Intuitive Guide to Principles of Communications
www.complextoreal.com

Trellis Coded Modulation (TCM)

Trellis Coded Modulation Tutorial
Before reading this tutorial, make sure you have read the tutorial on Convolutional coding and Modulation. This is essential.

What does a rate 1/2 convolutional code do? It takes one bit, codes it, puts out 2 bits. This is something that is done at the digital level. These 2 bits are then modulated (i.e. converted to analog form via a sinusoidal carrier) and transmitted. The coding is a digital function and modulation which is an analog function, are done separately and independently in most modulations. In Trellis Coded Modulation (TCM), however the two are combined in one function.

TCM concepts

Euclidean Distance

A straight line distance between any two points is called the **Euclidean distance**. For a point p_1 at (x_1, y_1) and another point p_2 at (x_2, y_2), the Euclidean distance is given by the familiar formula

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

It is implicit that this distance is the shortest distance between these points. The Euclidean distance is an analog concept, the very concept of *distance* that we normally use day-to-day in the world of real numbers. For signals, we define this distance in the I-Q plane. In Figure 1 we have a 8PSK signal constellation. The radius is equal to 1 and represents the maximum amplitude. Each point of the constellation is a certain combination of a particular amplitude and phase. The distance between these points is can be measured in the manner described above and these are given in the Figure below. The distances given in Figure 1 are squared and are called **Squared Euclidean Distance (SED).** The smallest of these distances is called the **Minimum Squared Euclidean Distance (MSED)**, designated as $d^2_{\min}$ for a particular constellation.

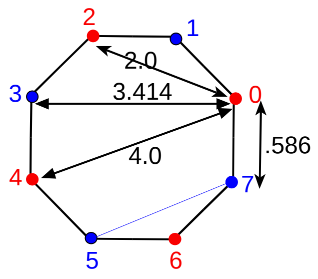


Figure 1 – 8 PSK constellation and squared Euclidean distances between symbols

Hamming distance

Just as real numbers have a concept of distance, so do the binary numbers. Take two binary numbers, *011011* and *101101*. The distance between these is the number of places these two numbers differ. And that number is 4. This distance is called the **Hamming distance** between these numbers. The distance would be zero, if these two numbers were the same. A zero distance means the numbers are the same, same interpretation as in Euclidean concept of distance.

We distinguish these two types of distances by recognizing that one belongs to the analog world of real numbers and the other to the binary world. Both concepts are useful in signal processing. In coding Hamming distance is most often used as a performance metric whereas it is Euclidean distance in the analog world.

Distance between sequences

We can also talk about Euclidean distances between *sequences* by comparing distances between corresponding points of the sequences. Let's take for example an 8PSK signal that consists of a sequence of these symbols.

S_0 S_3 S_2 S_1 S_0

In bits, we can map these as: 000 011 010 101 100 000

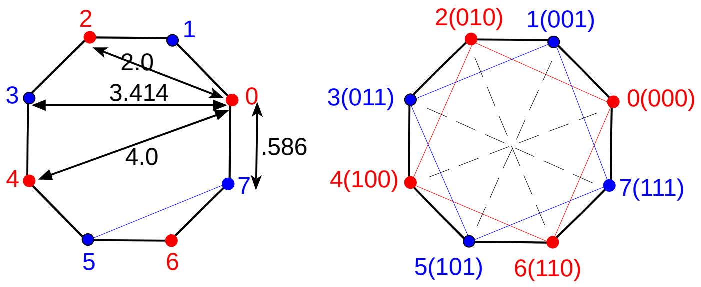


Figure 2 – Euclidean and sequence Hamming distance

The Euclidean distance for this sequence is the distance between each symbol in this sequence and a reference sequence. If we designate the **all-zero-symbols as the reference sequence**, then the squared Euclidean distance (SED) is the distance between each one of these symbols and the symbol S_0.

s_0 to s_0 = 0.0, s_0 to s_1 = .586, s_0 to s_2 = 2.0, s_0 to s_3 = 3.414

The **Sum of the Squared Euclidean Distances (SSED)**, also called d^2_{free} of this sequence, from the all zero sequence is 3.414 + 2.0 + 0.586 = 6.0

This cumulative distance gives a feeling of how easy or difficult it would be to mistake one sequence for another. For the reference sequence we could have used any other sequence than the all zero, and the results would be the same. However using an all-zero sequence is convenient and conventional.

Trellis Coded Modulation (TCM)

TCM uses many diverse concepts from signal processing. In simplest terms it is a combination of coding and modulation, hence it name **Trellis Coded Modulation**, where the word trellis stands for the use of trellis (also called convolutional) codes. Whereas we normally talk about coding and modulation as two independent aspects of the communications link, in TCM they are combined.

TCM is a complex concept to understand particulalry due to the non-linear nature of the performance. It uses ideas from modulation and coding as well as dynamic programming, lattice structures and matrix math. A convolutional code that has optimum performance when used independently may not be optimum in TCM. Gray coding is helpful in uncoded signalling and constellation mapping, but not always so in TCM. So it is a not an easy topic and my hats off (if I wore one) to Mr. Ungerboeck and others who came up with it.

Fortunately, there is not a lot of math to deal with here. But you will need to know concepts of convolutional codes, trellis, lattice, cosets, and coset generators.

Communications theory says that it is best to design codes in long sequences of messages. The **allowed sequences** should be very different from each other. The receiver can then make a decision between sequences using their statistics rather than on symbol-by-symbol basis. When decoding this way, the probability of error is an inverse function of the sequence length. In general form the probability of error between sequences is given by the expression

$$p_e \sim e^{-d^2_{\min}/2\sigma^2}$$

where d_{min} is the sequence Euclidean distance between sequences and is σ^2 the noise power. We measure the performance of TCM (and many other schemes) by **Asymptotic Coding Gain (ACG)**. This is the gain obtained over some baseline performance at high SNR in a Gaussian environment. AGC is not achievable in practice because we do not transmit signals at high SNRs, have hardware and channel imperfections that depart from **Additive White Gaussian Noise (AWGN)** assumptions. So recognize that all gains quoted herein are maximum possible only in theory. Actual numbers are determined by test and simulation in the given environment.

The functions of a TCM consist of a **Trellis code** and a **constellation mapper** as shown in Figure 3. TCM combines the functions of a convolutional coder of rate $R = k/k+1$ and a M-ary signal mapper that maps $M = 2^k$ input points into a larger constellation of $M = 2^{k+1}$ constellation points.

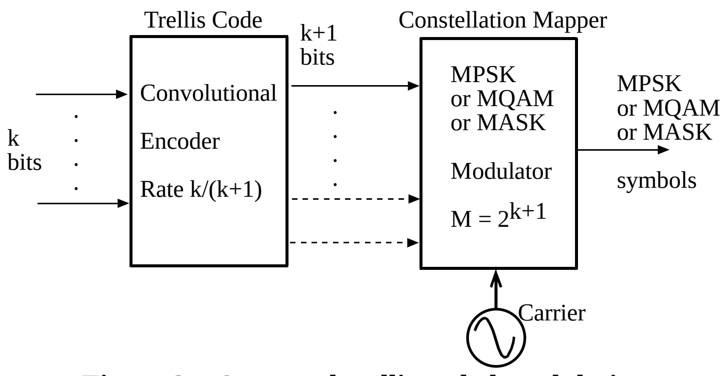


Figure 3 – A general trellis coded modulation

For k = 2, we have a code of rate 2/3 that takes a QPSK signal (M = 4) and putsout a 8-PSK signal (M = 8). So instead of expanding the bandwidth as the signal goes from QPSK to 8PSK, it instead doubles the constellation points. It is kind of an upgrading system, where you take a chosen signal and upgrade it to another with larger number of constellation points as shown in Figure 4 below.

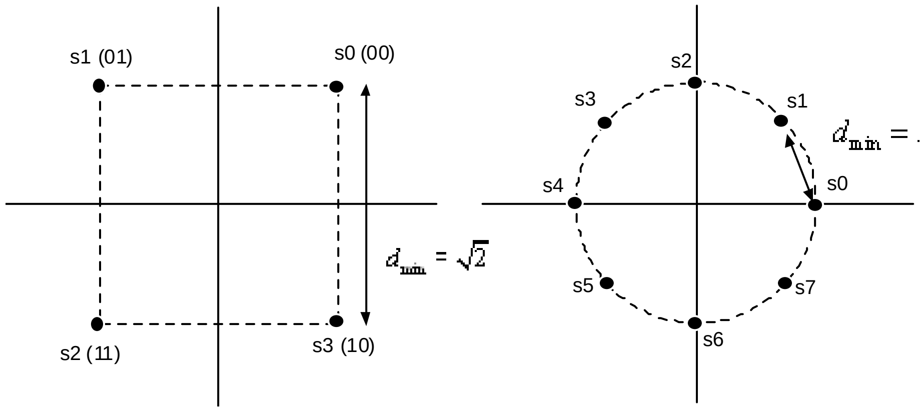


Figure 4 – Constellation doubling in TCM. A QPSK signal transmitted using a 8PSK constellation.

Main points of a TCM are

1. TCM is **bandwidth efficient modulation** which accomplishes this by the use of convolutional coding.
2. It conserves bandwidth by doubling the number of constellation points of the signal. This way the bit rate increases but the symbol rate stays the same.
3. Convolutional coding constrains allowed symbol transitions, creating **sequence coding**.
4. Unlike a true Convolutional coding, not all incoming bits are coded.
5. Increasing the constellation size reduces Euclidean distances between the constellation points but sequence coding offers a coding gain that overcomes the power disadvantage of going to the higher constellation.
6. Performance is measured by coding gain over an uncoded signal.
7. The decoding metric is the Euclidean distance and not Hamming distance.
8. Ungerboeck originally proposed TCM which used **set-partitioning** and small number of states with code rates that varied with the input signal type.
9. Pragmatic TCM uses a less than perfect rate ½ convolutional code with constraint length equal to 7 or 9. This is a widely available code and its use makes TCM less expensive to implement.
10. The **constellation mapping** in set partitioning is based on natural numbering where as gray coding is preferred in pragmatic TCM.

TCM is a general concept and by varying k, we can create a QPSK, 8PSK or higher level signals as shown in Figure 5. All of these are types of TCM.

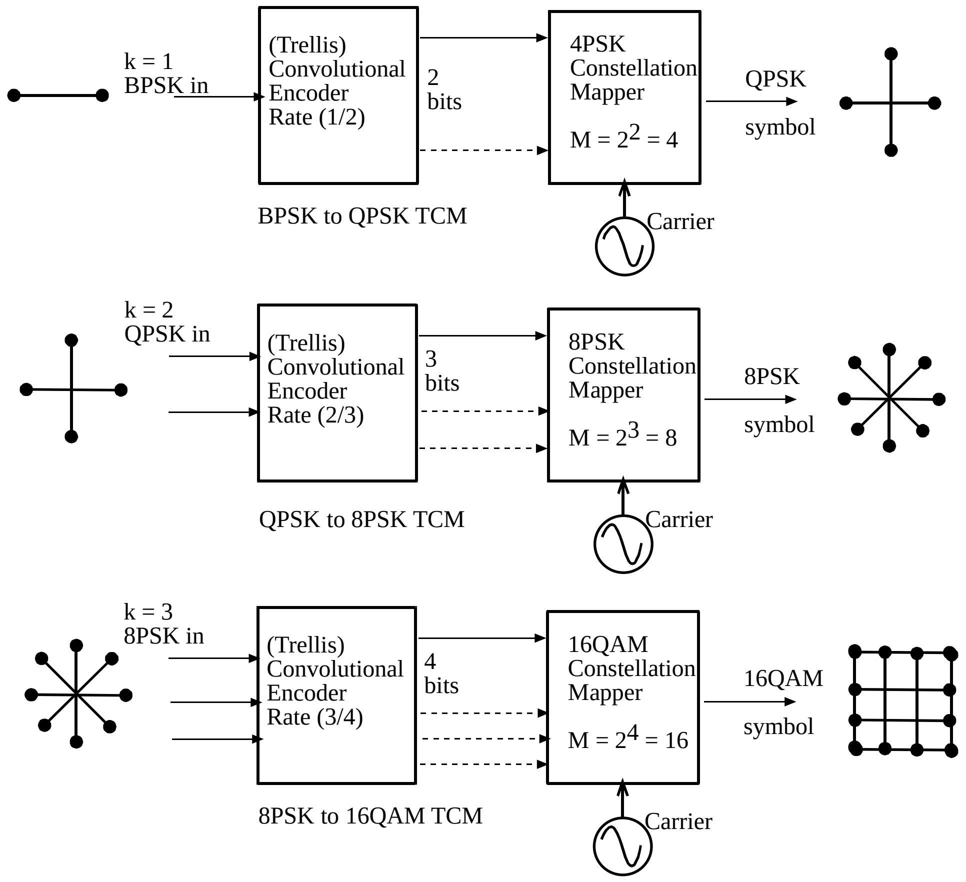

k = 1
BPSK in
(Trellis) Convolutional Encoder Rate (1/2)
2 bits
4PSK Constellation Mapper
M = 2^2 = 4
QPSK symbol
Carrier
BPSK to QPSK TCM
k = 2
QPSK in
(Trellis) Convolutional Encoder Rate (2/3)
3 bits
8PSK Constellation Mapper
M = 2^3 = 8
8PSK symbol
Carrier
QPSK to 8PSK TCM
k = 3
8PSK in
(Trellis) Convolutional Encoder Rate (3/4)
4 bits
16QAM Constellation Mapper
M = 2^4 = 16
16QAM symbol
Carrier
8PSK to 16QAM TCM

Figure 5 – General trellis coded modulation
(a) BPSK, code rate 1/2, output QPSK (b) QPSK, code rate = 2/3, output 8PSK
(c) 8PSK, code rate = 3/4, output 16QAM

Notice that in each case, the code rate is different. But they are all called TCM. The coding adds just one extra bit to the symbol bit size. The symbol size increases from k bits to k + 1 bits. If coding increases the bit rate by 1 extra bit, then we need to double the constellation size to accommodate this bit as we see below, where L is original number of bits per symbol.

$$2^{L+1} = 2 \times 2^{L}$$

So if original signal is BPSK, then a TCM encoder will put out a QPSK, a QPSK will become 8PSK and if 8PSK was chosen, it becomes a 16QAM signal. As constellation expands but the signal energy is kept the same, the distance between the symbols decreases. That implies a worsening of a performance, not improving. So where is the improvement coming from?

We start with a given bandwidth B, because in real life this is a big constraint. From this bandwidth, we determine the maximum possible symbol rate, which is never more than 2B but usually less. Now determine the size of the alphabet that can deliver the needed signal BER at the given available power.

The uncoded QPSK signal (k = 2) is TCM'd

Let’s say that we intend to transmit a QPSK signal. Figure 6 shows the constellation of a QPSK signal of amplitude = 1. The minimum Euclidean distance between the four constellation points is $d_{\min} = \sqrt{2}$. Now we draw the trellis of this QPSK signal which is a trivial case as in Figure 6 (b).

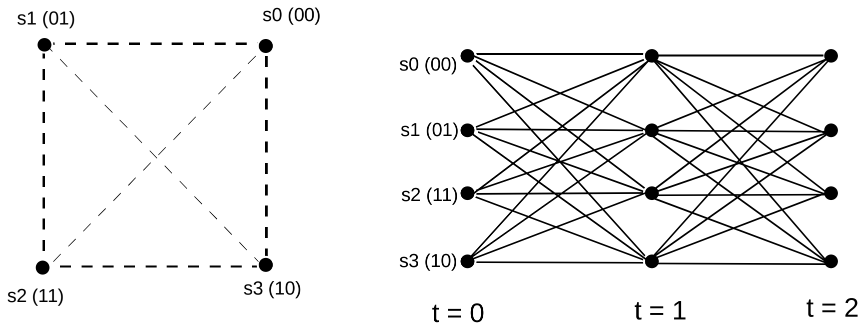


Figure 6 – QPSK signal constellation and its trellis

(a) Four constellation points of QPSK, (b) the signal trellis of a QPSK signal which allows all transitions at each time period.

This trellis shows how symbols are transmitted in an uncoded QPSK signal. Sixteen paths are shown in two time periods. Any of these 16 paths is possible for an uncoded signal. At time t = 0, we can pick any of the four possible symbols, s_0 to s_3, and then same at each subsequent time tick, again we can pick any of the four possible symbols. It is trivial but I want to get across the point that there are no transition restrictions in an uncoded signal. All sequences are possible ergo, this signal has no **sequence coding**.

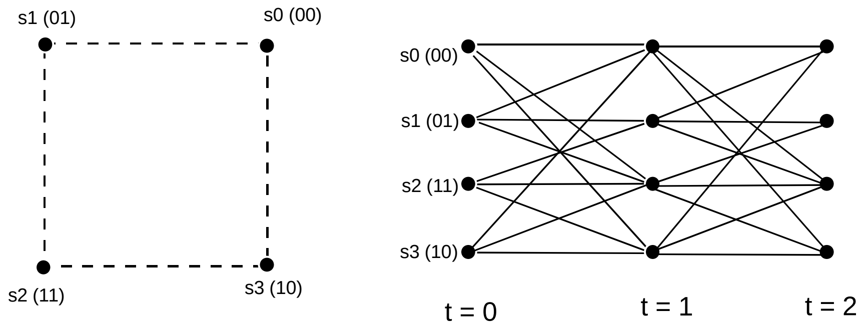


Figure 7 – SQPSK signal constellation and its trellis

(b) Four constellation points of S-QPSK, (b) the signal trellis of a S-QPSK signal which allows only three transitions from each point at each time period.

In Figure 7, we show the trellis for a staggered QPSK (also called offset QPSK) and we see that here we do have restrictions on what transitions are allowed. From each state, we can only go to the two adjoining symbols.

The **Minimum Squared Euclidean Distance (MSED)**, is the minimum squared distance between all points of a constellation and is usually the distance between any two adjacent points. The decoder on the receive side makes the decision about which symbol was sent based on which decision region the signal falls in sort like a dartboard. The error performance is a function of the

MSED, d^2_{min} which for signals of Figure 6 and 7, is 1.414.

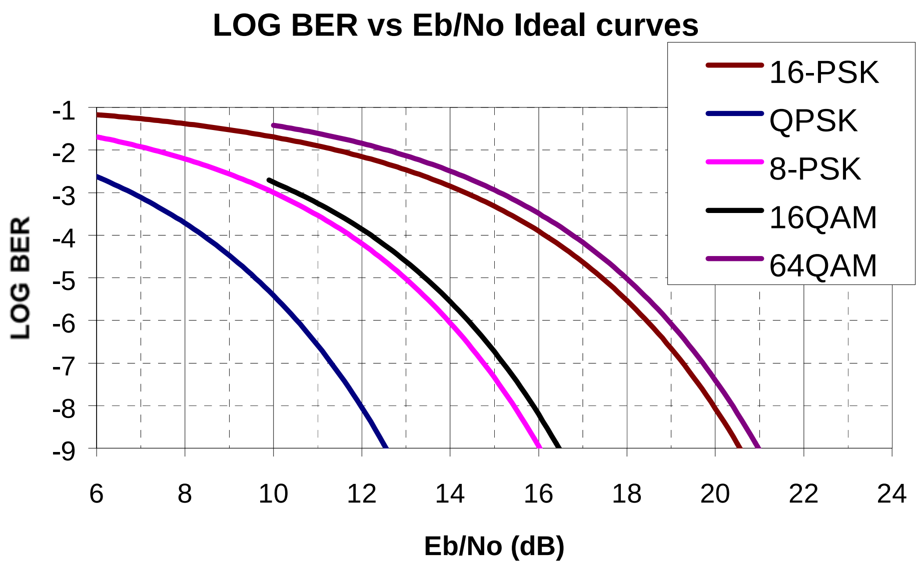


Figure 8 - Ideal BER vs. Eb/N0 in AWGN environment

Let’s say we want to transmit a QPSK signal, uncoded with a BER of 10^{-5}. This is going to require 9.6 dB of energy per the ideal Eb/No vs. BER relationship. If that much power is not available because the transmitter is small, then an options is to add a code of rate 2/3 to reduce the BER which will give this BER at a smaller Eb/N0. But then we have a another problem. If we keep the same bit rate for the information bits and let the coded bit rate increase to accommodate the overhead bits, then bandwidth requirements increase by the 1/R. So addition of coding increases the bandwith by 3/2. If we cannot allow the bandwidth to change, then information rate will have to decrease by the same proportion.

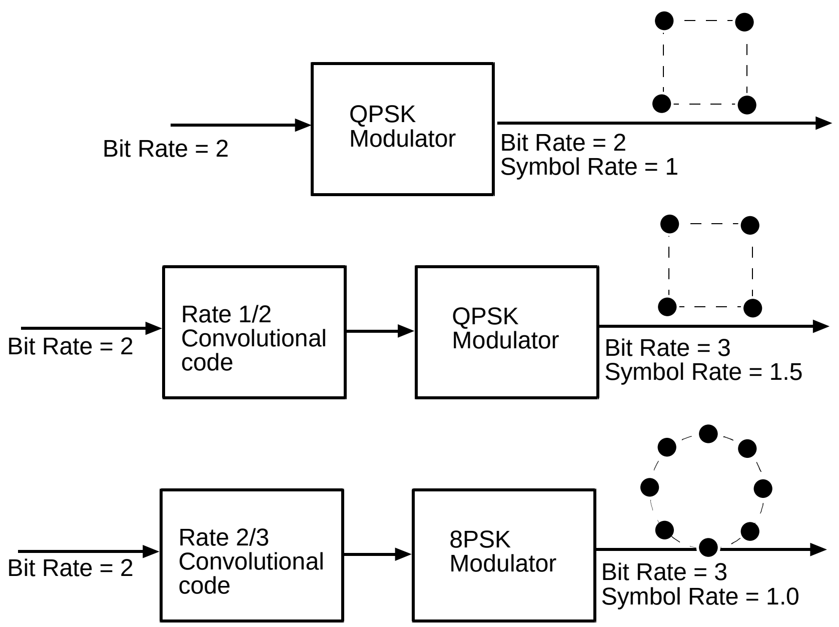


Figure 9 – Adding independent convolutional coding to a QPSK signal increases its bit rate and required bandwidth by the code rate (a) uncoded signal (b) Add a rate ½ code infront of the QPSK modulator but this increases the symbol rate. (c) add rate 2/3 code and transmit 8PSK signal.

For the QPSK example, before coding, the information bit rate was 2 bps, and after adding one bit for coding, the bit rate becomes 3 bps. The symbol rate which was 1 sps before coding is 1.5 sps afterwards. What happens to bandwidth when you increase the symbol rate from 1 to 1.5? As symbol rate is increased, the bandwith (as measured by the frequency of the first null) also increases. A signal with symbol rate of 1 sps has a lowpass bandwidth of 1 Hz, a signal of 2 sps has a bandwidth of 2 Hz and so on. The bandwidth of the signal increases from 1 Hz for the uncoded signal to 1.5 Hz for the coded signal.

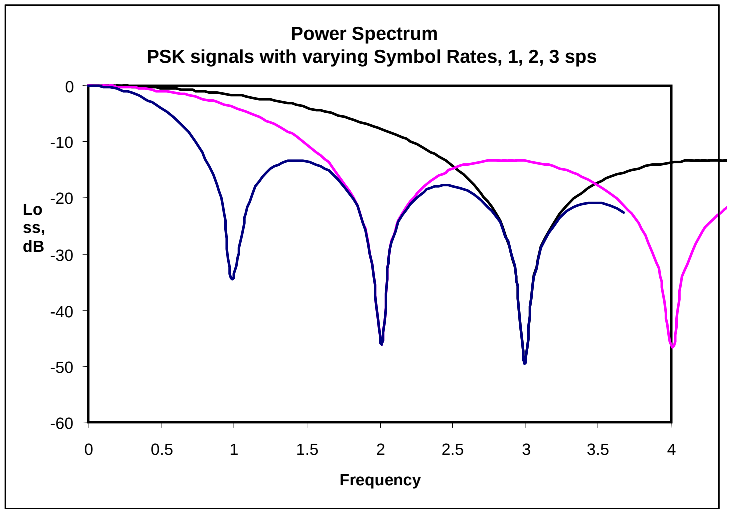


Figure 10 – Symbol rate vs. required bandwidth of a PSK signal.

For PSK as long as the symbol rate is the same, all M-PSK modulations have the same bandwidth, whatever the M. Now look at Figure 10 where required bandwidth is a function of the bit rate instead of symbol rate. All three of these signals carry the same bit rate but 8PSK requires the least bandwidth. The bandwidth requirements for a given bit rate decrease as M is increased in a M-PSK signal which is what is meant when these are called bit-efficient modulations.

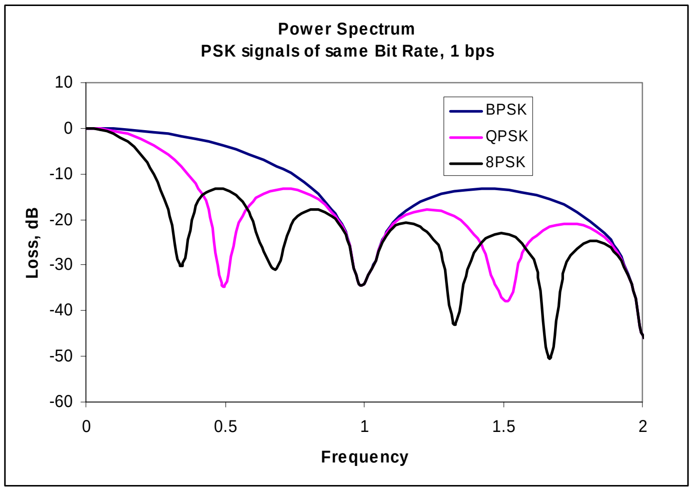


Figure 11 – For same bit rate, the required bandwidth decreases as M increases in a M-PSK signal.

In third option in Figure 9, we move up in modulation order and in this way can transmit a signal which has coding but without changing the bandwidth or the spectrum. Now the problem is that 8PSK symbols are closer together. Looking at the curve from Figure 8 we see that it needs app. 3.5 dB higher power to deliver the same BER. Can a code of rate 2/3 give us a coding gain of at least 3.5 dB? Because that is the only way this idea will be work. Otherwise we have not gained anything over the uncoded QPSK option.

The coding gain of coded signal over an uncoded signal

The coding gain of a coded signal is given by

$$\gamma = \frac{d^2_{free/coded}}{d^2_{\min/uncoded}} \frac{E_{s/coded}}{E_{s/uncoded}} \qquad (1)$$

where $d_{free/coded}$ is equal to the coded sequence Euclidean distance and $d_{min/uncoded}$ is minimum distance between the signals in the uncoded constellation as previously defined. This coding gain is referenced to the baseline signal. We are starting with QPSK and want a BER of 10^{-5}. We said that this requires a Eb/N0 of 9.6 dB. The number we come up for coding gain from Eq. 1 is counted as reduction from the baseline Eb/N0. Fact that 8PSK actually requires a higher Eb/N0 than the QPSK baseline is accounted for in Eq. 1.

So even though $d_{\min/coded} < d_{\min/uncoded}$, if we can increase $d_{free/coded} > d_{\min/uncoded}$ we will get a positive gain. We know what d_{min} is, but what is d_{free}?

How do you measure the Euclidean distance of a coded signal when in fact, its constellation Euclidean distance is smaller than d_{min}. Now we need to talk about distances between sequences rather than distances between signals.

d_{free} is defined as the Euclidean distance of a coded signal in terms of the smallest possible distance between all allowed sequences. Instead of checking all possible combinations, this distance is measured from the all-zero sequence, same as the measurement of Hamming distance and Hamming weight, which are both referenced to an all-zero code word.

Let say that we transmit an all-zero message. During decoding, errors occur and wrong trellis path is followed by the decoder. The only way, the decoder can get depart and then back to the correct path, which is the all-zero path, is by diverging and then remerging as shown below.

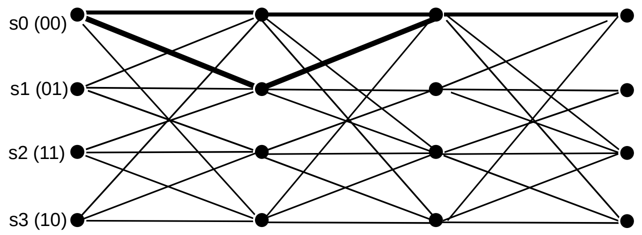


Figure 12 – diverging and remerging of alternate sequences. The correctly decoded sequences is the upper straight line. The shortest incorrectly decoded sequences is the darker 2 segment path.

The Euclidean distance of a coded signal is defined as the smallest distance between an all-zero sequence and one that diverges from it and then remerges. What exactly does that mean?

Remember that each sequence is a set of demodulated symbols. When two paths diverge, it means, there was an error and the decoder made a wrong decision. Assuming that the probability of such error is small, at subsequent junctions, the further decisions should take the path back to the correct one. A small allowed path which incorrect is more likely than a long one, so the distance of this small path is a measure of the error correcting capability of the code. This is the same concept as when a receiver is most likely to pick neighboring signal points because they are most like the correct signal, rather than picking one that is quite different, as such the fundamental idea behind **Maximum Likelihood Decoding (MLD).**

In the example of the simple trellis above we see that if an error is made at the time t = 0 and the path diverges, then it can get back to the correct sequence in only two segments as shown. The sum of the squared Euclidean distanced for this path is called the free distance, d_{free} of the code.

The Euclidean distance here is determined by looking at the diverging path followed and its minimum distance from the all-zero path, symbol by symbol. For the example above, this is sum of SED of each of the two segments.

$$\begin{aligned} d^2{}_{free/coded} &= d^2_{\min} + d^2_{\min} \\ &= 2+2 \\ &= 4 \end{aligned}$$

Let's compute d_{free} and the coding gain of a rate 2/3 convolutional code to see what it can do.

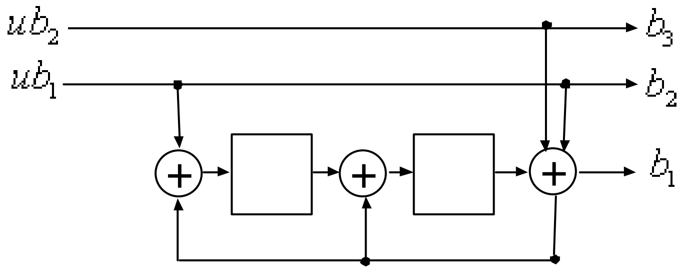


Figure 13 – a Rate 2/3 convolutional encoder

This is a four state convolutional code. Looking at the above diagram you can write down its transfer function. It is a systematic encoder since two uncoded bits go through and are appended by one parity bit.

We are going to look at the performance of this code based on two different bits to symbol mappings, in (b) the symbols are mapped in the natural binary order and in (c) they are mapped with Gray coding so that adjacent bits differ by one bit only.

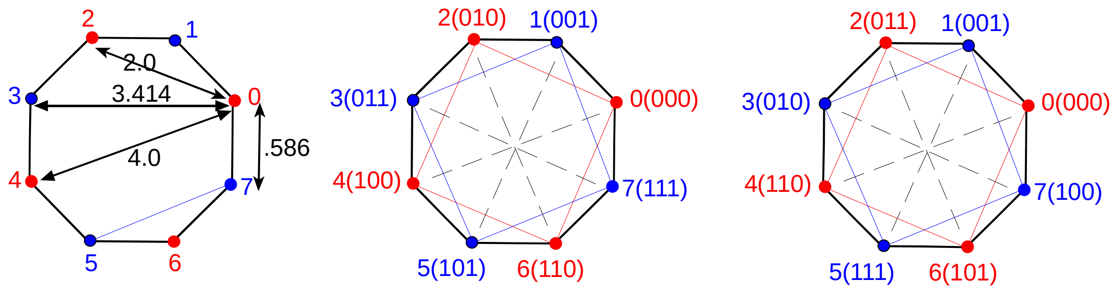


Figure 14 – (a) Mapping the 8PSK symbols (b) Natural mapping, (c) Gray coding

Note that although d^2_{min} for a QPSK signal was 2.0, the same number is 0.586, nearly 75% smaller for 8PSK.

We write down the trellis of the 2/3 convolutional code by stating with 0's in the memory registers and then putting in all three bit combinations through the registers. (See Convolutional Coding tutorial on how to do this.)

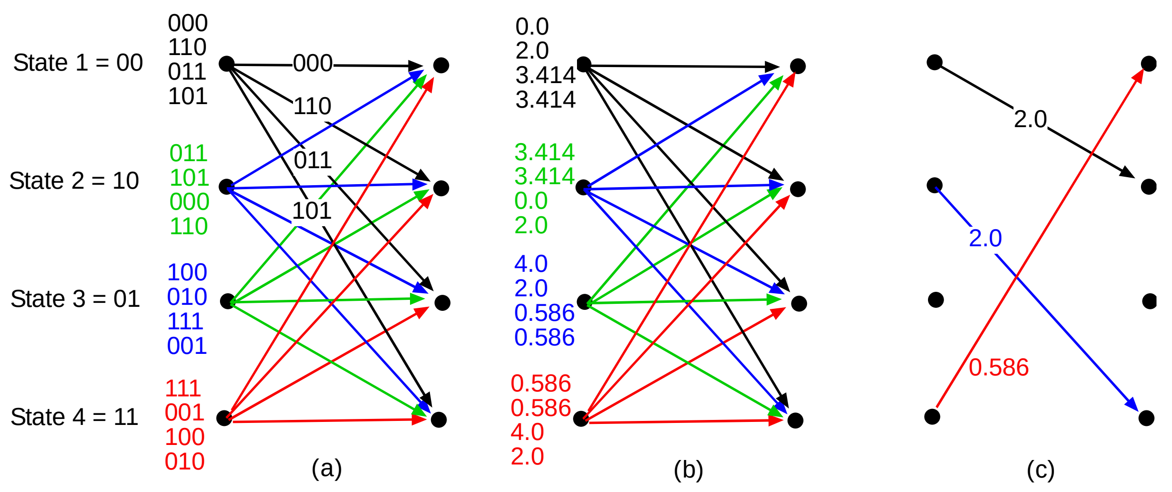


Figure 15 - Trellis of a rate 2/3 convolutional code, (a) Trellis with bit assignments (b) each path replaced with its SED (c) Three segments that diverge and then merge back to state 00.

In (a) the trellis shows the four possible states of the registers with each line diverging in a bit combination in order from top to bottom. In (b) we replace the symbols by their squared Euclidean distance from the zero symbol reference. (These distances are written on the side rather on the lines to keep the diagram easy to decipher.) At each state, there are four different diverging paths.

In order to determine the coding gain, we need to know the d_{free} of this code. To determine the minimum distance path, we follow from each state the path with the smallest squared distance (but not 0). This is 2.0 for the path starting at state 00. This takes us to state 2. From here we follow again the minimum distance path, which is 2.0. This takes us to state 4 and from here we return to state 0 via a path that has a squared distance of 0 .586. There is no other path that can take us back to 00 state and has a smaller total distance.

The total minimum squared Euclidean distance (MSED) for this sequence is the sum all three of these squared distances.

2 + 2 + .586 = 4.586.

To determine the coding gain, we divide this distance by the $d^2_{\min}$, the minimum distance of the uncoded QPSK constellation which 2.0 The coding gain from Eq. 1 (assuming that both coded and uncoded signals have same energies, is

$$10\log\left(\frac{4.56}{2}\right) = 3.6 \text{ dB}$$

This is great but maybe we can do better. Let's try an alternate mapping of the symbols (Gray coding) as shown in (c). This gives the trellis of Figure 15. As before, we convert each symbol to its distance from the 0 symbol and then we follow the minimum path. This time to our chagrin, we only get a total MSED of

.586 + .586 + .586 = 1.758, or actually a loss of approximately 0.5 dB.

So we see that mapping is an important consideration. Gray coding was not helpful here. Adding any code of rate 2/3 in front of a 8PSK modulator is not beneficial.

Now see how TCM can help improve on this.

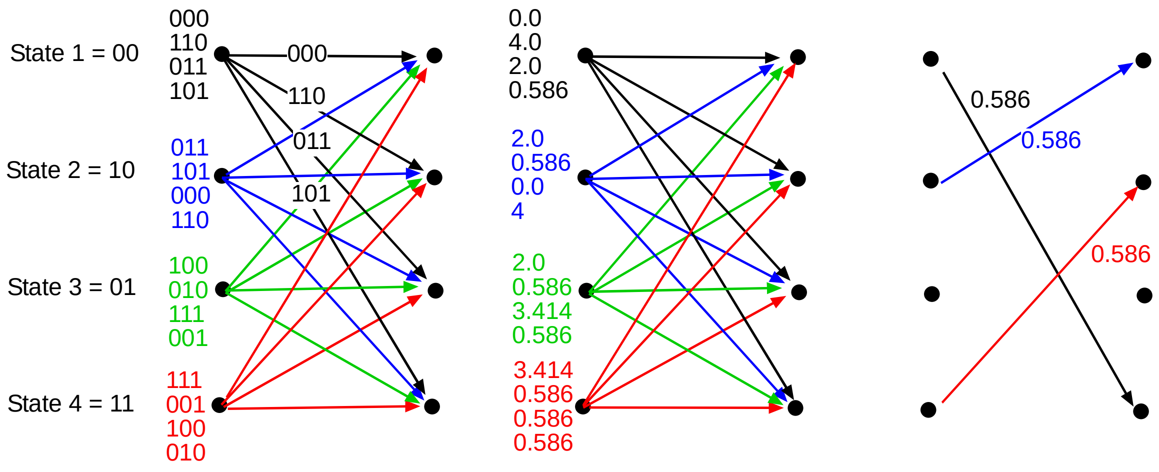


Figure 16 – Euclidean distance change if the symbol to bit mapping is changed.

Ungerboeck's brain storm

Ungerboeck had an idea to improve upon this code by a mapping the signals in a special way called set partitioning. The basic idea is to map k information bits to 2^{k+1}constellation points such that we can limit the transitions to occur only along the largest SED. As the subsets are partitioned, the signals get further apart increasing the Euclidean distance between the signals in that set.

The Figure below shows this process. 8PSK signal constellation is partitioned (using lattice structure and terminology). We have three incoming bits from the code. They are

b_1 b_2 b_3

Proceed with bit b_3 and using this bit, proceed down the decision tree as shown below.

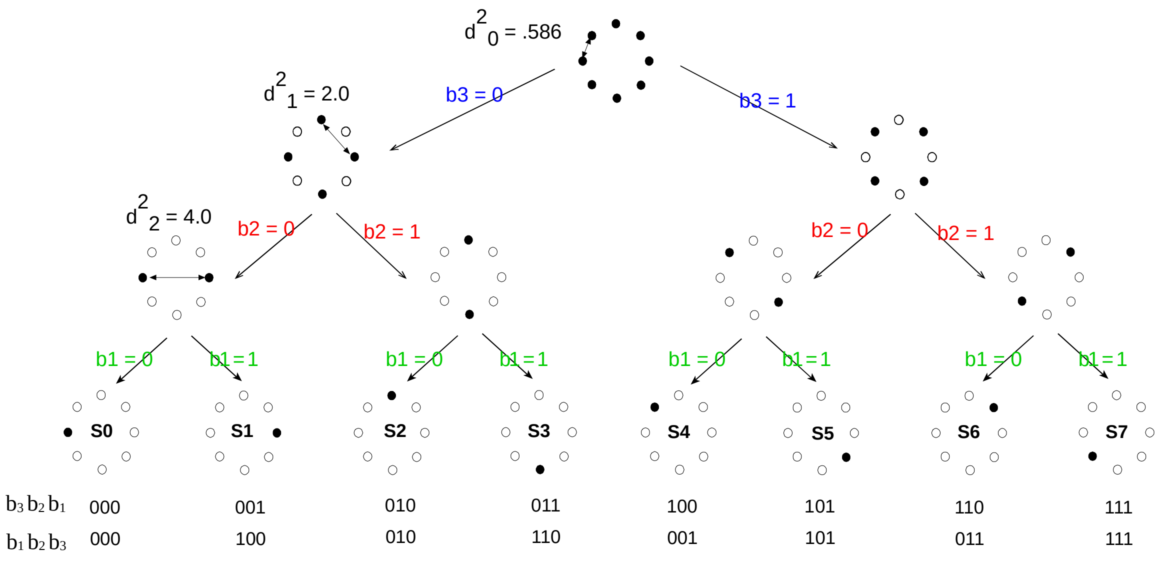


Figure 17 – Partition of a 8PSK constellation to ever increasing Euclidean distance subsets.

This Figure shows how the 8 points of 8PSK are successively portioned into disjoint cosets such that the SEDs are increasing at each level. There are total of four partitions counting the first un-partitioned set. At top-most level, the MSED is 0.586. At the next level, where there are only four points in each of the two cosets, the MSED has increased to 2.0 and at the last level, the MSED is 4.0. Each subset is also called a coset and by the lattice terminology, we can show the partition with its coset generators in this way.

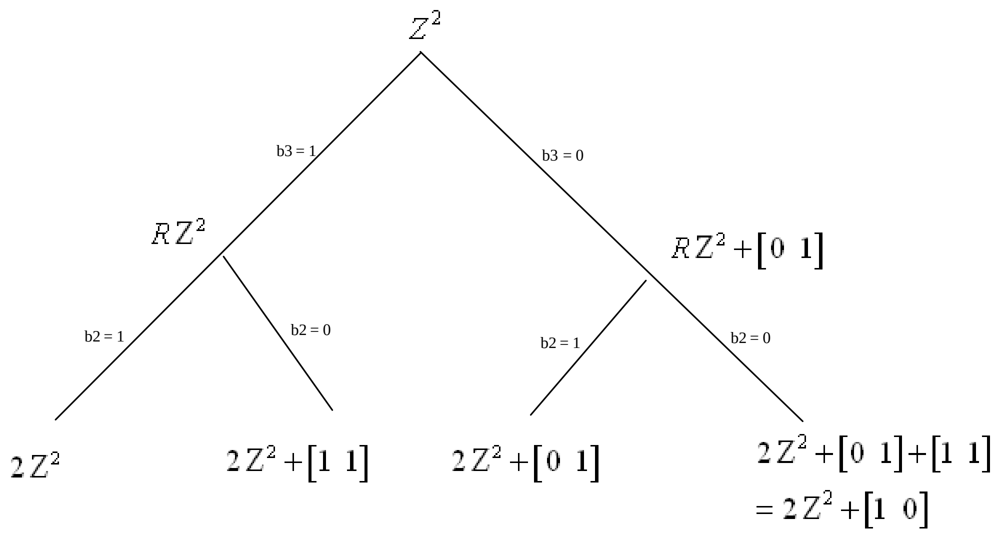


Figure 18 – Lattice structure and coset generators for the 8PSK set partition (Top 3 levels)

Since the top two levels have smaller distances, these errors are more likely, we will use the coded bits to traverse through this part. b_3 to and b_2 which are coded can be used to decide which partition (or coset) to choose and then we can use the uncoded bit at the last level to pick the signal transmitted. The most significant bit b_1, used at the last level, has a large Euclidean distance from its complement and would require an error of 180 degrees to be corrupted.

Example: 010 bits would be mapped to symbol s_2 using this mapping.

Ungerboeck approach leaves the most significant bit uncoded and lets it take care of itself via its large Euclidean distance. This turns out to be the key to larger coding gains of this approach. Only the bits that decide at the top levels with smaller SEDs are coded, thus reducing coding rates and increasing bit efficiency.

OK, so here is how we can do TCM using a lower code rate leaving the MSB uncoded.

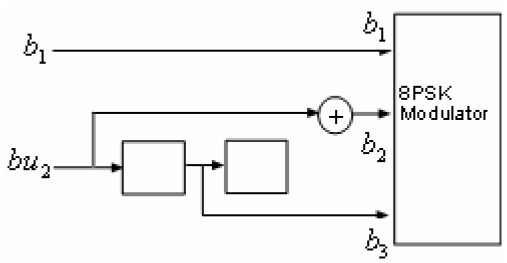


Figure 18 – 4 state rate 1/2 convolutional code as a basis of TCM

Now instead of using a rate 2/3 code on both incoming bits, we leave one bit uncoded and then use a rate ½ code on the other bit. Out come three bits same as with a rate 2/3 code. However, the second approach is more promising since a rate ½ code has a better coding gain.

This is TCM at its most basic, to selectively code only some of the bits and take advantage of the increasing Euclidean distances obtained by set partitioning, then leaving uncoded those bits that are protected naturally by their large SED.

We start with a four state encoder. First we draw the trellis of the rate ½ convolutional coder. As we did for the rate 2/3 code, this trellis is converted to squared distances and then a minimum length path is identified.

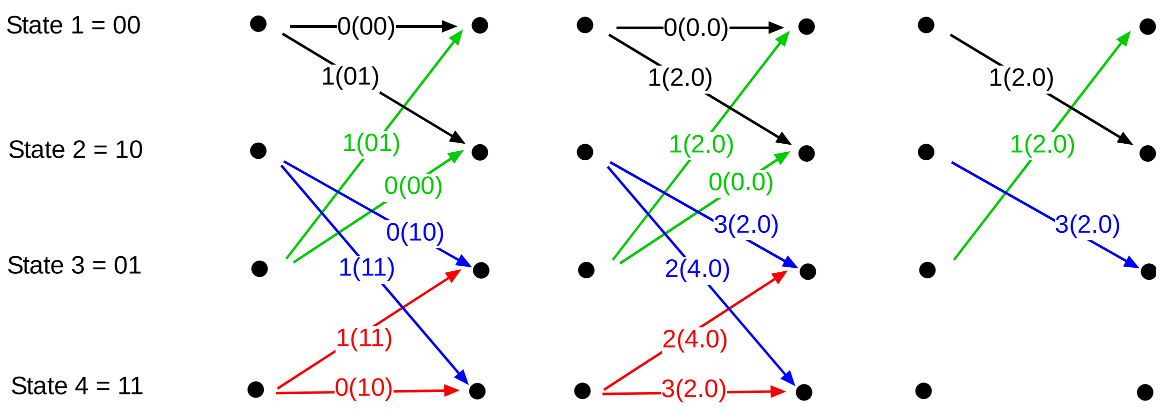


Figure 19 – Trellis of the rate 1/2 code (one bit in, 2 bits out)

The minimum length path is labeled as 01 10 01. The corresponding distances from the 00 bit symbol are 2.0, 2.0, 2.0, the sum of which is 6.0 The MSED for this code is 6. The coding gain is

$$10\log\left(\frac{6}{2}\right) = 4.77dB$$

This is better than the 3.6 dB gain we had from the rate 2/3 code, so this looks promising. But not so fast! This is just the coding gain of the rate 1/2 code. What about the uncoded bit. We have not accounted for that.

Modifying the code trellis to account for the uncoded bit

Now comes the complicated part. We want to modify this trellis to show the effect of the uncoded bit. We have 3 bits going into the modulator, b_1, b_2, b_3. b_2 and b_3 are the result of coding. b_1 is uncoded. At each state we have two coded bits incoming as well as one uncoded bit, so each path doubles to account for the two choices for the uncoded bit. At state 00, if coded bits are 10, then we can get either 110 if uncoded bit is 1 or 010 if it is 0. This doubling of choices is called **parallel transitions**. (This is of consequence only in a 4-state code.)

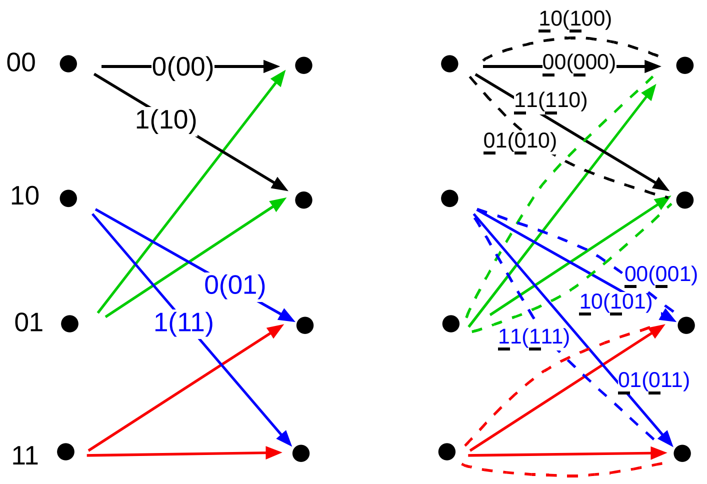


Figure 20 – (a) Code rate 1/2 trellis, (b) code rate 1/2 trellis modified to include the third uncoded bit.

Modify the whole trellis to incorporate parallel transitions at each state. Now we draw the trellis for the TCM which has both coded and uncoded bits. At each junction, we now have an option of getting either a 0 or 1 uncoded bit in addition to the 2 coded bits. Each trellis path is now doubled. The dashed lines in Figure 20, 21 indicate the uncoded bit is a 0 and the solid line means the bit is 1. Now there are 3 bits inside the parenthesis which is symbol. The first bit is the uncoded bit and the last two are the coded bits from the top trellis.

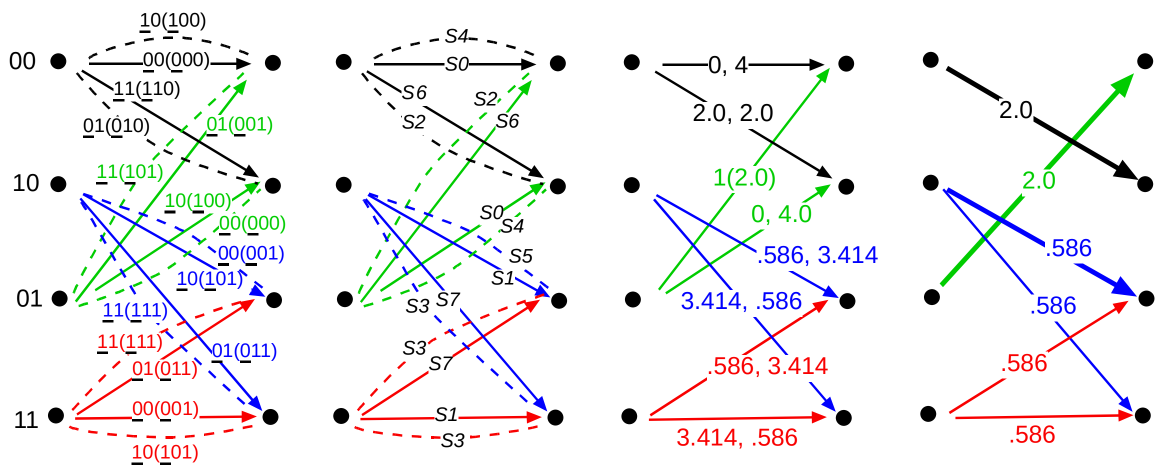


Figure 21 – (a) Modified trellis for the uncoded bit , (b) replace each bit combination by chosen symbol map,)c) for each path, pick the one with the lowest Euclidean distance to symbol S_0, (c) Find shortest path that starts at state 00 and then remerges back (in dark lines)

A parallel decision making process occurs in decoding the incoming signal. First the decoder makes a decision about the coded bits and then the uncoded bit, sending it down one of the four paths at each state. Even if the decoded decision is correct, there would still be a possibility that the uncoded bit will be decoded incorrectly. Now we need to determine which of these two independent events i.e. the error probability of the coded vs. the uncoded bits is larger. The d_{free} of this code is

d_{free} = 2 + .586 + 2 = 4.77

Now look at the parallel paths at t = 0, state 00. We have two parallel pairs, 100 and 000 pair and the 110 and 010 pair. If you check these against the set partition of Figure 17, (Note that in Figure 17, the bit order is $b_3\, b_2\, b_1$ whereas in the trellis is it $b_1\ b_2\ b_3$!) Each pair is 180 degree phase shift apart. But this corresponds to a MSED of 4.0, and it is smaller that the sequence SMSED (4.77), this error is more likely. In other words, it is more likely that the uncoded bits will be decoded incorrectly that the two coded bits, simple because they have a larger SMSED. Of course, this is not always true in TCM, This is also called a **single stage error**.

$\delta^2_{\min}$ = 4.0 (not $d_{\min}$, but just the SED at the bottom level of the partition.)

The terminology is: δ^2_{free} is the sum of the MSED of the coded bits and $\delta^2_{\min}$ is distance at the lowest level which contains the uncoded bit. It is the smaller of the two that determine the overall performance.

$$d^2_{free} = \min\left[\delta^2_{free}, \delta^2_{\min}\right] = \min\left[4.77, 4.0\right]$$

We take the minimum of these two and then divide by $d^2_{\min/uncoded}$ to get the coding gain. $d^2_{\min/uncoded}$ is equal to 2.0 for QPSK.

Coding gain = $10\log\left(\frac{4}{2}\right) = 3dB$

This is not as good as 3.6 dB gain we got when we used the rate 2/3 code alone, but we did get this gain without any increase in bandwidth or symbol rate, which is a remarkable thing.

The reason why this number is low is because this a limiting case due to single state errors and this limits the coding gain to 3 dB. No matter what code we use, as long as we have parallel transitions, the coding gain is limited to 3 dB. With four state convolutional code, we cannot do any better than that. It is our limiting case.

So now we have to admit that although we computed the coding gain of the rate ½ code as 4.77 dB, we are not going to see this gain in TCM because it is trumped by the smaller Euclidean distance of the uncoded bit. We need to do better than that. We can do that by eliminating the parallel transitions. One way to eliminate parallel transitions is to increase the number of states. This way we can assign paths so that they do not have parallel transition.

Figure below shows the redone trellis for the same code but one that has 8 states instead of 4. (This is done by increasing the number of memory registers from 2 to 3.)

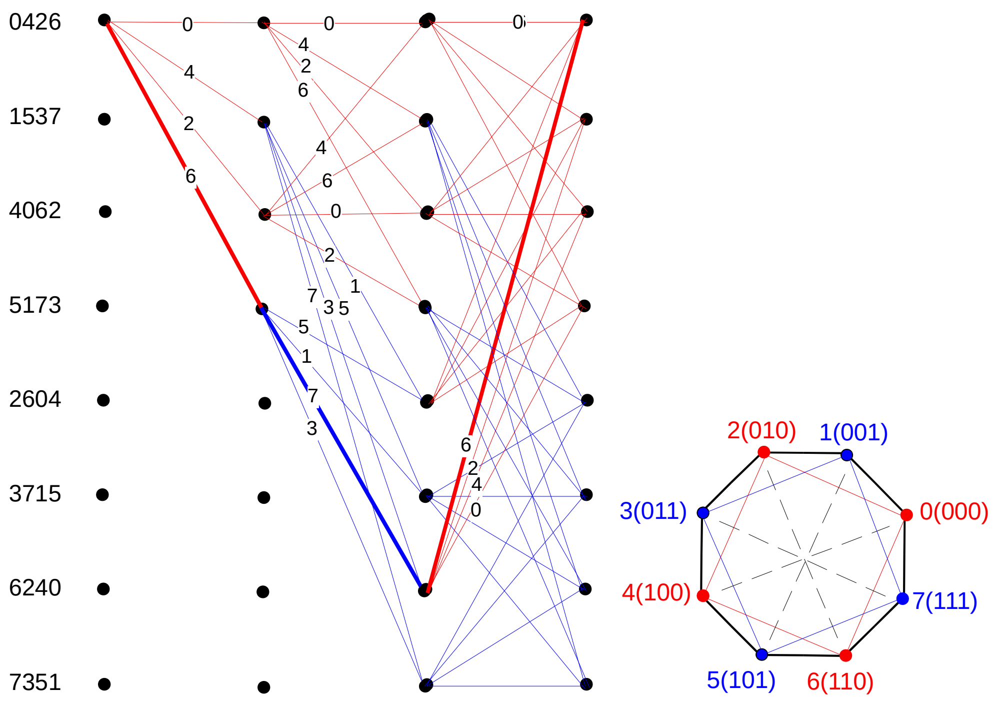


Figure 22 – Trellis of a code of rate 1/2 with eight states (b) bits to symbol mapping

With more states available, we assign the four paths of Figure 21 to other states so there are no parallel transitions. In the Figure above, the left side lists the symbol numbers possible at each state. At state 0, we take path 6, because it has the smallest distance, from there we go to symbol 7 and then symbol 6 again. Without proving, we state that this path has the smallest SED. The sum of each of the distances is

s_0 to s_6 = 2.0, s_0 to s_7 = .586, s_0 to s_6 = 2.0

2 + .586 + 2 = 4.586 and now the coding gain is

10 log (4.586/2) = 3.7 dB

This is an improvement over the case of four states which had a coding gain of only 3 dB and also an improvement over the rate 2/3 code by itself. More states improve this yet further. Here is a list of coding gains possible as number of states is increased still further. We conclude that any TCM with number of states greater than 4 can beat the performance of using the convolutional code alone.

Table I – Coding gain vs. number of states in a 8PSK TCM

Number of States	Coding Gain, dB
4	3.0
8	3.6
16	4.1
64	4.8
128	5
256	5.4
512	5.7

The examples have all been for 8PSK. But in TCM we can we can start with a 8PSK and then go to 16QAM as the final signal or any other such doubling. In V.32 modulation a constellation of 32 points is used.

To examine how 8PSK to 16QAM would work, we take 3 incoming bits.(u stands for uncoded.)

ub_1 ub_2 ub3

Of these, we would code bits ub_2 and ub_3 with rate 2/3 encoder to get three bits, and now we have

ub_1 cb_2 cb_3 cb_4

Starting with cb_4, we would then select a signal using the partition in Figure 23.

Example: we have 1011, this would map to symbol s_{11}. Note that the last mapping is done based on the first uncoded bit. It has a ED which is 16 times larger than at the first level and hence offers good error protection on its own.

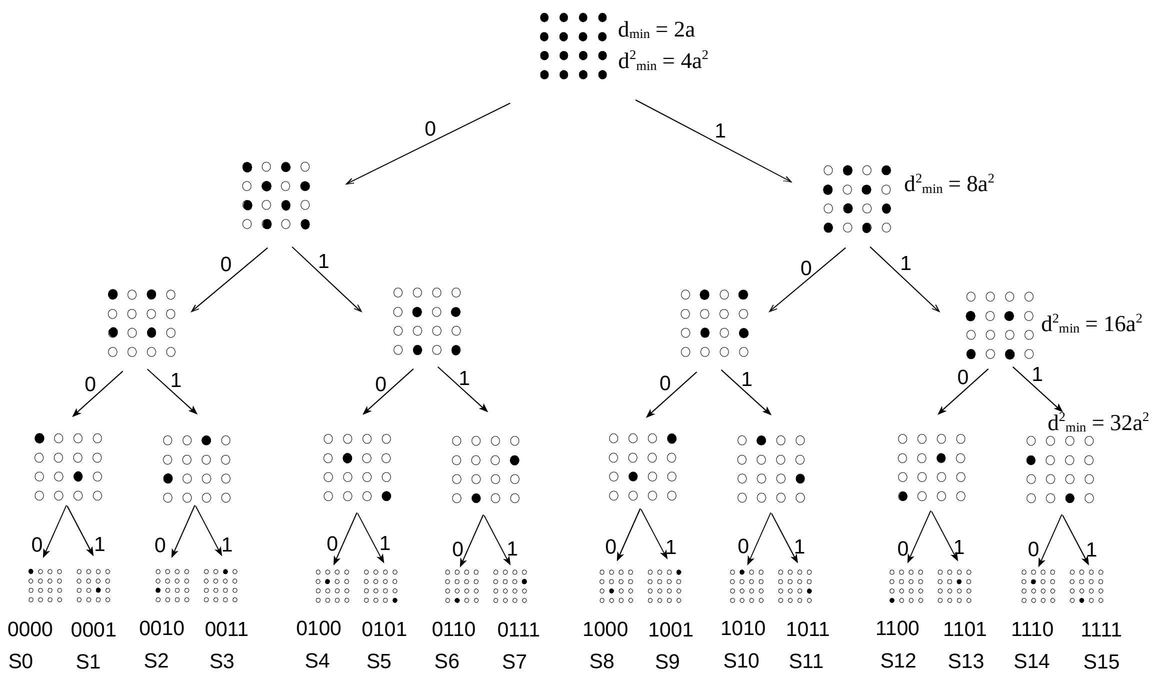


Figure 23 – 16 QAM partition

If the at the lowest distance is 2a with SED equal to $4a^2$, then at the next level it is $8a^2$, then $16a^2$ and then $32a^2$ and so on, doubling each time. Compare this to the SED of QPSK to which we would compare this to determine the coding gain. The coding gain based on this would be 4.4 dB A list of coding gains possible for 16QAM are given below.

Number of states and coding gain for 16QAM

States	Gain, dB
4	4.4
8	5.3
16	6.1
64	6.8
128	7.4

The Pragmatic TCM

Each version of TCM as created by Ungerboeck, requires a different rate code. To avoid this problem, Viterbi (see Ref 5) suggested the use of the 64 state standard rate ½ convolutional code which had been a de facto standard in communications for quite a while. The other advantage of using this “off-the-shelf” code is that with minor modification it can be used to decode both the coded and the uncoded bit. The use of this code and not the custom codes of the original concept is called the **Pragmatic Approach**. Pragmatic, meaning practical but not necessarily optimum, because you use what you have available. Viterbi algorithms can be used to decode the pragmatic TCM so this approach is popular, although it is not as effective as the set-partitioning approach of Ungerboeck.

The pragmatic approach uses the common ½ rate coder (which can be punctured to provide all other rates.) and can create the QPSK, 8PSK and 16QAM signal TCMs.

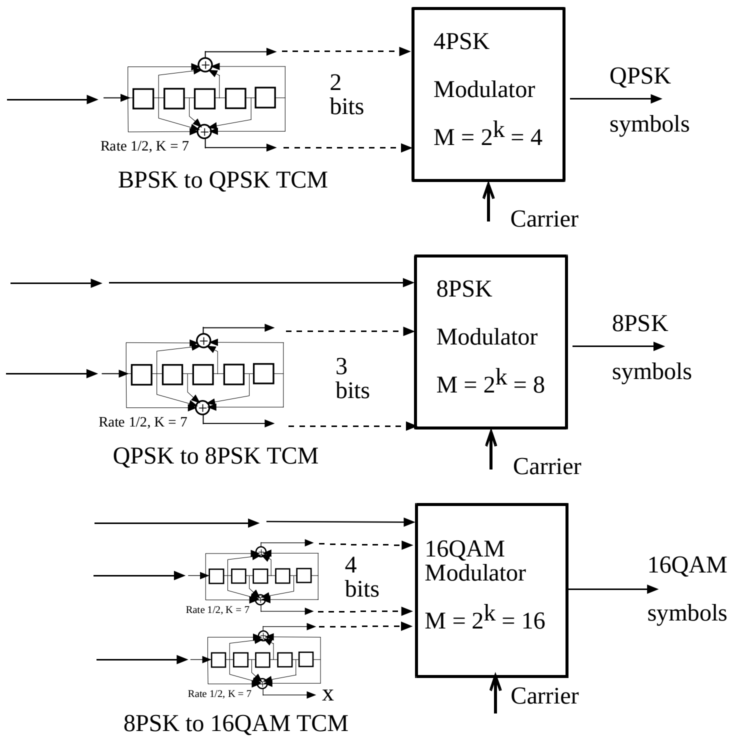


Figure 24 - Pragmatic TCM uses a standard rate 1/2 convolutional code with modified Viterbi decoding.

Decoding using the Viterbi Algorithm

Viterbi algorithm uses a metric and tracks this metric for several trellis paths at once. The path with larger metric is dropped when it merges with another. In hard-decision Viterbi decoding, this is done using the Hamming distance as a metric. In TCM the decoding is done with soft-decision algorithm and Euclidean distance is used as the metric. The objective is to track n possible sequences, keep track of cumulative MSEDs. When paths merge at a state, follow only the one with the smallest metric.

Multi-dimensional TCM

BPSK, PAM are one dimensional modulations. QPSK is a two dimensional modulation composed of two BPSK signals in quadrature. 8PSK similarly is also a 2D modulation. The TCM we have been talking about so far is called 2LD-MPSK-TCM, with L = 1, a 2LD modulation with a dimensionality factor L = 1.

A method of increasing the dimensionality of TCM (L > 1) signals was proposed by Wei. (See really excellent paper on this by Pietrobon, Ref. 4) The term L denotes L dimensions of 2D MPSK signals. With L = 1, we transmit just one TCM symbol. With L = 2, we transmit 2 symbol, so that the number of symbols transmitted is equal to L. Another way multi-D is referred is by L x MPSK. So if we say 3 x MPSK, that is 3 symbols, and a 4 x MPSK is four symbols. All of these are 2D signals so, the total dimension is 2L. The 4 x MPSK is also called 8D TCM.

The main concept in multi-dimensionality is increasing the number of symbols created in one processing period. The transmitted symbols are generated together and this co-generation creates dependence and allows better performance. The term multi-dimensionality does not mean anything other than a form of multi-processing. We do not have a multi-dimensional signal in the usual sense here. In fact typically multi-dimensionality implies independence and here we have the opposite. So the term is confusing. It should have been called higher-order TCM instead.

The advantage of multi dimensional TCM are

1. We can transmit fractional information rates. Instead of the effective code rate being 2/3 as it is in 1 x 8PSK, here it can be higher. We can reduce the code overhead by effecting more than one symbol so we can use code rates like 5/6, 8/9 and 11/12.

2. Better bit efficiency is possible. We define bit efficiency as number of input information bits divided by the number of symbols transmitted in one processing period.

3. Smaller Peak to average ratio –The peak to average ratios are seen to go down for these signals since random coherence problems are lessened.

4. No additional hardware complexity, we can use standard rate ½ codes (punctured) with standard decoding. Multi-D TCM typically uses the pragmatic version of TCM.

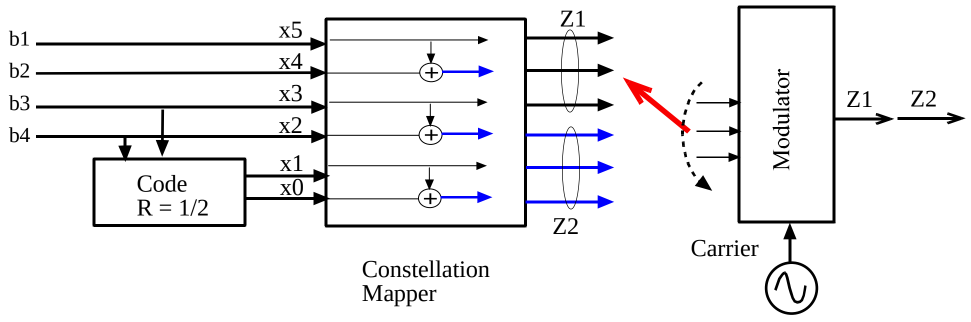


Figure 25 – 2 x 8PSK, 4 input bits, 2 remain uncoded, 2 are coded to produce 4 bits for a total of 6 bits going into the constellation mapper. The constellation mapper uses a algorithm to reorder these bits to output 2 symbols.

This Figure shows the conFigureuration for generating a 2 x 8PSK signal. 4 bits come in. 2 of these go into a rate ½ encoder and generate 2 parity bits, for a total of 6 bits. These six bits are then mapped in a special way by the constellation mapper. It is this mapping function that creates the symbol inter-dependency. The effective code rate is 2/3 and the bit efficiency as defined by number of information bits per symbol is 2 bits per symbol.

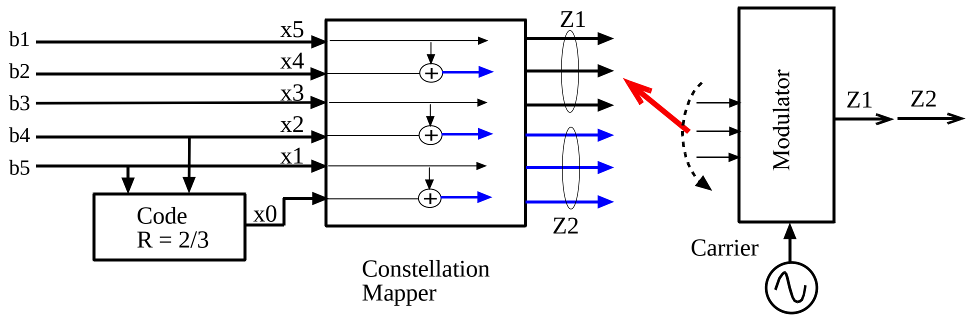


Figure 26 - 2 x 8PSK bit efficiency = 2.5 bits per symbol, code rate of 5/6

By playing with code rate, such as a rate 2/3 encoder in Figure 26, with five input bits producing the 2 symbols, we can change the effective code rate. In example of Figure 26, the code rate is 5/6 and bit efficiency is 5/2 = 2.5 bits per symbol.

The mapping function of the constellation mapper is bit more complex than in a 1 x 8PSK. We do not just take the six incoming bits and map them sequentially to the two symbols. There is processing going on inside this box. We will now examine how this mapping is done.

Example: Constellation mapping of a rate 2/3 2 X 8PSK TCM (L = 2)

Now imagine that we have two parallel 8PSK symbol producers. Each is numbered from 0 to 7. If we transmit a pair of symbols, then there are total of 64 possible pairs. Any one of these pairs can be mapped to the 6 input bits.

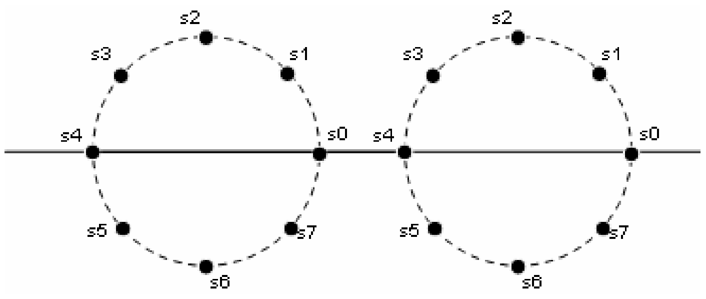


Figure 27 – A pairs of symbols are being produced in one processing period in a 2 x 8PSK (or also 4 D TCM)

We write the output vector of two symbols as

$$Y == \begin{bmatrix} Symbol_1 \\ Symbol_2 \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} y_1^2 & y_1^1 & y_1^0 \\ y_2^2 & y_2^1 & y_2^0 \end{bmatrix}$$

Table II – 2 x 8PSK signal set partitioning

(a) Top level

00	01	02	03	04	05	06	07
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37
40	41	42	43	44	45	46	47
50	51	52	53	54	55	56	57
60	61	62	63	64	65	66	67
70	71	72	73	74	75	76	77

(b) First partition

00	01	02	03	04	05	06	07
10	11	12	13	14	15	16	17
20	21	22	23	24	25	26	27
30	31	32	33	34	35	36	37
40	41	42	43	44	45	46	47
50	51	52	53	54	55	56	57
60	61	62	63	64	65	66	67
70	71	72	73	74	75	76	77

In this table II we list all 64 possible pairs of symbols. Let's take the two neighbor points (00) and (01). These two sets of symbol sequences are about as close as two sequences can get. The MSED between them is .586. (The distance between the first symbols 0 and 0 is 0.0, the distance between the second symbols 0 and 1 is .586 for a total of .586) This is the smallest distance at this level where all pairs are possible.

Now partition top level (a) into two cosets as shown in two different colors in Table II (b). There are two cosets in this partition. One coset can be transformed into the other by coset generator g_0 = (0, 1). If you take any symbol from the one set and if you add 0 to the first symbol and 1 to the second, you will shift position the symbol from one coset to the other. If you examine the sequence distance of points in the colored coset (b) you see that they are now further apart than at the top level. For points 00 and 11, which are two closest neighbors, the sum of MSED is (.586 + .586) = 1.172 at this level.

(c)

00		02		04		06	
	11		13		15		17
20		22		24		26	
	31		33		35		37
40		42		44		46	
	51		53		55		57
60		62		64		66	
	71		73		75		77

(d)

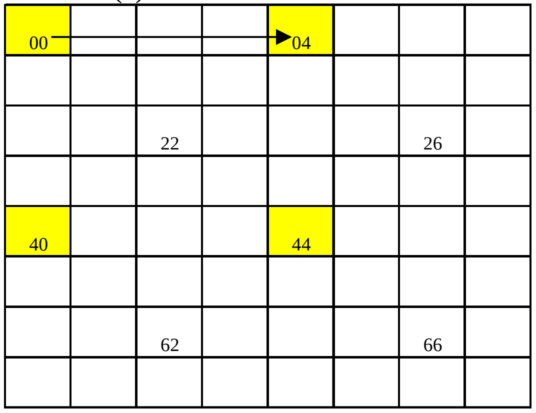
00		02		04		06	
20		22		24		26	
40		42		44		46	
60		62		64		66	

Now partition again. We again get two cosets, related by the coset generator g1 = {1, 1). as in (c). The distance between the points in the new coset has increased again. Now the two nearest points are 00 and 02. The distance between these is 2.0.

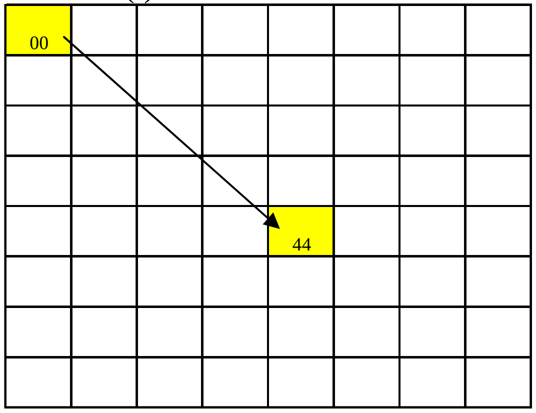
Now partition the shaded coset again, we get Partition (d). These points are now 4.0 apart. Now the two closest points are 00 and 22. The SMSED is (2 + 2) = 4.

We do this two more times to get to an individual signal. At the last level, the points are (4 + 4) = 8 units apart. We partitioned the set six times to get to a single signal.

Partition (e)



Partition (f)



How the math works

All of this is easily coded into a chip using simple math. Let’s take at look at each of the coset generator which is making decisions about which assignment to make based on the received bit. We have six incoming bits (after all incoming bits have been coded) and we have six partitions in response to each of these bits. Let’s write down all the coset generators as we will need them,

$$g0=(0,1),\quad g1=(1,1)$$
$$g2=(0,2),\quad g3=(2,2)$$
$$g4=(0,4),\quad g5=(4,4)$$

Now we take the six input bits and put them through this transformation to come up with the coded and related symbols. Take for example an incoming vector x = 010101 in form of

$$=x^5,x^4,x^3,x^2,x^1,x^0$$
$$=(0,1,0,1,0,1)$$

We multiply each bit by its coset generator to determine which coset it falls in. We get (0, 7) if math is done mod(8).

$$=x^5(4,4)+x^4(0,4)+x^3(2,2)+x^2(0,2)+x^1(1,1)+x^0(0,1)$$
$$=(0,7)$$

This looks a bit incomprehensive until you do it in binary form. Let's rewrite the coset generators in binary form by replacing the ordinal number with binary equivalents.

$$g_5=\begin{bmatrix}1&1\\0&0\\0&0\end{bmatrix}\; g_4=\begin{bmatrix}0&1\\0&0\\0&0\end{bmatrix}\; g3=\begin{bmatrix}0&0\\1&1\\0&0\end{bmatrix}$$

$$g_2 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} g_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} g_0 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

The output two symbols and their constituent bits are given by

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} y_1^2 & y_1^1 & y_1^0 \\ y_2^2 & y_2^1 & y_2^0 \end{bmatrix} \text{ or}$$

$$Y = (Y_1, Y_{2)}) = \left[(y_1^2, y_1^1, y_1^0), (y_2^2, y_2^1, y_2^0) \right]$$

Each bit falls in a particular coset and the total vector is sum of each bit times its corresponding coset generator. There are six generators so index goes from 0 to 5.

$$Y = (y_1, y_{2)} = \sum_{0 \le p \le 5} x^p g_p$$

For the example bit sequence for (0,1,0,1,0,1), x^5, x^3, and x^1 are zero. Crossing these out, we get

$$= \cancel{x^5} \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \oplus x^4 \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \oplus \cancel{x^3} \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 0 \end{bmatrix} \oplus x^2 \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \oplus \cancel{x^1} \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \oplus x^0 \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$= x^4 \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \oplus x^2 \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \oplus x^0 \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$= 1 \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \oplus 1 \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \oplus 1 \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \text{ or}$$

$$Y = \left[(000)\ , (111) \right]$$

and this is the same as

$Y = \left[0, 7 \right]$ in ordinal numbers. So for that given input sequence two output symbols are s_0 and S_7.

If you take the following equation and put it in a picture form as in Figure 27, you will see that the symbol mapper is doing just this math.

$$= x^5 \begin{bmatrix} 1 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \oplus x^4 \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \oplus x^3 \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 0 \end{bmatrix} \oplus x^2 \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \oplus x^1 \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 1 \end{bmatrix} \oplus x^0 \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$= x^5, x^3, x^1 \quad (x^5 + x^4), (x^3 + x^2), (x^1 + x^0)$$

The last equation says that symbol 1 consists of bits 5, 3 and 1 and symbol consists of sums of all the bits. This is same as the constellation mapper shown in Figure below. We can see this function as a kind of re-ordering or interleaving.

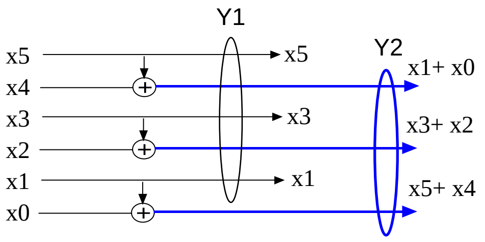


Figure 27 - What the constellation mapper is doing for 2 x 8PSK TCM to the 6 incoming bits to produce the output symbols.

As we increase L, the constellation mapping gets more complex.

The free distance of a multi-D TCM can be increased by both increasing the number of states or by increasing the number of dimensions. We determine the performance the same way as before by determining d_{free} of the code trellis. Extensive list of Asymptotic Coding Gain (ACG) for a L x 8PSK for various L can be found in Ref 1..

3 x 8PSK constellation mapper

In this 3 x 8PSK TCM there are 8 input bits and it puts out 3 8PSK symbols, its efficiency is 8/3 bits per symbol and its code rate is 2/3.

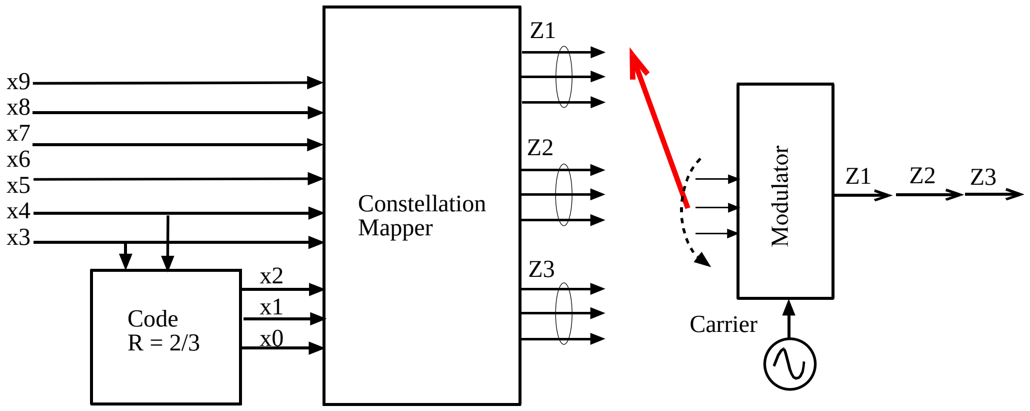


Figure 28 - 3 x 8PSK TCM, producing three symbols at a time.

A 3 x 8PSK generates three independent 8PSK symbols to pick from. The total number of input symbols, three at a time, are 8 x 8 x 8 = 512. This partition will require 9 levels, since ($2^9 = 512$) This is too complex to draw so I will just give the list of set coset generators, of which there are 9.

$$g0 = (0,0,1) \quad g1 = (0,1,1) \quad g2 = (1,1,1)$$
$$g3 = (0,0,2) \quad g4 = (0,2,2) \quad g5 = (4,4,4)$$
$$g6 = (2,2,2) \quad g7 = (4,4,0) \quad g8 = (0,4,4)$$

The three symbols are given by expressions

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \begin{bmatrix} y_1^2 & y_1^1 & y_1^0 \\ y_2^2 & y_2^1 & y_2^0 \\ y_3^2 & y_3^1 & y_3^0 \end{bmatrix}$$

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = x^8 \begin{bmatrix} 0 \\ 4 \\ 4 \end{bmatrix} + x^7 \begin{bmatrix} 4 \\ 4 \\ 0 \end{bmatrix} + x^6 \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix} + x^5 \begin{bmatrix} 4 \\ 4 \\ 4 \end{bmatrix} + x^4 \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix}$$
$$+ x^3 \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} + x^2 \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + x^1 \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} + x^0 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$= (4x^5 + 2x^6 + x^2) \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + (4x^8 + x^4 + x^1) \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} + (4x^7) \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + (2x^3 + x^0) \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

or alternately in mod(8) math

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \begin{bmatrix} 4x^7 + 2x^6 + 4x^5 + x^2 \\ 4x^8 + 4x^7 + 2x^6 + 4x^5 + 2x^4 + x^2 + x^1 \\ 4x^8 + 2x^6 + 4x^5 + 2x^4 + 2x^3 + x^2 + x^1 + x^0 \end{bmatrix}$$

If input bits are 101011001, then we get

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 4+4+2 \\ 4+2+2+1 \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \\ 1 \end{bmatrix} \bmod(8)$$

$$= \begin{bmatrix} 100 \\ 010 \\ 001 \end{bmatrix} \bmod(2)$$

Note that there can be more than one way to subdivide so the above list is not unique.

4 x 8PSK or 8D TCM

NASA uses a 4 x TCM (also called 8D) for some deep space links. A 4 x 8PSK would have 4 8PSK constellation and as such would have 64 x 64 = 4096 total possible signals. This will require 12 levels of partitions since (2^{12} = 4096) Four symbols are generates in one period. By adjusting the number of bits coming out of the encoder, we can create fairly large bit efficiencies and consequently reduce the overhead rate.

The following Figure shows using a code of rate ¾ , so for this case, the bit efficiency would be 8/4 = 2.0 bits per symbol.

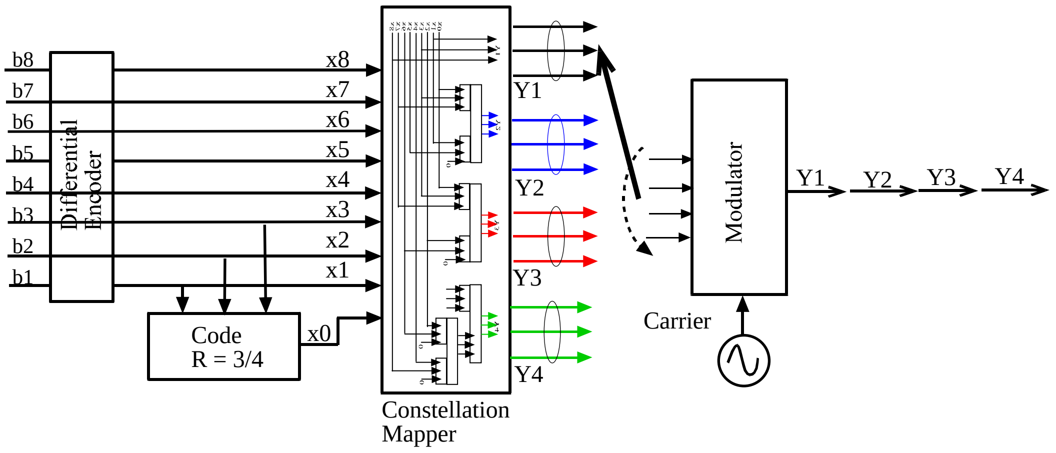


Figure 29 - 4 x 9PSK TCM

To determine the constellation mapping, again we look at the coset generators. Since there are total of 12 levels in the 4 x MPSK partition, we have a much deeper partition. The coset generator which we need to map the input bits to out bits are given here for one case. Others are also possible.

$$\begin{array}{lll} g0=(0,0,0,1) & g1=(0,0,1,1) & g2=(0,1,0,1) \\ g3=(0,0,0,2) & g4=(1,1,1,1) & g5=(0,0,2,2) \\ g6=(0,2,0,2) & g7=(0,0,0,4) & g8=(2,2,2,2) \\ g9=(0,0,4,4) & g10=(0,4,0,4) & g11=(4,4,4,4) \end{array}$$

The SED at the lowest level is 16 and the transformation matrix is given by

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \left[(4x^8 + 2x^5 + x^1)\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} + 4\begin{pmatrix} 0 \\ x^7 \\ x^6 \\ x^7 + x^6 + x^4 \end{pmatrix} + 2\begin{pmatrix} 0 \\ x^3 \\ x^2 \\ x^3 + x^2 + x^0 \end{pmatrix} \right]$$

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \begin{pmatrix} 4x^8 + 2x^5 + x^1 \\ 4x^8 + 4x^7 + 2x^5 + 2x^3 + x^1 \\ 4x^8 + 4x^6 + 2x^5 + 2x^2 + x^1 \\ 4x^8 + 4x^7 + 4x^6 + 2x^5 + 4x^4 + 2x^3 + 2x^2 + x^1 + 2x^0 \end{pmatrix}$$

If input bits are 110011001, then output bits are

$$\begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = \begin{pmatrix} 4 \\ 4+4+2 \\ 4+4 \\ 4+4+4+2+2 \end{pmatrix} = \begin{pmatrix} 4 \\ 2 \\ 0 \\ 0 \end{pmatrix} \mathrm{mod}(8)$$

$$= \begin{pmatrix} 100 \\ 010 \\ 000 \\ 000 \end{pmatrix} \mathrm{mod}(2)$$

Phase Invariance and TCM

TCM is always used with differential encoding. Differential encoding allows recovering from phase ambiguity. Take a BPSK signal. Let us say that a phase shift of 180 degrees occurs, the received bit is now 0 instead of 1. How is the receiver to know that it was a 1? There is no way, unless differential encoding/decoding is used.

Phase lock is a critical problem in demodulation. A 8PSK signal has 45 degrees phase shifts. When the signal is acquired, it is best if any phase shift near one of the allowed phases is locked on instead of having to worry about if it is the correct phase. Once a phase lock is obtained, then if the lock was incorrect then all other decoding would be valid but wrong. For example, if symbol 0 is shifted by 45 degrees, then symbol 7 is now demodulated as 000.

Take a gray coded QPSK signal. If one symbol is locked on the loop and the resulting bits are 11 instead of 00, the differential encoder would be able to correct this error. The code in this case would be considered 180 degrees **phase invariant**. If a phase error of 90 degrees occur and bits 01 are confused for 00, then differential decoder which can only correct complements error is unable to correct this. In this case, this code is not phase invariant for 90 degrees. Similarly 8PSKs signal are 90 degrees phase invariant when numbered in natural order but not 180 or 45 degrees phase invariant.

Certain codes when combined with differential encoding, allow recovering from phase shift errors. The sequence demodulated is automatically corrected for the given phase invariance.

TCM sequences are phase lock sensitive. Multi-D TCM codes offer an advantage that they offer phase invariance. Many different phase shifts error can be tolerated and are quickly taken out.

One of the most comprehensive coverage of this material is in Ref, the book by Lin and Costello

References

In doing this tutorial, I relied mostly on the following four references. The first is a book, which I recommend highly. I didn't like its pervious edition but this last edition is great. If you have to buy just one book on coding, this is the one. The 2nd and third are the seminal papers on this by Ungerboeck and the last is an excellent paper on multi-dimension TCM by Steven S. Pietrobon. The last paper is by Andrew Viterbi et all, an excellent paper on pragmatic TCM.

1."Error Control Coding" Second Edition Shu Lin and Daniel J. Costello, Pearson, Prentice Hall

2. Gottfried Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets Part I: Introduction", IEEE Communications Magazine, vol. 25, no. 2, February 1987, pp. 5-11.

3. Gottfried Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets Part II: State of the Art", IEEE Communications Magazine, vol. 25, no. 2, February 1987, pp. 12-21.

4. Trellis-Coded Multidimensional Phase Modulation, Steven S. Pietrobon, Robert H. Deng, Alain Lafanechere, Gottfried Ungerboeck, Daniel Costello, 0018-9448. IEEE Transactions on Information Theory, Vol 36, No. 1, January 1990

5. Andrew J. Viterbi, Jack K. Wolf, Ephraim Zehavi, and Roberto Padovani, "A Pragmatic Approach to Trellis-Coded Modulation", IEEE Communications Magazine, (probably vol. 27, no. 7), July 1989, pp. 12-21.

Other references

Saud A. Al-Semari and Thomas E. Fuja, (abstract of) "I-Q TCM: Reliable Communication Over the Rayleigh Fading Channel Close to the Cutoff Rate", IEEE Trans, on Information Theory, vol. IT-30, no. 6, November 1984, pp. 784-791.

John B. Anderson and Desmond P. Taylor, "A Bandwidth-Efficient Class of Signal-Space Codes", IEEE Trans, on Information Theory, vol. IT-24, no. 6, November 1978, pp. 703-712.

Sergio Benedetto, M. Ajmone Marsan, Guido Masera, Gabriella Olmo, and Z. Zhang, "Encoded 16-PSK: A Study for the Receiver Design", IEEE Journal on Selected Areas in Communications, vol. 7, no. 9, December 1989, pp. 1381-1391.

Ezio Biglieri, "High-Level Modulation and Coding for Nonlinear Satellite Channels", IEEE Trans, on Communications, vol. COM-32, no. 5, May 1984, pp. 1091-1096.

A.R. Calderbank and N.J.A Sloane, "An Eight-Dimensional Trellis Code", Proc. IEEE, vol. 74, no. 5, May 1986, pp. 757-759.

A.R. Calderbank and N.J.A Sloane, "New Trellis Codes Based on Lattices and Cosets", IEEE Trans, on Information Theory, vol. IT-33, no. 2, March 1987, pp. 177-195.

Robert Calderbank and James E. Mazo, "A New Description of Trellis Codes", IEEE Trans, on Information Theory, vol. 43, no. I, January 1997, pages not given.

Pierre R. Chevillat and Evangelos Eleftheriou, "Decoding of Trellis-Encoded Signals in the Presence of Intersymbol Interference and Noise", IEEE Trans, on Communications, vol. 37, no. 7, July 1989, pp. 669-676.

Robert H. Deng and Daniel J. Costello, Jr., "High Rate Concatenated Coding Systems Using Multidimensional Bandwidth-Efficient Trellis Inner Codes", IEEE Trans, on Communications, vol. 37, no. 10, October 1989, pp. 1091-1096.

Dariush Divsalar, "Multiple Trellis Coded Modulation (MTCM)", IEEE Trans, on Communications, vol. 36, no. 4, April 1988, pp. 410-419.

Dariush Divsalar, Marvin K. Simon, and Joseph H. Yuen, "Trellis Coding with Asymmetric Modulations", IEEE Trans, on Communications, vol. COM-35, no. 2, February 1987, pp. 130-141.

Karin Engdahl and Kamil Sh. Zigangirov, "On the Calculation of the Error Probability for a Multilevel Modulation Scheme Using QAM-Signaling", IEEE Trans, on Information Theory, vol. 44, no. 4, July 1998, pp. 1612-1620.

G. David Forney, Jr., Robert G. Gallager, Gordon R. Lang, Fred M. Longstaff, and Shahid U. Qureshi, "Efficient Modulation for Band-Limited Channels", IEEE Journal on Selected Areas in Communications, vol. SAC-2, no. 5, September 1984, pp. 632-647.

G. David Forney, Jr., and Lee-Fang Wei, "Multidimensional Constellations - Part I: Introduction, Figures of Merit, and Generalized Cross Constellations", IEEE Journal on Selected Areas in Communications, vol. 7, no. 6, August 1989, pp. 877-892.

Roger L. Freeman, Reference Manual for Telecommunications Engineering. Second Edition, Wiley Intescience, New York, 1994, pp. 326-328, 1387-1424.

Allen Gersho and Victor B. Lawrence, "Multidimensional Signal Constellations for Voiceband Data Transmission", IEEE Journal on Selected Areas in Communications, vol. SAC-2, no. 5, September 1984, pp. 687-702.

Michael L. Honig, "On Constructing Embedded Multilevel Trellis Codes". IEEE Trans. on Communications, vol. 36, no. 2, February 1988, pp. 218-221.

Rob Howald, "Desperately Seeking Shannon", Communications Systems Design. March 1999, pp. 12-16.

Kamal Jain, Ion Mandoiu, and Vijay V. Vazirani, "The 'Art of Trellis Decoding' Is Computationlly Hard - For Large Fields", IEEE Trans, on Information Theory, vol. 44, no. 3, May 1998, pp. 1211-1214.

Rolf Johannesson and Emma Wittenmark, "Two 16-State, Rate R - 2/4 Trellis Codes Whose Free Distances Meet the Heller Bound", IEEE Trans, on Information Theory, vol. 44, no. 4, July 1998, pp. 1602-1604.

Donald A. Johnston and Stephen K. Jones, "Spectrally Efficient Communication via Fading Channels Using Coded Multilevel DPSK", IEEE Trans, on Communications, vol. COM-29, no. 3, March 1981, pp. 276-284.

James M. Kroll and Nam C. Phamdo, "Analysis and Design of Trellis Codes Optimized for a Binary Symmetric Markov Source with MAP Detection", IEEE Trans, on Information Theory, vol. 44, no. 7, November 1998, pp. 2977-2987.

L. H. Charles Lee, "Combined Convolutional Coding and Modulation", Chapter 7 of Convolutional Coding: Fundamentals and Applications. Artech House, Boston, 1997, pp. 149-167. Also see Chapter 2 on Convolutional Code Structures, pp. 10-55, in the folder TCM-My Notes.

M. Ajmone Marsan, G. Albertengo, and S. Benedetto, IEEE Trans, on Communications, vol. COM-35, no. 9, September 1987, pp. 969-972.

Steven V. Pizzi and Stephen G. Wilson, "Convolutional Coding Combined with Continuous Phase Modulation", IEEE Trans, on Communications, vol. COM-33, no. 1, January 1985, pp. 20-29.

Mischa Schwartz, 'Trellis-Coded Modulation", Section 7-10 of Information Transmission. Modulation, and Noise". 4th edition, McGraw-Hill, New York, 1990.

Desmond P. Taylor and Hing C. Chan, "A Simulation Study of Two Bandwidth-Efficient Modulation Techniques", IEEE Trans, on Communications, vol. COM-29, no. 3, January 1981, pp. 267-275.

Anthony Tehan and John O. Scanlan, "A Simulation Study of Trellis-Coded Modulation for a Satellite Link", IEEE Trans, on Communications, vol. 45, no. 11, November 1997, pp. 1371-1374.

Hemant K. Thapar, "Real-Time Application of Trellis Coding to High-Speed Voiceband Data Transmission", IEEE Journal on Selected Areas in Communications, vol. SAC-2, no. 5, September 1984, pp. 648-658;

Gottfried Ungerboeck, "Channel Coding with Multilevel/Phase Signals", IEEE Trans, on Information Theory, vol. IT-28, no. 1, January 1982, pp. 55-67.

Lee-Fang Wei, "Rotationally Invariant Convolutional Channel Coding with Expanded Signal Space - Part I: 180 degrees", IEEE Journal on Selected Areas in Communications, vol. SAC-2, no. 5, September 1984, pp. 659-671

Error	Symbol	Errored Symbol	Sq. Distance Const1	Sq. Distance Const2
001	000	001	0.586	0.586
001	001	000	0.586	0.586
001	010	011	0.586	0.586
001	011	010	0.586	0.586
001	100	101	0.586	0.586
001	101	100	0.586	0.586
001	110	111	0.586	0.586
001	111	110	0.586	0.586
010	000	010	2	3.414
010	001	011	2	0.586
010	010	000	2	3.414
010	011	001	2	0.586
010	100	110	2	3.414
010	101	111	2	0.586
010	110	100	2	3.414
010	111	101	2	0.586
011	000	011	3.414	2
011	001	010	0.586	2
011	010	001	0.586	2
011	011	000	3.414	2
011	100	111	3.414	2
011	101	110	0.586	2
011	110	101	0.586	2
011	111	100	3.414	2
100	000	100	4	0.586
100	001	101	4	3.414
100	010	110	4	0.586
100	011	111	4	3.414
100	100	000	4	0.586
100	101	001	4	3.414
100	110	010	4	0.586
100	111	011	4	3.414
101	000	101	3.414	2
101	001	100	3.414	2
101	010	111	3.414	2
101	011	110	3.414	2
101	100	001	3.414	2
101	101	000	3.414	2
101	110	011	3.414	2
101	111	010	3.414	2
110	000	110	2	4
110	001	111	2	4
110	010	100	2	4
110	011	101	2	4
110	100	010	2	4
110	101	011	2	4
110	110	000	2	4
110	111	001	2	4
111	000	111	0.586	3.414
111	001	110	0.586	3.414
111	010	101	0.586	3.414
111	011	100	0.586	3.414
111	100	011	0.586	3.414
111	101	010	0.586	3.414
111	110	001	0.586	3.414
111	111	000	0.586	3.414

Error Vector	Min. sq distance Const 1	Const 2
000	0	0
001	0.586	0.586
010	2	0.586
011	0.586	2
100	4	0.586
101	3.414	0.586

Finding the MSED