i = i-1;
  j = j-1;

  for (x = 1;x<=numpar;x++) {
    int a = (i%ns)+1 ;
    int b = (j%pow2in)+1;
    state[x] = a;
    in[x] = b;
    i /= ns;
    j /= pow2in;
  }

  int sum=0;
  int mult =1;
  for (x=1;x<=numpar;x++,mult*=pow2out)
    sum+=(int) (StateOutput[ind((int)state[x],(int)in[x])]-1)* mult;
  sum++;

  return sum;
}
```

```
int viterbi::gstate(int i,int j,int numpar,int InBits,int OutBits) {
  matrix state(1,numpar);
  matrix in(1,numpar);
  int x;

  int pow2in = (int)pow(2,InBits);
  int pow2out = (int)pow(2,OutBits);
  int ns = StateSeq.GetRows();

  i = i-1;
  j = j-1;

  for (x = 1;x<=numpar;x++) {
    int a = (i%ns)+1 ;
    int b = (j%pow2in)+1;
    state[x] = a;
    in[x] = b;
    i /= ns;
    j /= pow2in;
  }

  int sum=0;
  int mult =1;
  for (x=1;x<=numpar;x++,mult*=ns)
    sum+= (int)(StateSeq[ind((int)state[x],(int)in[x])]-1)* mult;
  sum++;

  return sum;
}
```