

IMPROVING BER

David Spreadbury, Plextek Ltd.
djs@plextek.co.uk

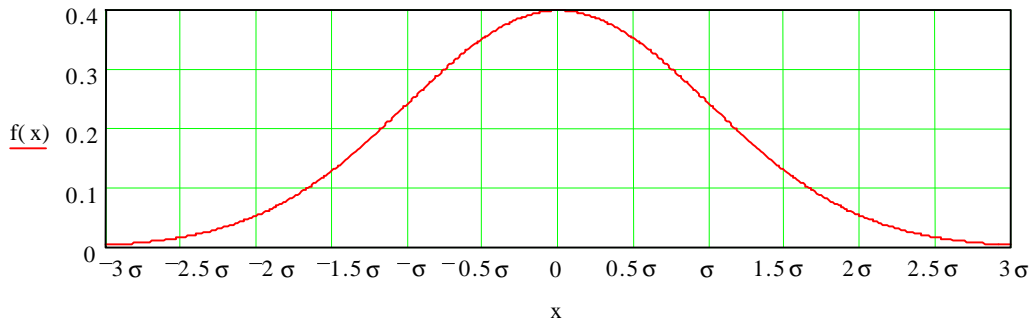
Though it cost all you have, get understanding. Proverbs 4:7

Introduction

The inescapable reality of finite signal and noise powers leads to a finite probability that any bit in any message may be corrupted. Seemingly small bit error probabilities can lead to an unacceptably low message success rate. This presentation looks at ways in which the probability of success of message reception may be improved through intelligent coding.

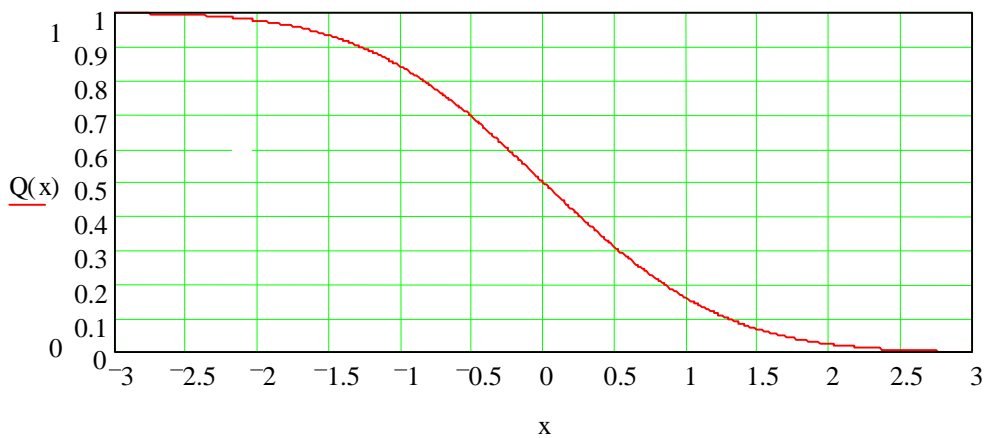
Noise

Noise is considered here to be Gaussian with a probability distribution¹ $f(x) := \frac{1}{\sqrt{2 \cdot \pi} \cdot \sigma} \cdot e^{-\frac{1}{2} \cdot \left(\frac{x}{\sigma}\right)^2}$



The total area under this curve is 1. Integrating this curve from x to +∞ gives the probability that the noise will exceed x:

$$Q(x) := \frac{1}{\sqrt{2 \cdot \pi}} \cdot \int_x^\infty e^{-\frac{1}{2} \cdot v^2} dv$$



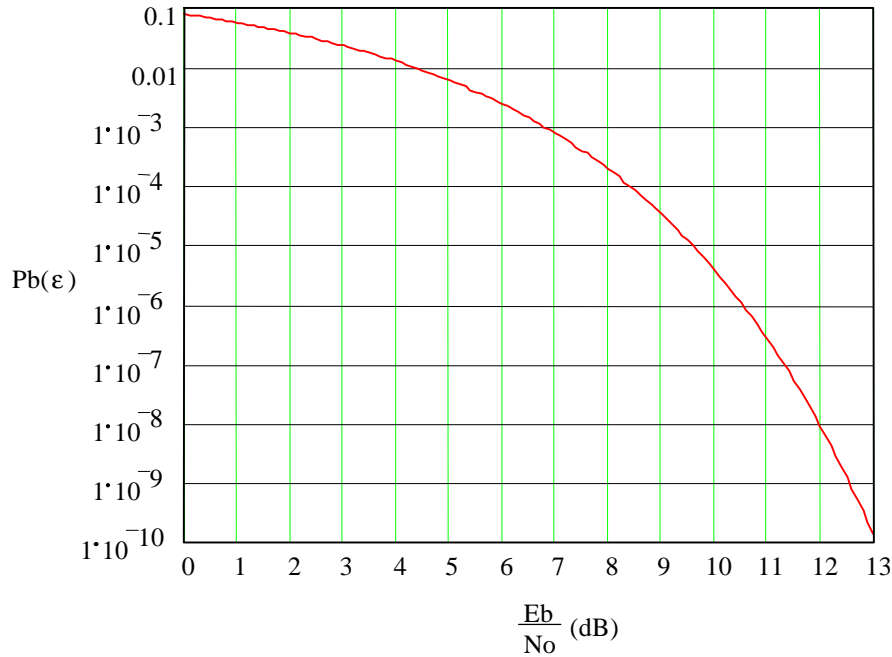
¹ Gaussian noise samples with $\sigma=1$ can be generated from two normally distributed random numbers using $SQR(-2 \cdot LN(RND)) \cdot COS(2 \cdot PI \cdot RND)$ where SQR is square root, LN is natural log and the RND's are two uncorrelated random values in the range 0 to 1. If RND can equal zero, test and replace LN(0) with a const.

BER Calculations

The application of Q(x) can be seen in the bit error probability for, say, BPSK:

$$P_b(\epsilon) := Q\left(\sqrt{2 \cdot \frac{E_b}{N_0}}\right)$$

Which creates the text book curve



If a BPSK signal with $E_b/N_0 = 6\text{dB}$ contains 16 bits, then $P_b(\epsilon) = 2.388 \times 10^{-3}$ and the probability that the complete message is received correctly (i.e. every bit is correct) is

$$(1 - P_b(\epsilon))^{16}$$

which evaluates to 0.962, i.e. there is a 3.8% probability that the message will contain at least 1 bit in error. The more general probability that k out of n bits will be in error is given by

$$P(\epsilon) := \frac{n!}{k!(n-k)!} \cdot P_b(\epsilon)^k \cdot (1 - P_b(\epsilon))^{n-k}$$

Which is the product of

$\frac{n!}{k!(n-k)!}$ the number of ways in which k bits can be selected from n,

$P_b(\epsilon)^k$ the probability that k bits will be in error, and

$(1 - P_b(\epsilon))^{n-k}$ the probability that (n-k) bits will not be in error.

Redundancy

Many signals contain redundancy (e.g. speech), but the only *useful* redundancy as far as BER reduction is concerned is that which is bit-related and can be processed meaningfully by a demodulator. Most efficient communication devices remove any signal redundancy in order to make room for BER-reduction redundancy.

Block Coding

The first technique for improving the probability of message success is to add redundant bits to the message in such a way that the same number of possible messages can be sent (2^{16} here) but they all differ from each other by at least d_{min} bits, where d_{min} is called the *minimum distance* of the code

The number of bits by which any two bit patterns (of the same length) differ from each other is termed their *Hamming distance*.



Consider the two valid codes a and b above, separated by a Hamming distance d_{min} of 7. A little thought will show that if *all* valid messages are at least d_{min} apart, then:

- a) Up to $d_{min}-1$ bits in error cannot change one valid message into another and so $d_{min}-1$ bit errors can be detected
- b) If no more $(d_{min}-1)/2$ bits are in error the nearest valid message will be the correct one, so up to $(d_{min}-1)/2$ bit errors can be corrected.

The above capabilities *are not available simultaneously*, since both require their own Hamming distance margin. For example, if code c above is assumed to be code b with 2 errors and correctable, it cannot be code a with 5 errors and detectable. They must be traded off against each other. For example, if $d_{min}=7$ then the possibilities are:

Maximum correctable errors	0	1	2	3
Maximum detectable errors	6	5	4	3

Each option provides its own probability of *success* (message good or correctable), *failure* (errors detected but uncorrectable) or *error* (errors undetected) [author's own terminology]. These probabilities are calculated by summing the probability of each number of errors occurring into its appropriate category.

Example: There exists a 31 bit BCH (Bose-Chaudhuri-Hocquenghem) code which allows the correction of 3 bit errors and carries 16 user data bits. The first implication of applying this code is that there are now 31 bits in place of the original 16 so E_b/N_0 must go down by 2.87dB to 3.13dB which increases the BER to 0.021. With this revised figure the probability of errors (in the 31 bits) is distributed thus:

Number of bit errors	Probability
0	5.127×10^{-1}
1	3.462×10^{-1}
2	1.132×10^{-1}
3	2.383×10^{-2}
4	3.634×10^{-3}

Number of bit errors	Probability
5	4.275×10^{-4}
6	4.036×10^{-5}
7	3.140×10^{-6}
8	2.052×10^{-7}
9	1.143×10^{-8}

Using these figures, the probabilities in each category sum to

Bits corrected	0	1	2	3
Probability of success	0.513	0.859	0.972	0.996
Probability of failure	0.487	0.141	0.027	0.000
Probability of error	0.000003	0.000004	0.00005	0.004

This shows that the correction capability has given a net increase in message success rate, equivalent in this case to 1.85dB increase in EB/No over the original 16 bit transmission.

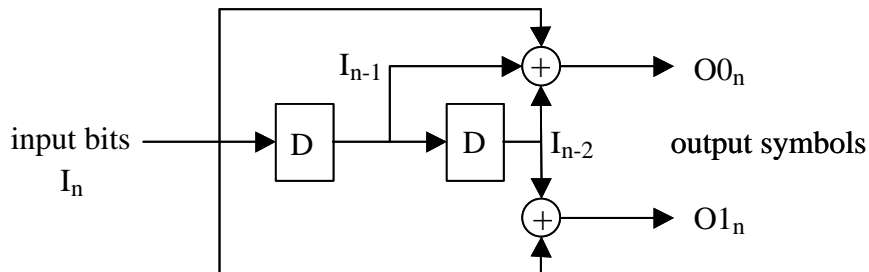
Note also that the probability of error varies considerably, and can be made very small. In some applications acting on the right message is desirable but it is more important not to act on a wrong message. In such cases the probability of error should be minimised, with all but a minute fraction the erroneous messages being detected and discarded.

Convolution Coding

Instead of adding redundancy to blocks of data, it can be more attractive to include it on a continuous basis. Convolution coding works by generating symbols with redundancy from each bit (or group of bits) not in isolation but with a number of bits (or groups of bits) preceding it. This preceding amount, plus 1, is called the *constraint length*, and the ratio of input bits per output symbol/output bits per output symbol is called the *rate*.

The simplest example is the rate = 1/2, constraint length = 7 code:

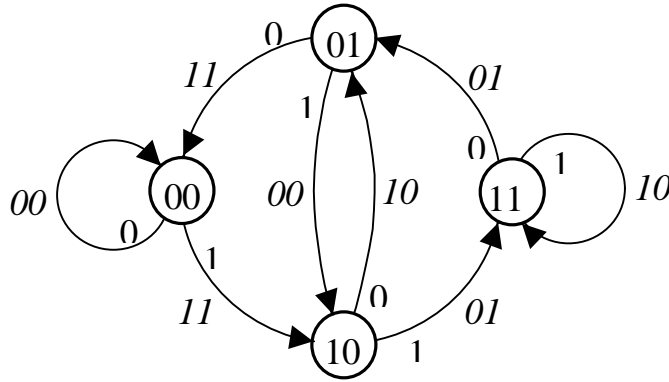
The encoder looks like this, where D is a clock delay (flip-flop) and + is an XOR operation:



The logical response of this circuit is

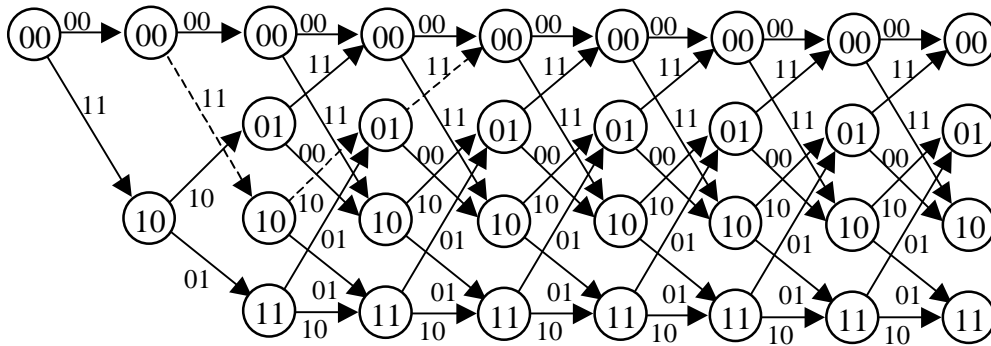
I_n	I_{n-1}	I_{n-2}	O0_n	O1_n
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	1
1	0	1	0	0
1	1	0	0	1
1	1	1	1	0

Which can be drawn more usefully as a *state transition* diagram:



The circled numbers represent the states $I_{n-1}I_{n-2}$, the single digit numbers represent the value of I_n for each state transition, and the italicised double digit numbers represent the output symbols O_0O_1 .

The *trellis* diagram shows possible output symbols and state transitions against time:



The receiver has to select the legitimate path which most closely resembles that received.

Assuming that the transmitter sends continuous 00 symbols (which is legitimate because of the linear nature of the code). Examination of the trellis will reveal a possible erroneous route (dashed) which starts and stops on state 00, is 3 symbols in length, and includes 5 1's, i.e. has a Hamming distance of 5 from the all 0's code. Thus the *free distance* of this rate $\frac{1}{2}$, constraint length 3 code is 5. There are an infinite number of erroneous routes, which may be discovered mathematically rather than by inspection. Those with distances up to 10 are:

Distance	Number of Paths of given Symbol Length											Bit Errors per Path	
	3	4	5	6	7	8	9	10	11	12	13		
5	1												1
6		1	1										2
7			1	2	1								3
8				1	3	3	1						4
9					1	4	6	4	1				5
10						1	5	10	10	5	1		6

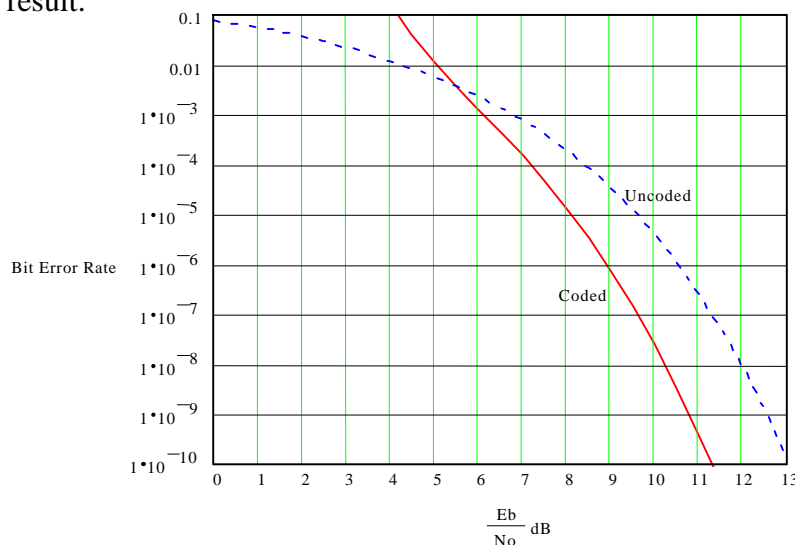
A number of patterns can be seen in this table, and these arise directly out of the encoder structure.

Note that "Distance" means how many bit errors occur in the encoded signal, while "Bit Errors per Path" indicates how many *user* bits will be in error as a result of the incorrect symbol decisions.

The best (maximum likelihood) decoder for convolution coding normally employs the *Viterbi* algorithm, which is an elegant way to make the necessary decisions with the minimum of effort. It is based on the simple observation that of the numerous valid path histories which could be tested at any instant in the decoding process, only *one* terminating on each state (4 here) needs to be remembered (the *survivor*), and that should be the one with the minimum distance from the received signal (known as the path *metric* or cost). When the next symbol is received, for each and every possible current state, the decoder selects that previous legitimate state and its history, which together have the lowest cost, to be the new history for the current state. To its cost is added the cost of the transition to the current state. As this process continues the most probable histories survive, and usually all agree a few symbols back. In practice, therefore, the histories may be truncated to typically a few constraint lengths. The output bit is decoded from the symbol which “drops off” the end of the lowest cost history.

The performance can be estimated in terms of an upper bound on the resulting bit error rate, which is calculated by summing the bit errors associated with each error path weighted by the probability of that error path being chosen. The probability of choosing a path of distance d is the sum of probabilities of enough of the specific d bits being in error to cause this decision (i.e. $(d+1)/2$ to d if d is odd, $d/2$ to d if d is even, halving the $d/2$ result because the decision could go either way).

As before, the bit error rate of the coded signal will be degraded because of the extra bits generated by the coding. The rate of $1/2$ doubles the bit rate and therefore reduces E_b/N_0 3.01dB. Calculating the probability components above for distances up to 30 gives the following result:



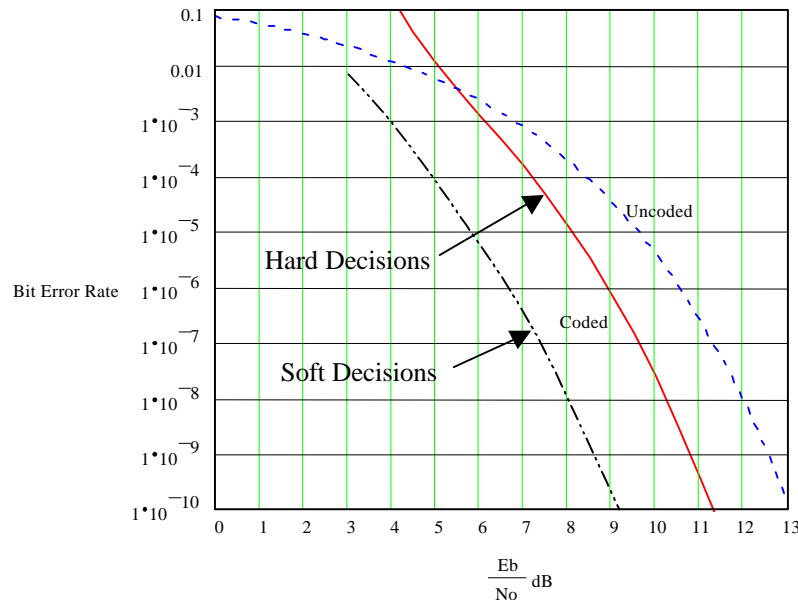
With an original E_b/N_0 of 6dB the raw bit error rate of the coded stream is 2.3×10^{-2} and the resulting user data bit error rate is 1.356×10^{-3} , which is equivalent to a net gain in E_b/N_0 of about 0.53dB. Since the calculations here are for an upper bound (which is why the coded BER rises dramatically for low E_b/N_0) this is a pessimistic figure. The coding gain increases both with increasing constraint length and reducing rate.

Soft Decisions

In the above example, the coding gain is small and would probably not justify the effort. There is, however, a better way of performing the maximum likelihood decoding. The process described above makes a hard binary decision about each incoming bit and then uses only the Hamming distances. This simplifies the hardware, but does not result in optimal performance.

A soft decision decoder looks at the unquantised (or finely quantised) amplitude level, and uses the distance between this and the ideal symbol points to derive a more precise cost function. On a two dimensional constellation, distance² is a convenient measure because it is readily computed using Pythagoras and can meaningfully be summed over a history.

The effect of this approach is that the path errors are compared more faithfully, resulting in more informed decisions. The performance is much more complex to analyse and is best approached through simulation. For the example above, soft decisions would give a further Eb/No gain of about 2.3dB, the general response being:



This discussion has considered the coded symbols as 2 separate bits. In the author's opinion, it is preferable to implement the coding by increasing the complexity of the original symbols rather than adding extra symbols as this removes any requirement or dependency on additional synchronisation.

Example: It is interesting to note the similarity between the state transition diagram above and that resulting from the MSK modulation which uses 1 and 1½ cycle symbols given as an example in the previous paper on Clock Extraction. The received constellation can be thought of as 4 states, i.e. two symbols each of dual polarity, which have the same set of legal transitions. The author did in fact use a simple soft decision Viterbi decoder with distance² costs to demodulate these signals in the equipment mentioned.

Concatenated Coding

For those applications which have the requirement and resource for the ultimate in error correction, (e.g. deep space communications) the two types of coding discussed (block and convolution) can be very effectively combined. The coding process starts with block coding (typically *Reed-Solomon*), after which the bits are interleaved so as to distribute the contents of each block over several blocks worth of time. This revised bit sequence is convolution coded and transmitted. In the decoder, the signal is soft decision Viterbi decoded. It is a characteristic of convolution coding that output errors tend to come in multiples rather than singly. The subsequent de-interleaving stage reconstructs the blocks, and in doing so spreads any multiple errors over a number of blocks. Each block is then decoded individually and corrected in the normal way.

Summary

The principles of block and convolution coding to detect and correct errors have been introduced, albeit in simple form. Far from being mysterious, these techniques are now in common use in mobile phones and CDs and will increasingly appear in RF power budget calculations. They are based on mathematical principals and can be readily modelled on a PC and implemented in hardware or software.

References

Viterbi, A.J.; "Convolution Codes and Their Performance in Communication Systems", IEEE Trans. COM-19, No. 5, October 1971, pp 751-772.

Forney, G.D.Jr.; "The Viterbi Algorithm", Proc. IEEE Vol. 61, No. 3, March 1973, pp 268-278.

Bibliography

Ziemer, R.E. and Peterson, R.L.; "Introduction to Digital Communication", Maxwell 1992, ISBN 0-02-946431-5.

Proakis, G.P.; "Digital Communications, Third Edition", McGraw-Hill 1995, ISBN 0-07-113814-5.

The following are useful but far from easy:

Houghton, A.D.; "The Engineer's Error Coding Handbook", Chapman & Hall 1997, ISBN 0 412 79070 X.

Hill, Raymond.; "A First Course in Coding Theory", Oxford University Press 1996, ISBN 0 19 853803 0.

MacWilliams, F.J. and Stone, N.J.A.; "The Theory of Error-Correcting Codes", North-Holland 1992, ISBN 0 444 85193 3.

David Spreadbury graduated from The City University in 1971 with a first class honours degree in Electrical and Electronic Engineering. An experienced hardware designer, low level programmer, and systems engineer, he has spent twenty five years "doing DSP", in both military and commercial environments. Most of his projects employ novel algorithms or architectures, some in ASIC, to achieve speed, size, performance or cost targets. David is married, with two teenage children. He is actively involved in his local church and enjoys fixing things, reading, maths, origami, and a strictly limited amount of sport.

Plextek Ltd. is an independent communications technology consultancy, located in the village of Great Chesterford near Cambridge, which specialises in digital telecommunications and mobile radio communications. The principal services offered are product and system development, with hardware, software and ASIC design capabilities, and radio system planning and survey. Plextek also offer industry, product and technology studies and undertake standards and regulatory assignments.