TAPR Error Control Coding Seminar

Phil Karn
KA9Q

March 1995

Overview of Seminar

This seminar will:

– Present the theoretical limits to communication over noisy channels
– Give you a feel for what coding can and cannot do
– Describe the basic kinds of codes and the more important examples of each kind along with their applications

This seminar will *not*:

Make you an expert. Dozens of textbooks and thousands of journal articles and PhD theses have been written on this subject!

Disclaimer: I'm not an expert either. But teaching is a good way to learn a subject...

Limits to Communication

The fundamental limits on communication over noisy channels were published by Claude Shannon in 1948 in the classic paper, "A Mathematical Theory of Communication" (Bell System Technical Journal, July 1948)

Shannon showed that error-free communication is possible on a noisy channel provided that the data rate is less than the *channel capacity*

Channel capacity depends on the bandwidth and the S/N ratio

Channel Capacity Formula

Channel capacity is given by the formula:

$$C = B \log_2 (1 + \frac{S}{N})$$

where:

C = channel capacity, bits/sec

B = channel bandwidth, Hz

S = signal power, W

N = noise power, W

Comments on Shannon

Random data bits are assumed. If not, compress them. This is the subject of another branch of information theory called *source coding* (not the subject of this seminar).

The formula applies *only* to the additive white gaussian noise (AWGN) channel. Fading, interference, distortion, etc are not considered

Channel capacity is a theoretical limit only; it describes the best that can possibly be done with any code and modulation method. Beating this limit is like building a perpetual motion machine

You can send data faster than capacity, but it won't be error free. If you add an error-correcting code to deal with the errors, the corrected data rate will always be less than the channel capacity, *but you may still be ahead of where you started*

Shannon does not actually show how to build a practical system that reaches the limit. This is what everybody has been working on since.

Another Look at Capacity

In the standard form of the Shannon equation

$$C = B\log_2(1+\frac{S}{N})$$

the noise power is

$$N = N_0 B$$

where B is the bandwidth and N_0 is the noise spectral density in watts/Hz.

Doubling the bandwidth doesn't quite double the capacity because the noise also doubles. So what's the net effect?

Capacity Restated

Let's define

E_b = energy per bit, joules (watt-seconds)
R = relative data rate, bits/sec/Hz (so-called "spectral efficiency")

Then after some (omitted) algebra, we find that for a system to obey the Shannon limit, it must satisfy the inequality

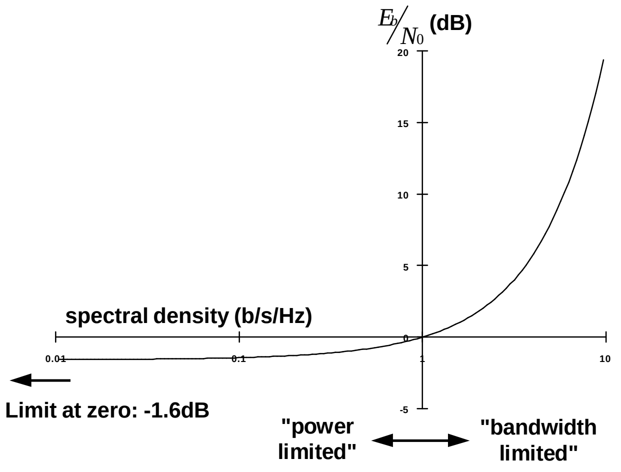
$$\frac{E_b}{N_0} \geq \frac{2^R - 1}{R}$$

The ratio E_b/N_0, pronounced "ebb-know", describes the overall power efficiency of the system. It is usually expressed in dB.

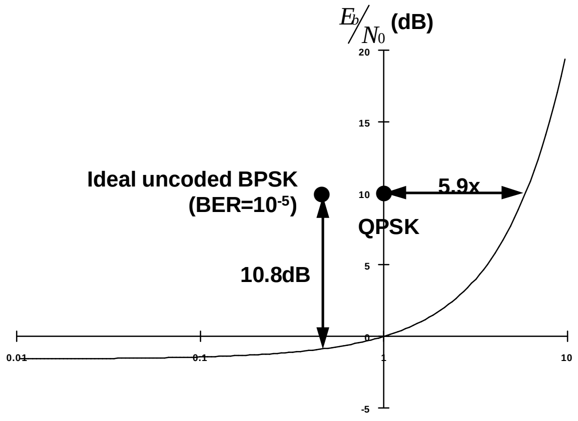
Large R (many bits/sec/Hz) requires a large E_b/N_0; conversely, small R allows a small E_b/N_0. I.e.,

☞Bandwidth can be traded off against power☜

Minimum E_b/N_0 vs Bandwidth



Potential Coding Gains



What is Error Control Coding?

Error Control Coding is the addition of carefully designed redundancy to a data stream before modulation to help the receiver detect and/or correct errors introduced by the transmission process

When used for error correction, this is called Forward Error Correction (FEC)

The redundant information is computed according to an algorithm that is known to the receiver ahead of time

The encoded information consists of *symbols*, as distinguished from the original data bits

In some systems, ECC and modulation/demodulation are very closely intertwined

Why Use Coding?

Reduce power requirements on power-limited channels,
e.g., radio links (emphasis of this seminar)

> Besides the obvious benefits of lower power (cheaper RF hardware, longer battery life, reduced biohazards, etc), interference resistance is increased. This is important on shared channels

Reduce bandwidth on bandwidth-limited channels,
e.g., telephone circuits

Shape signal spectrum

> Usually to remove low frequency components (Manchester coding on Ethernet, EFM on Compact Disc)

Power-Limited Coding

Power-reduction coding generally uses extra bandwidth.
Example:

- –Uncoded QPSK requires E_b/N_0 =10dB for a bit error rate of about 10^{-5}.
- –QPSK with rate 1/2, K=7 Viterbi-decoded convolutional coding requires E_b/N_0 =4dB for the same BER, only 25% as much power for the same data rate.

But there's a catch: twice as much bandwidth is required

Bandwidth-Limited Coding

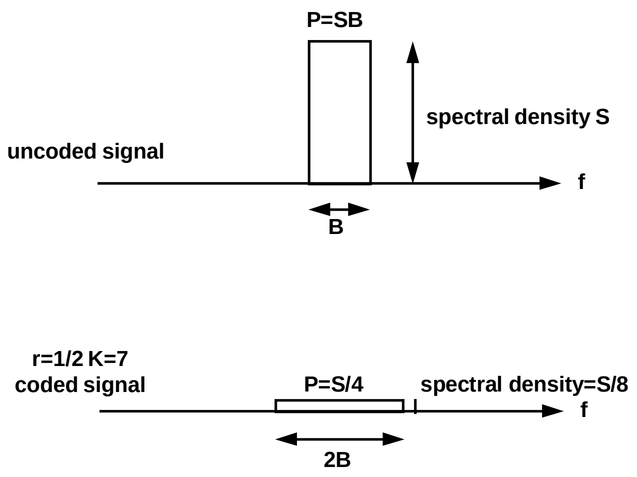
The bandwidth "penalty" of coding can be relieved by combining it with higher-level modulation schemes (more bits/sec/Hz).

This may seem counterproductive, but it works. Done right, the S/N increase required by the higher-level modulation method is more than compensated for by the S/N decrease allowed by coding.

The most common example is *trellis coding,* a generalization of convolutional coding, which is extremely popular in telephone modems (V.32, V.32*bis*, V.34)

Coding as Spread Spectrum

Coding for a power-limited channel is really a form of spread spectrum:



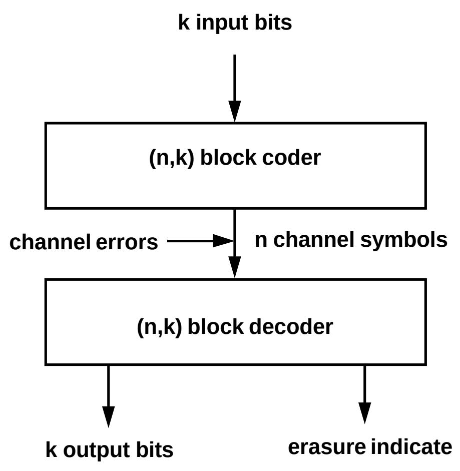
ECC Types

There are two major types of error control codes:

Block codes
- CRC
- Orthogonal (Reed-Mueller, etc)
- Hamming
- Golay
- BCH
- Reed-Solomon
- etc

Convolutional codes
- Sequentially decoded
- Viterbi decoded

Block Coding



Error Detection Coding

The lowly CRC is a block code used for error detection only. I.e., the decoder produces the k output bits only if there are no channel errors. If even a single error occurs, the decoder signals an erasure. A higher level ARQ (Automatic Request-Repeat) then retransmits the block

More sophisticated codes can correct some number of errors. Only if the number of errors exceeds the code's correction ability will it signal an erasure

Systematic Codes

Most block codes are *systematic*. That is, the k input bits appear as-is in the n output symbols. The additional n-k symbols are *parity* symbols

In a nonsystematic code, every output symbol is a function of more than one input data bit, i.e., the input does not appear imbedded in the output

Nonsystematic linear block codes have no particular advantage over systematic codes, so systematic block codes are generally used in practice. This lets decoders cut corners (at a price in lost performance)

Hamming Distance

The Hamming distance (or just "distance") between two equal-length bit strings is the number of places in which they disagree.

For example, these strings have distance 2:

```
0 0 1 1 0 1 0 0 1
1 0 0 1 0 1 0 0 1
↑   ↑
```

Distance Properties

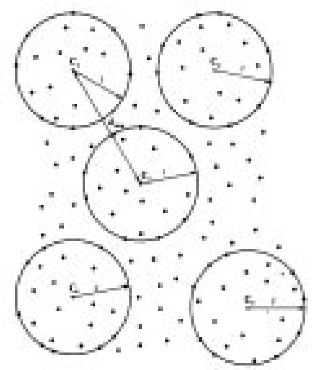
A (n,k) block code has 2^k possible inputs, so only 2^k of the possible 2^n encoded output blocks (codewords) are valid.

The code designer tries to maximize the distance between every pair of valid codewords as this maximizes the number of errors that can be detected and/or corrected. This implies a fairly uniform distribution of codewords

A code used only for error detection declares an erasure if the received codeword isn't valid

A code used for error correction finds the valid codeword nearest the one that was received and assumes that it was sent. I.e., it assumes that a few errors are more likely than a lot of errors

Distance Properties - 2


A representation of code words as centers of spheres of radius $t = \lfloor (d_{min} - 1)/2 \rfloor$

Distance Properties - 3

All other things being equal, bigger code blocks provide better performance for the same relative overhead - at the expense of added delay

Tradeoffs exist between the erasure rate and the undetected error rate. For example, the designer might specify that if a received codeword is "too far" from a valid codeword, an erasure is declared even though the error could have been corrected. This reduces the undetected error rate but increases the erasure rate

In most block codes, the distribution of errors in a block is irrelevant; only the total number of errors in a block matters. This makes large block codes good for burst errors, e.g., against pulsed interference

Resistance to error bursts can be further increased by *interleaving* (more later)

Orthogonal Signalling

Signal sets with elements that can be independently detected without mutual interference are *orthogonal*

The number of elements in the set, *m,* is usually a power of 2

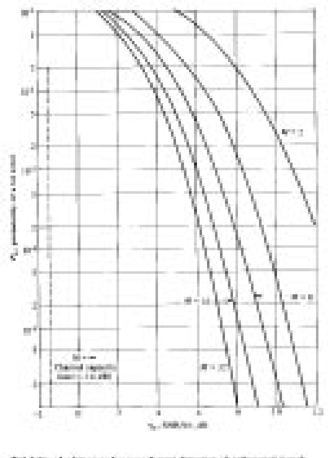
This can be seen as a family of block codes of the form (m,$\log_2$m), e.g., (2,1); (4,2); (8,3); etc

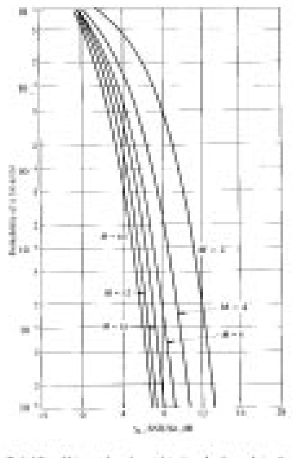
Ordinary binary (2-ary) FSK is an orthogonal signal set with 2 elements

Given enough bandwidth, orthogonal signal sets can be constructed with as many elements as you like. As m increases, performance for both coherent and noncoherent detection (slowly) approaches the Shannon limit of E_b/N_0= -1.6dB.

Besides sine waves, other orthogonalizing functions include Walsh functions, the rows of a Hademard matrix. These are *Reed-Mueller* codes

Performance


Probability of a bit error for noncoherent detection of orthogonal signals


Probability of bit error for coherent detection of orthogonal signals

Uses of Orthogonal Signals

Orthogonal codes were among the first to be analyzed according to Shannon

The big problem with orthogonal signalling is the slow E_b/N_0 improvement with bandwidth. But it found use as an error control code in the first NASA deep space probes to use digital transmission; a (32,6) *biorthogonal* code was used on Mariner 6 and 9 at Mars

Orthogonal codes are now used in spread-spectrum systems, usually in combination with other, more powerful coding (e.g., with Viterbi-decoded convolutional coding in the Qualcomm CDMA reverse link)

The benefit of orthogonal coding is its performance with noncoherent demodulation. This is an important advantage in severe multipath fading where there is no stable carrier phase reference (HF and EME modem designers take note)

Some Block Codes

CRC (Cyclic Redundancy Check)

– Extremely popular for error detection
– Adaptable to varying block sizes with a constant number of parity symbols n-k
– Distance properties poor for large n - size should be limited

Hamming

– One of the first block codes; not particularly strong
– A family of codes with $n = 2^m - 1$, k=n-m; m>2.
  E.g., (7,4) (15,11) (31,26) (63,57) (127,120)
– Minimum distance between valid codewords = dmin = 3
– Error correcting ability =(dmin-1)/2 = 1
– Now used mainly to correct errors in RAMs (e.g., Microsat)

Some More Block Codes

Golay

- A (23,12) block code with dmin=7 (corrects 3 errors)
- Often extended to (24,12) with dmin=8
- Fairly easy to decode
- Better than Hamming, but not the best
- Used in Kantronics' G-TOR

BCH

- A large class of codes usually with block length 2^m-1 (m>0)
- Can be binary or nonbinary
- Used in AMPS cellular

Reed-Solomon Codes

Reed-Solomon codes are a subclass of non-binary BCH codes. That is, instead of individual binary digits (bits), R-S codes operate on symbol alphabets of more than two values. These alphabet sizes are usually powers of 2 and are written GF(2^m)

A R-S code block size is one less than the alphabet size. E.g., a R-S code over GF(256) will have a block size of 255 symbols or 2040 bits. There is an "extended" R-S code with a block size equal to the alphabet size (256 symbols or 2048 bits for our example)

The number of data symbols within a R-S codeword can be chosen according to the desired error correcting ability. To correct up to E errors in a block, you need 2E parity symbols, with the rest being data. E.g., if you want to correct up to 10 errors in a R-S code over GF(256), we will need 20 parity symbols. This leaves 255-20 = 235 data symbols per block. This results in a (255,235) code over GF(256)

R-S codes can correct as many erasures (known errors) as parity symbols, ie., twice as many erasures as errors

R-S Codes - 2

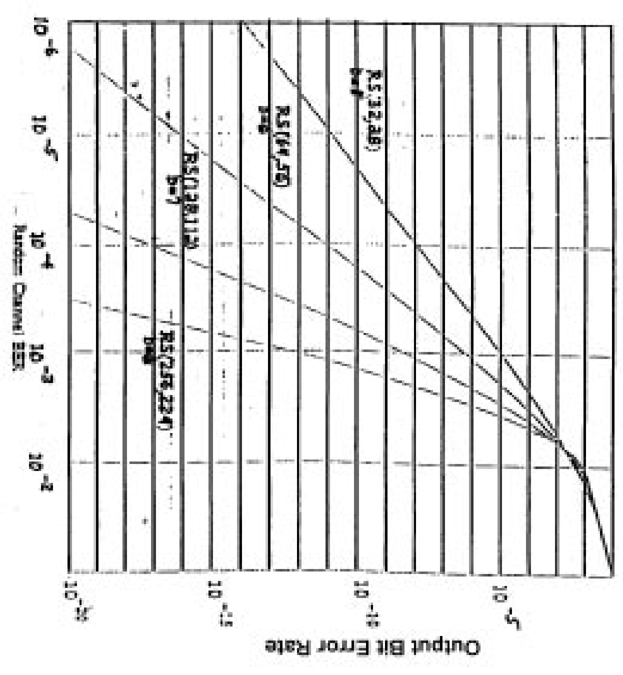
A Reed-Solomon code achieves the best performance possible for any linear code with the same block size. They are also relatively easy to implement at large block sizes (better performance)

This has made them extremely popular; the R-S code is now arguably the single most important block code used for error correction

Applications of R-S codes include digital audio (Compact Disc, DAT, etc), deep space communications (Voyager and Galileo, both in combination with convolutional coding), and the HAL Clover II HF modem

The multi-bit R-S symbol is a natural fit to an m-ary orthogonal modulation scheme. E.g., a GF(256) R-S coder could send each symbol as one of 256 orthogonal carriers in a 256-ary FSK system designed for HF channels with severe fading

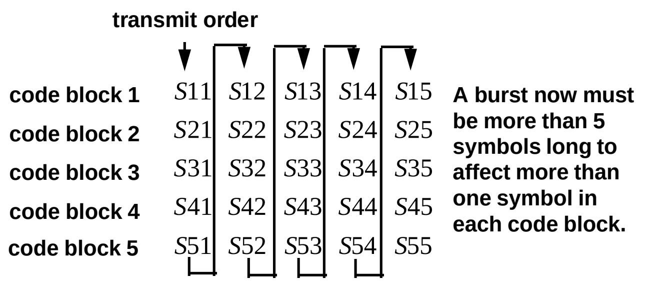
Performance of Reed-Solomon Codes of Rate 7/8 Over GF(2^b) (random error channel)



Interleaving

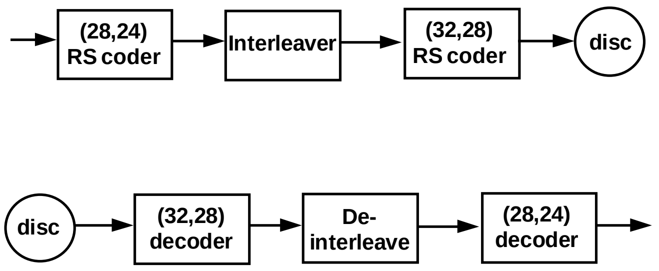
Interleaving is a technique that allows a designer to construct a big block code out of a smaller one. I.e., given a (n,k) block code, we can construct a (L*n,L*k) block code

A common way to do this is to allocate a block of memory as a 2-dimensional matrix. The encoded symbols of each block are written into memory in rows. Then they are read out and transmitted in columns. The receiver reverses the procedure, decoding each row in turn.



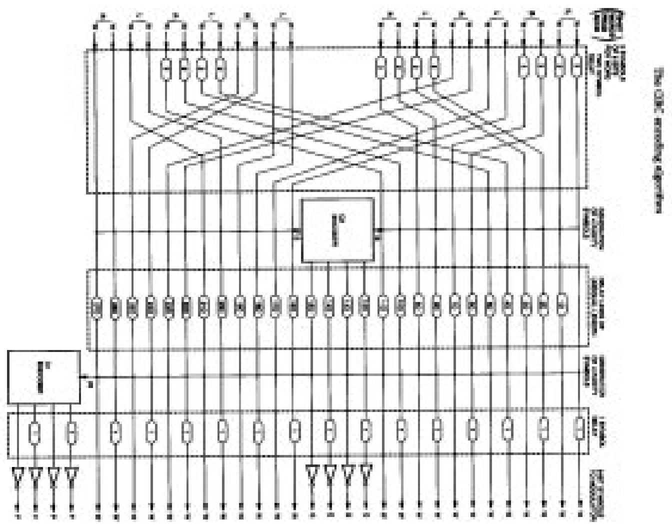
The Compact Disc

The Compact Disc uses two concatenated Reed-Solomon codes over GF(256) with an interleaver between them



(This is a simplified diagram. The EFM channel coding isn't shown, and there is additional interleaving before the first coder and after the second.)

CIRC (complete)



The CD - 2

By itself, the inner (32,28) RS code isn't very strong. More than (32-28)/2 =2 symbol errors to the decoder causes it to fail and flag an erasure on all 28 of its output symbols.

But the deinterleaver distributes these 28 erasures across 28 different blocks of the outer (28,24) RS code, one erasure per block. And because these are erasures, not errors, the outer code can correct up to 28-24=4 of them in a block. So the outer code easily fills in the missing symbols.

The concatenated system is so strong that it can completely correct an error burst of up to 3,874 bits - assuming no more bursts occur until the interleavers have flushed!

The two CD codes are examples of "shortened" R-S codes. An 8-bit symbol size would ordinarily imply a 255-byte block, but the encoder sets the unused symbols to zero and the decoder takes this into account.

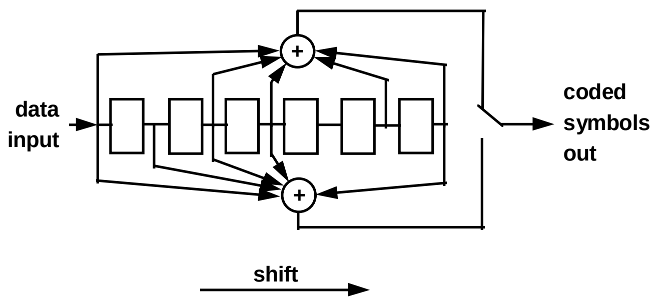
Convolutional Coding

Convolutional codes (also known as tree codes) operate on continuous streams of data, rather than on fixed size blocks

A convolutional encoder is extremely simple: a shift register, some XOR gates and a multiplexer:

r=1/2, K=7

NASA Standard code



Code Parameters

Convolutional codes are described by two parameters:

K = the *constraint length,* the length of the shift register+1

r = the "rate" of the code; the number of data bits per encoded channel symbol. Usually expressed as a rational fraction b/v. In most common codes, b=1. The most popular rates are 1/2, 1/3 and 1/4.

Coding gain increases directly with K and inversely with r; diminishing returns are reached

Convolutional Decoding

This is the fun part!

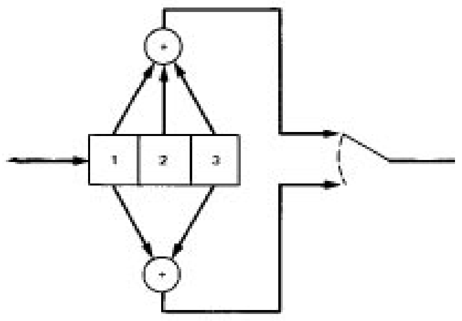
There are two main types of algorithms:

- Sequential decoding - complexity insensitive to K, so large values are generally used (e.g., 32)
- Maximum Likelihood (Viterbi) decoding - complexity increases as 2^k , so k>9 is rare (Galileo uses k=14!)

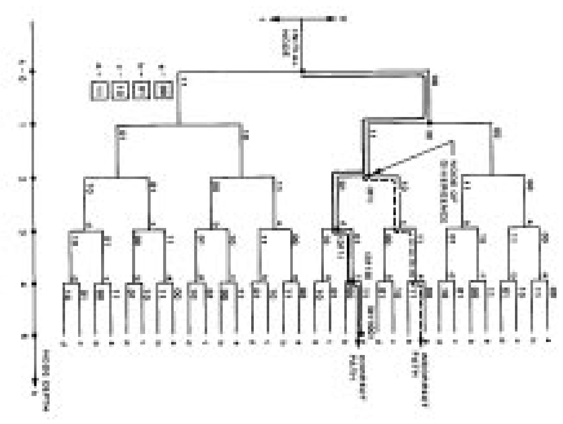
Both are relatively insensitive to r

Unlike block codes, nonsystematic convolutional codes outperform systematic codes, so nonsystematic codes are more often used

Example: r=1/2, K=3



Code Tree



Sequential Decoding

Sequential decoding looks for the path through the code tree that most closely matches the received sequence of symbols

Two sequential decoding algorithms are commonly used: Fano and stack. Though they look quite different, they accomplish the same thing

The decoder measures the “goodness” of the match. This is called the *metric.* As long as it continues to increase, it moves forward

If the metric decreases, the decoder assumes it may have made a mistake earlier due to noise. It backs up and sequentially examines alternate paths until it finds one whose metric increases again

The decoder may look at the “quality” of each received symbol and give it an appropriate weight in the total metric. This is called *soft decision* decoding and is a significant advantage of convolutional codes over block codes (2 dB on AWGN channels). Just a few bits of precision give almost all of this gain; 3 bit A/D sampling is common

Sequential Decoding Performance

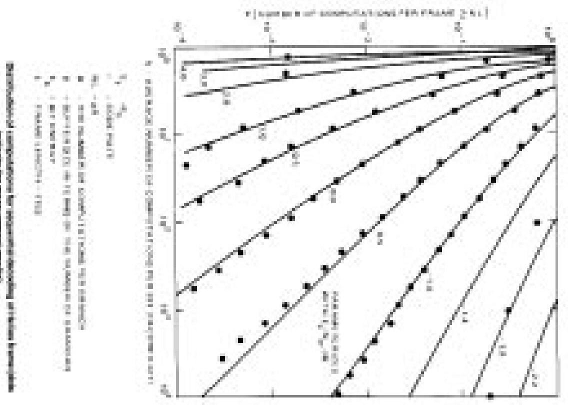
The performance of a sequential decoder is strongly dependent on the number and distribution of errors. Burst errors are particularly nasty as the decoding time is exponential with the burst length. This makes interleaving almost mandatory

When there are few errors in the symbol stream, sequential decoding is very fast; it rarely examines more than one branch per n data bits

But when the error rate is high, the decoder can take an impractially long time to finish. Practical decoders use timers; if a timeout occurs, the entire packet is erased

The decoding time of a sequential decoder is therefore a random variable. If the signal is too weak, the average decoding time goes to infinity

Example Performance



Viterbi Decoding

Despite its excellent coding gain, the random speed of sequential decoding is a problem for some real-time applications (e.g., digital voice, synchronous transmission links)

Sequential decoding, as the name implies, is also not well suited to parallel implementation in VLSI

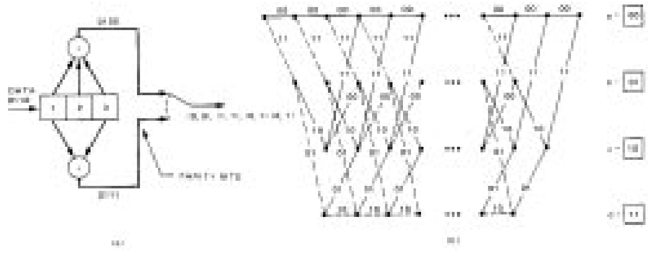
An alternative is ***Maximum-Likelihood Decoding***, also known as ***Viterbi Decoding***

A Viterbi decoder exploits the finite length of the encoder shift register. Data more than K bits old "falls off" the end of the shift register and can no longer influence the encoded symbols

Because of the finite shift register, branches in the code tree eventually remerge, forming a ***trellis*** with a number of states equal to the number of possible shift register states, i.e., 2^{k-1}

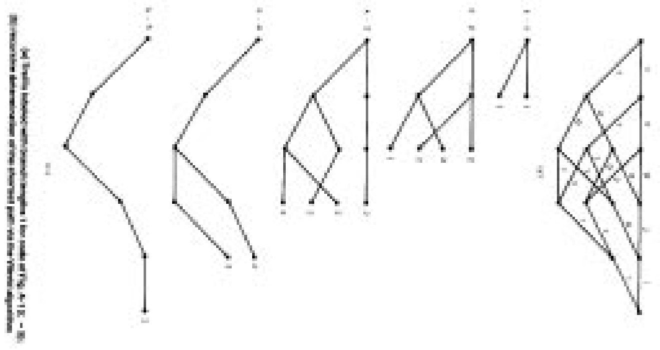
Trellis Diagram

Number of trellis states = 2^{k-1}

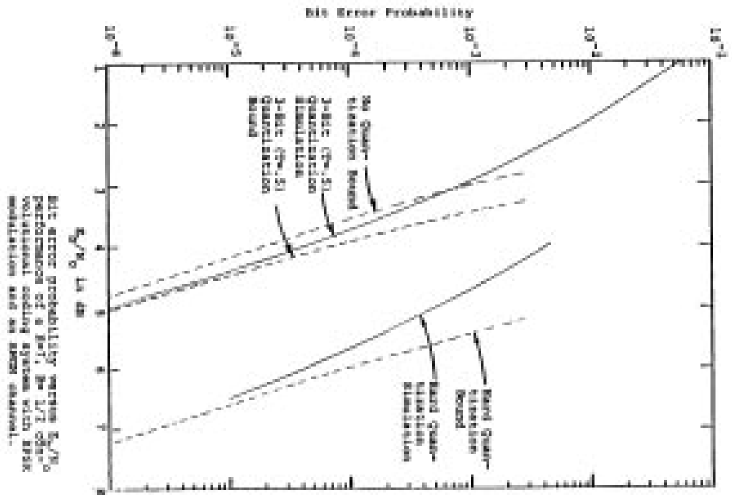


[illegible]

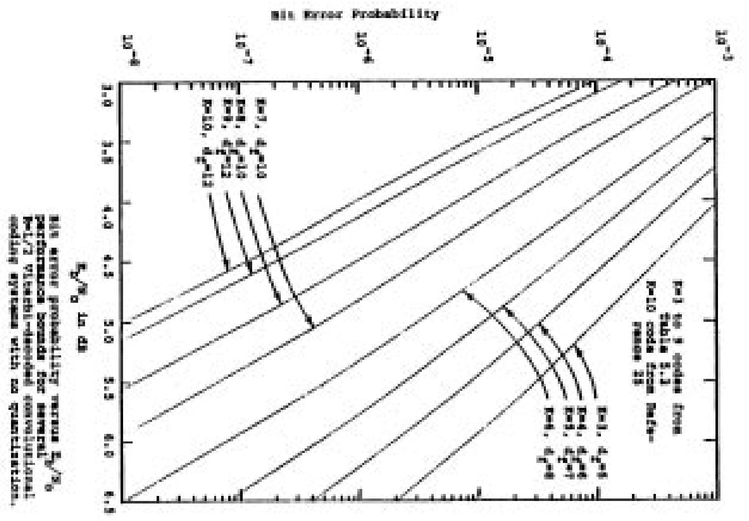
Decoding Example



Viterbi Decoder Performance



Performance vs k



Bit error probability versus E_b/N_0 performance bounds for several R=1/2 Viterbi-decoded convolutional coding systems with no quantization.

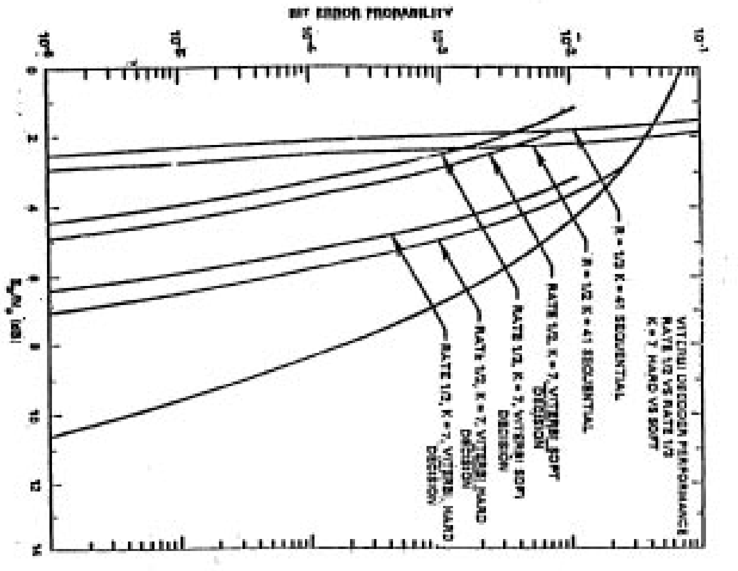
Sequential vs Viterbi

Sequential decoding allows the use of very long constraint length codes. This gives high coding gain with a steep “waterfall” curve. The probability of an uncorrected error is small compared to the probability that the decoder will time out. And the decoding process is relatively fast when the signal is not too noisy

These properties makes sequential decoding well suited for medium speed packet communications, especially when implemented in software

Viterbi decoding operates at a constant rate, regardless of the channel error rate. The algorithm is also ideally suited to parallel implementation in VLSI. This makes it well suited to synchronous, real-time communications at high speeds

Sequential vs Viterbi



Block vs Convolutional

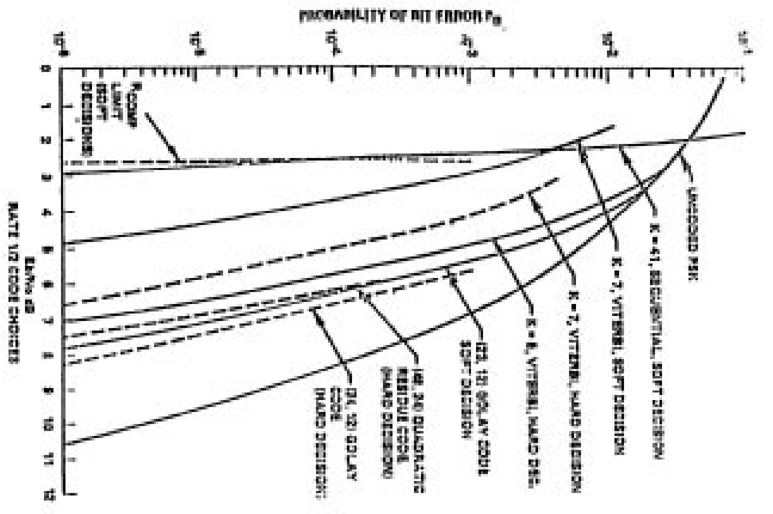
Block

- Practical at very large block sizes for improved performance against burst errors
- Inherently difficult to adapt to soft-decision decoding (Chase method)
- Requires external synchronization

Convolutional

- Outperforms block codes at comparable implementation complexity
- Constraint lengths have practical limits. Interleaving to protect against burst errors
- Easily adapted to soft-decision decoding
- Viterbi decoders can self-synchronize

Short Block vs Convolutional



Concatenated Coding

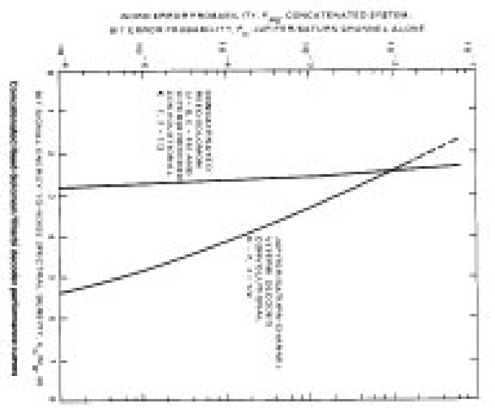
The properties of block and convolutional codes are complementary. This suggests combining the two in order to get the best of both worlds

For example, Viterbi decoders are ideally suited for the power-limited AWGN channel (e.g., a deep-space link) because of their ability to use soft decisions. But the decoder output error rate may not be low enough to satisfy all applications (e.g., compressed images).

Furthermore, when a Viterbi decoder makes a mistake, it generally emits a burst of errors until it recovers. This suggests combining it with an interleaver and a Reed-Solomon decoder good at correcting bursts

These *concatenated codes* have been widespread in deep space communications since Voyager

Voyager Code Performance



Galileo Coding

The failure of the high-gain antenna on Galileo gave coding a chance to “pull out the stops” to make the most of the low gain antenna

The high gain antenna has a r=1/4, K=14 convolutional encoder. A hardware Viterbi decoder was built for this code. The low gain antenna has a standard r=1/2, K=7 convolutional encoder. Both are hardwired to their respective antennas

JPL designed a convolutional coder that executes in software on the onboard computer. When concatenated with the existing K=7 encoder on the low gain antenna, it gives a (different) r=1/4, K=14 convolutional code. This code is Viterbi decoded in software on a Sparc 1000, which is fairly easy at only 10 bps!

Concatenated with the convolutional code is a GF(256) Reed-Solomon block code and interleaver. The RS blocks have varying amounts of parity

Galileo-2

The variable overhead in the RS blocks means that some of the blocks may decode successfully while others do not.

When a RS block decodes successfully, this information is fed back to the Viterbi decoder

The Viterbi decoder uses this information to “pin down” the states it may consider in a second pass over the original received message

The improved Viterbi decoder performance results in better data being fed to the RS decoder, which in turn is able to correct even more erasures, feeding even more information back to the Viterbi decoder

After 4 or 5 iterations, things stop improving. The result is a phenominally low E_b/N_0 requirement of only 0.56 dB!

Coding Over Fading Channels

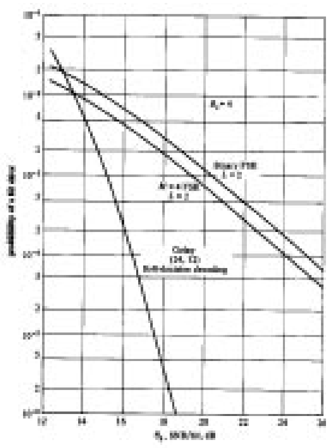
So far the discussion has dealt almost solely with nonfading channels; wideband thermal noise is the only problem

This is true for satellite and deep space communications, where much of the early work in coding was done. But it is not true for many practical terrestrial channels, e.g., mobile radio, HF

Coding is often even more powerful on a fading channel than it is on the AWGN channel. This is especially true for short, deep fading or pulsed interference (e.g., radar). Without coding, the waterfall curves can flatten out an an unacceptably high error rate for any practical E_b/N_0

On HF, coding can be seen as a more efficient form of diversity. Equivalently, simple diversity can be seen as a trivial form of coding (*repetition coding)*

Diversity vs Coding



example of performance obtained with conventional diversity versus coding for $R_c = 4$.

Hybrid ARQ-FEC Protocols

Error control coding is highly effective at optimizing the use of bandwidth and/or power when the channel characteristics are relatively stable and the required error rate is not too low

But computer networking applications have extremely tight error requirements; essentially *no* errors are acceptable. And real channels often vary in quality

This explains the traditional popularity of ARQ (Automatic Request-Repeat) protocols in computer communications

The ideal scheme combines FEC and ARQ to give the advantages of both. Two hybrid ARQ-FEC types exist: Type I and Type II

Type I ARQ-FEC

In a Type I Hybrid ARQ-FEC system, an ARQ protocol is layered on (computer-speak) or concatenated with (code-speak) a FEC scheme such as Viterbi decoded convolutional coding

What errors remain after FEC decoding are detected (e.g., with a CRC) and corrected by ARQ retransmissions. The inner FEC scheme reduces the error rate seen by the ARQ protocol so the latter functions reasonably efficiently, i.e., without too many retransmissions.

The two layers operate more or less independently, though the ARQ layer may be able to ask the FEC layer to change its coding rate to optimize overall performance as the channel changes

The V.32/V.32bis/V.34/V.42bis modem is a very good example of a Type I hybrid. LAPM (in V.42bis) is the ARQ component and the trellis coding in V.32/V.32bis/V.34 is the FEC component.

Type II ARQ-FEC

In a Type II hybrid ARQ-FEC system, the two layers are more closely coupled.

A low rate FEC code that can be *punctured* or *inverted* is chosen. That is, the decoder is capable of decoding a message with a large number of missing symbols

The transmitter first sends only enough of the coded symbols to allow successful decoding on a “clean” channel. If this is successful, an ACK signals the transmitter to proceed to the next data block; the remaining symbols from the acknowledged block are discarded unsent

If the receiver cannot decode the packet, it sets it aside without discarding it. The transmitter then sends additional symbols from the encoded packet. The receiver combines these with the previous transmission and tries again to decode the result

The process repeats until decoding is successful. If all of the coded symbols have been sent and the receiver still can’t decode the packet, the symbols are sent again

Type II - 2

As more of the symbols of a packet are sent, two things happen: the total received E_b/N_0 increases, and the effective code rate decreases, increasing the coding gain.

Eventually the packet can be decoded successfully, assuming that the synchronization mechanism still works

Type II hybrid ARQ-FEC schemes are more complex, so they aren't as popular as Type I schemes. But they are highly efficient and adapt quickly to changing channel conditions, which makes them interesting for radio

Kantronics' G-TOR is a Type II hybrid. The FEC code in G-TOR is the Golay block code

Pactor resembles a Type II hybrid, but the "FEC" scheme (simple repetition of the same uncoded packet) has no coding gain. Pactor relies solely on E_b/N_0 accumulation for its performance

Summary

Error control coding has been around a long time. Most of the important codes and algorithms were discovered over 25-30 years ago. But until relatively recently, strong FEC was used mainly in deep space communications where the benefits outweighed the considerable computational costs

But the computer revolution has changed all this. It is now possible to implement strong FEC at reasonable speeds and at reasonable prices, even in software on general purpose computers

Inexpensive consumer products (CD players, V.32 modems) now incorporate strong FEC that brings these systems remarkably close to the Shannon limit

It's time we brought FEC to ham radio as well!

For Further Reading

Odenwalder, J.P., “Error Control Coding Handbook (Final Report)”, 15 July 1976, Linkabit Corporation, San Diego CA (Excellent practical overview of the field, though slightly dated)

Viterbi, A.J., Omura, J.K., “Principles of Digital Communication and Coding”, McGraw-Hill, 1979. ISBN 0-07-067516-3 (The Viterbi algorithm analyzed by its inventor)

Lindsey, W.C., Simon, M.K., “Telecommunications Systems Engineering”, Prentice-Hall 1973. ISBN 0-13-902429-8. (Good discussion of orthogonal modulation)

Lin, S., Costello, D.J., “Error Control Coding: Fundamentals and Applications”, Prentice-Hall 1983. ISBN 0-13-283796-X. (A very popular course textbook. Good section on ARQ-FEC hybrids)

Proakis, J. “Digital Communications” (2nd Edition), McGraw-Hill 1989. ISBN 0-07-050937-9 (very good all-around reference)

Yuen, J.H., ed., “Deep Space Telecommunications Systems Engineering”, Plenum Press 1983. ISBN 0-306-41489-9. (Good reference for NASA/JPL practices)