TPSK31: Getting the Trellis Coded Modulation Advantage

A Tutorial on Information Theory, Coding, and Us with an homage to Claude Shannon and Ungerboeck's genius applied to PSK31.

By: Bob McGwier, N4HY

As just about everyone who is alive and awake and not suffering from some awful brain disease knows, PSK31 is an extremely popular mode for doing keyboard to keyboard communications. Introduced in the late 1990's (1,2) it quickly swept the amateur radio world.

Its major characteristics are a 31.25 baud differentially encoded PSK signal with keyboard characters encoded into an compression algorithm called varicode by Peter (1) before being modulated with the differentially encoded psk. This differential psk has some serious advantages on HF, including insensitivity to sideband selection, no need for a training sequence to allowing break in decoding to occur and myriad others discussed elsewhere. The slow data rate allows it to use our awful HF channels on 40 and 20 meters effectively and it is more than fast enough to have a good conversation just as we have always done with RTTY. Further, it is very spectrally efficient. If we are going to improve on it, we better not mess with these characteristics. We will not.

Peter will be the first person to tell you that he is not a trained communications theorist, engineer, or coding expert. Yet he is one of the great native intelligences in all of amateur radio. His inspirational muse (lend me some!) is one of the best and his perspiration coefficient are among the highest ever measured in amateur radio (this means he is smart and works hard, an extremely good combination).

We will give you a tour of some of this theory, show you how Peter used one piece of it well and then made an attempt to aid his new mode with what in this author's opinion was an abortive effort to help the mode achieve a lower error rate and an attempt to show us another way.

Shannon's Source Coding Theorem:

Claude Shannon's name is spoken with reverence by almost every serious communications theoretician or engineer. He completely revolutionized the field of communications and information theory with a few years of truly inspired work. He is called the father of information theory and that is not too strong a thing to call him.

Since we only care about digital communications systems in this paper we will limit our discussion of the information theory to information theory on discrete random variables. Furthermore, since we are talking about digital systems, we might as well think of our random variables as coin tosses with 0,1 as the outputs or encodings of the information, heads, tails.

Suppose we flip our coin and see what we can figure out. How much information is being transmitted by the flipping of this coin? Let's flip this thing 1,000,000 times (my arm is breaking just thinking about it). Let's write the sequence of results (my arm is really hurting now).

Let suppose I want to call Frank Brickle, AB2KT, who is staying in Vancouver right now and give him the results. How many bits will it take me to encode the results? THIS IS THE ESSENCE OF THE SOURCE CODING THEOREM and the right question. Most great advances in science come from asking the right question. This is no exception. I have information I want to give to Frank. How much information MUST I transfer to get the story across perfectly without loss of information on a perfect channel?

First, let us suppose it is a completely fair coin. The two outcomes happen each with probability ½. The only way for me to transfer the information perfectly to Frank is to tell him 0 or 1 (T or H). I must send the entire sequence of results and it will take 1,000,000 bits. This is going to be tedious. But what do you expect of noise like signals? The only way for me to tell you what the noise signal is in most systems, is to give you every sample. OTHERWISE I COULD EFFECTIVELY SUBTRACT THE NOISE and I would not have as noisy a channel! This is what we do with tone like interferers; we find a way to clobber them with filters or other predictive processing.

But now let us scrape our trusty coin about and really take a hammer to the thing. We dent it so badly, that heads occurs ¼ of the time and tails therefore occurs ¾ of the time on average. The entire sequence of flips can be sent in digital form in a little over 810,000 bits. This says to us that each flip of the coin, now that it is biased, has less information per flip than it did when it was fair. How do we dc this? Let's go further so the answer is more intuitive. Let's just go completely nuts. We will take a ten pound sledge to that poor coin. (I can see Frank shuddering in Vancouver right now!). Now the thing gives me a head only one time in 1000 on average. We expect very few heads and many tails, so does it make sense to transmit every flip? NO WAY.

Start flipping, at the first head, we tell Frank (head after N_1 flips). Keep flipping. At the next heads, tell Frank the distance to the next flip is N_2 flips later. This is a much more efficient way of conveying the information than T,T,T,T,T,T,T,T,T,T,T,T,T,T,….., H (whew at last),T,T,T,T, …,H (wow, another one!). This is well know to image encoders as a variant of RLE or run length encoding.

This is an example of an efficient coding. We were able to do it because the source contained very little information per flip. This sounds like the Digital Radio group on Yahoo$_{tm}$ right? Lots of stuff coming but little information? (Just kidding).

This is one of the ingenuous things Peter Martinez gave us in PSK31. English does not emit letters at the same rate. The small letter **e** occurs most frequently in typed English and therefore contains little information. Martinez, in his Varicode, did exactly this. He said when I see an e, I will send 11. On the other hand, how often do you send a NUL or a DLE (teletype or ascii characters)? NOT VERY OFTEN. So in those cases, he emitted his longest possible codewords, 1010101011 and 1011110111 respectively. Peter has distilled the information for randomness and found an efficient coding for the source emissions. Just like the unfair coin, it would be DUMB to send the 7 or 8 bit representation for e and the same ascii representation for NUL ! Martinez does not do this dumb thing. An additional feature he built into his compression scheme was self synchronization aides. All code words begin and end with a 1 and at a minimum, two zeros are transmitted between characters, and no varicode codeword has two zero's in a row in it. He cleverly combined several features into one device with this scheme.

How can we measure this stuff with mathematics so we can have some guide posts on ***what is possible for my communications system?*** We will go back and consider the coin flip. Suppose the probability of a head is p_h. Since tails is the only other outcome we will consider (standing on edge only counts in the Twilight Zone), the probability of a tail is 1 – probability of a head, $p_t = 1 - p_h$. We already know that for large differences in these probabilities, the information content is low. How can we capture this? We capture it through the mathematical device called ***Entropy***. We will measure the information in bits. The self information in each trial or flip X of the coin X is $I(X)$ where

$$I(X) = \log_2(1/p(X)) = -\log_2(p(X)). \tag{1.1}$$

The self information, measured in bits, in the flip X is the negative log base 2 of $p(X)$. The "bits" here come for the fact that we using logarithm base 2. The information entropy of the random variable X is the expected value of the self information or

$$H(X) = E\left(I\left(X\right)\right) = -p_h \log_2\left(p_h\right) - p_t \log_2\left(p_t\right). \tag{1.2}$$

So in the fair coin trial, $p_t = p_h = \frac{1}{2}$ and the formula gives us that each coin flip gives

$$-\frac{1}{2}\log_2\left(\frac{1}{2}\right) - \frac{1}{2}\log_2\left(\frac{1}{2}\right) = -\log_2\left(\frac{1}{2}\right) = 1. \tag{1.3}$$

That is, 1 bit of information in each trial just as we said. In the ridiculously unfair case, (or the utterances in the Digital Radio group) we get

$$H(X) = -0.001\ \log_2\left(0.001\right) - 0.999\ \log_2\left(0.999\right). \tag{1.4}$$

We get 0.0114 bits because the outcome is so biased. So in our 1,000,000 flip trials, I need to communicate about 11500 bits to Frank. What does this have to do with our communications system, Varicode, PSK31, and Peter? We wanted to give you a practical demonstration of WHY redundant communication channels can be effectively compressed. Because mathematically, they don't have as many bits of information in them as their alphabet would seem to indicate if the letters of the alphabet were issued at random. Peter noticed this about English and designed a compression code.

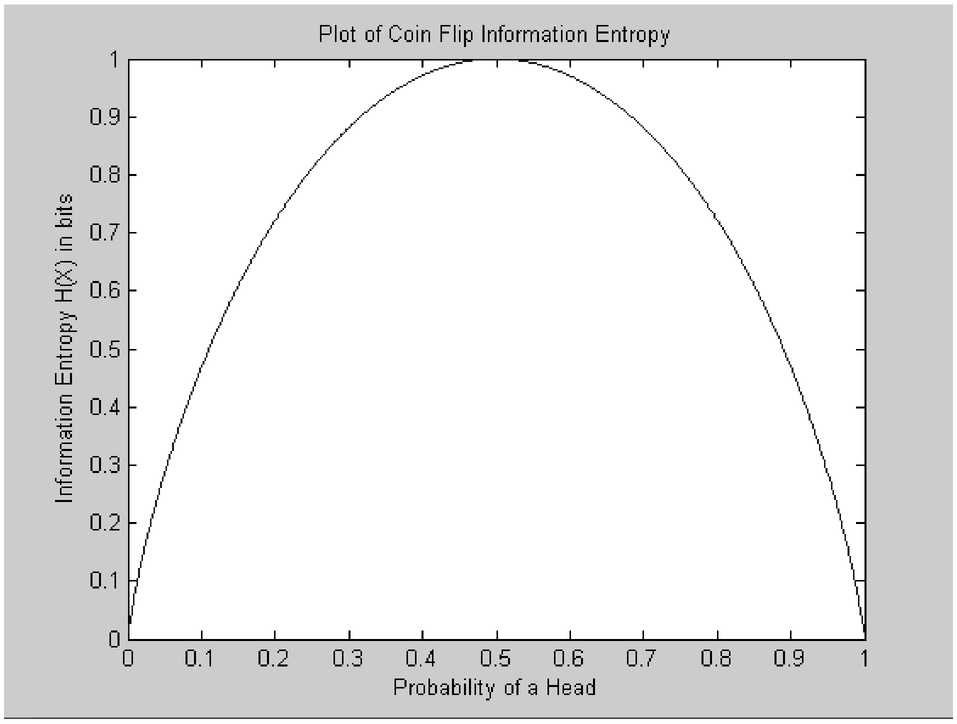


Figure 1: Information Entropy of Coin Flipping

So now we can state a fact. The highest information entropy for an alphabet of size N occurs when? When all members of the alphabet are issued with equal probability, $1/N$. In that case the information entropy is

$$H(X) = \sum_{1}^{N} -\frac{1}{N}\log_2\left(1/N\right) = -\log_2\left(1/N\right). \tag{1.5}$$

This is the maximum information entropy for an alphabet of size N. One of the most famous plots in mathematics, not just information theory, is the plot of the coin toss entropy as a function of the probability of a head is given in Figure 1.

So again, we return to our problem. What does your intuition tell you about transmitting information over a channel? Is it smart to use the entire sequence of heads and tails in the case of the very biased coin? Of course it is not smart. It is smarter to compress the information as much as possible and then use the “extra time” to send the data in such a way that it is more immune to the degradations omnipresent in communications channels. What does Shannon tell us? In one of the most celebrated results of science, in the 1940’s Shannon studied this problem and told us a great fact which we will state in terms of coin flips and then generally.

Shannon's Source Coding Theorem (version 1): If the coin flips have probably of a head p_h then we can compress without loss of information, the sequence of M flips into $MH(p_h)$ where we have used $H\left(p_h\right)$ as shorthand for the full entropy.

WHAT DID WE JUST DO? We compressed the redundant information with this shorthand $MH(p_h)$! So in my transmission to Frank I need to send (using (1.4))

$$1000000*\left(-0.001\ \log_2\left(0.001\right)-0.999\ \log_2\left(0.999\right)\right)\text{bits} \qquad (1.6)$$

In other words, 11408 bits is the LEAST number of bits I can send and hope to get it through without loss of information. Now let us state this theorem a little more generally

Shannon's Source Coding Theorem : *N* i.i.d. random variables each with entropy *H(X)* can be compressed into more than *NH(X)* bits with negligible risk of information loss, as *N* tends to infinity; but conversely, if they are compressed into fewer than *NH(X)* bits it is virtually certain that information will be lost

So we have shown that we CAN compress redundant information out of the data emitted from our source. We argue that we should. It is the responsible thing to do. Peter has done this to the best of his ability with Varicode. The complete table is in Appendix A.

PSK31:

Now that we have crushed the data to be sent into a smaller package, what do we do with it? It is really pretty straightforward. Between each varicode character, we send at least two zeros. No varicode character has two zeros in a row. Each varicode begins and ends with a 1 so the first 1 after a run of zeros is assumed to be the beginning of a varicode symbol. This it is trivial to break in synchronize. If we want to send a zero, we change the phase of our carrier by 180 degrees or π radians. If we send a 1, we make no phase change. So a long string of zero data means we are in idle and we change phase every symbol time.

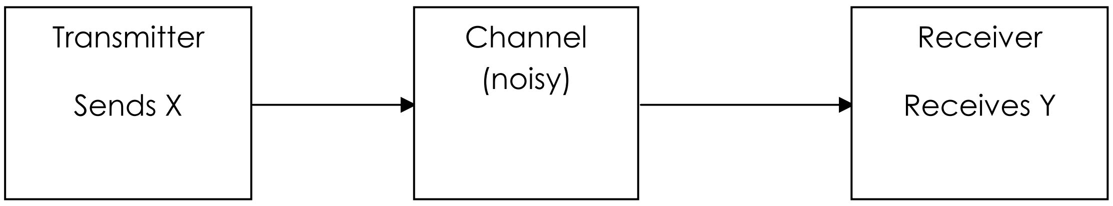
It became immediately apparent how much more spectrally efficient this mode was in relation to baudot RTTY. It required much less power and bandwidth to conduct keyboard to keyboard chat in comparison to RTTY. It has given birth to an entire generation of sound card based digital programs. In other words, it is a marvel and is still immensely popular.

In an attempt to introduce a new wrinkle into the system and try to improve the performance on some channels, Peter introduced QPSK31. It encodes the data stream with a convolutional code and then transmits the encoded data two bits at a time using QPSK. There is the belief in much of the writing about QPSK31 that it is 3 dB worse than PSK31 because half the power is split into two channels and that this can be made up with the convolutional code. There is a failure to recognize that in addition to sending ½ of the power per channel we have another impact. We are not sending them on ***separate channels***. The distance between points of the same voltage or power in the original BPSK we can say is 2. The distance between nearest neighbors IF all points still have the same power as before is now sqrt(2) in the qpsk constellation rather than 2. So this is another loss of $-10\log\left(\sqrt{2}\right)$ or -1.5 dB than is accounted for by the power computation alone. So the loss we need to make up with the coder is not 3 dB, but 4.5 dB. The constraint length 5 convolutional code used is not up to the challenge.

Shannon's Channel Coding Theorem:

Claude Shannon told us to compress the redundancy from the data. We can do this is a lossless way if we use Shannon's formula for the entropy to find the information rate and then design an appropriate code. Now that we have compressed the data, we want to know, can we send it in an efficient manner on the channel we have in front of us on our radios?

There are two forms of this theorem that information theory students learn. One is the Noisy Channel Coding Theorem and the other is the Shannon-Hartley Channel Coding Theorem. The Shannon-Hartley is an application of the Noisy Channel version to the archetypal channel we consider in digital communications. For completeness, we will give the Noisy Channel Theorem and then use only the Shannon-Hartley (5, 6, 7, and 8).



The transmitter sends the symbols X into the channel and the receiver gets Y. Consider X to be a stream of data and Y to be the received version of this stream of data. In all of our systems, we are sending a small finite alphabet. In all the systems we care about now, we are receiving noisy digitized samples. Thus both X and Y are discrete random variables. Since we know something about X immediately, we know how much information the transmitter is putting into the channel. It is $H\left(X\right)$. If we are to find the problem of understanding what the channel is doing to us, we need to make some FALSE but simplifying assumptions. We will assume that the channel is doing nothing but adding Gaussian noise. We will assume the channel does not remember in any way, what went before. The extent to which these assumptions apply to our problems, tells us the extent to which the theory is directly applicable. But under this assumption, we know a lot about Y, the received symbols. The important thing for us to know is what is Y if we assume we know what X is. This is a complete defined mathematical thing because we have assumed that all the channel has done is add noise. Thus, if we know X, we know the probability distribution of Y given that X has occurred. That is we know $P_{Y|X}\left(y\,|\,x\right)$, which is that we will see *y* given that *x* is emitted if we see the random sequence *Y* knowing the sequence *X* has been emitted. But since the channel has known distribution, we actually can easily derive the joint distribution of *y* and *x*, $P_{Y|X}(x,y)$ from the known quantity $P_X(x)$ which is called the prior and using Baye's rule. The prior here means we assume we know the probability that x is emitted by the transmitter. Thus we get

$$P_{X,Y}(x,y) = P_{Y|X}(x,y)P_X(x). \tag{1.7}$$

Under our assumption of discrete random variables, let us consider the all important quantity

$$I(X;Y) = \sum_{y\in Y}\sum_{x\in X} p(x,y)\log_2\left(\frac{p(x,y)}{p(x)p(y)}\right). \tag{1.8}$$

This quantity is known as the mutual information between the discrete random variables X and Y (in bits). Define the conditional entropy as

$$H(Y\mid X) = -\sum_{x\in X} p(x)\sum_{y\in Y} p(y\mid x)\log_2 p(y\mid x). \tag{1.9}$$

Then we get several forms for $I(X;Y)$:

$$\begin{aligned} I(X;Y) &= H(X) - H(X\mid Y) \\ &= H(Y) - H(Y\mid X) \\ &= H(X) + H(Y) - H(X,Y). \end{aligned} \tag{1.10}$$

Where $H(X,Y)$ is defined in exactly the same way as before except we use the joint probability functions for x and y.

We are ready to state the all important theorem and leave it about as quickly as we arrived after we set the plate. The channel in our diagram and in our statements is a thing independent mathematically from the transmitter and its symbols. This means it is a fundamental physical thing and a limiting facto on anything, any system, that wants to use it to communicate. So, if we want to know how much we can cram through this channel, we need to try ALL possible communications systems to find the best. Before you cry over the infinite possibilities, relief will immediately follow.

Noisy Channel Coding Theorem: The capacity of a channel such as the ones we have described is given by

$$C = \sup_{p_X} I(X;Y). \tag{1.11}$$

If $\varepsilon > 0$, for any rate $R < C$, there exists an encoding and decoding that can be used to ensure that the probability of block error is less than ε for a sufficiently long code. Also, at a rate greater than the channel capacity, the block error rate goes to 1.0 as the block length goes to infinity. Here sup means the largest value over all possible transmitter types, data sources, and source codings.

This is one of those painful mathematical theorems. It is difficult enough to get beyond the symbols, the careful use of language, and the concepts. But the you find that it does not tell you one thing about how to do it. It only says “it exists”.

The Shannon-Hartley is probably more useful to most amateurs. It gives us the capacity of the channel in familiar terms. There are no mutual information (apparent), no strange conditional probabilities, only “stuff” we have an intuitive understanding for. Basically it says that if you know the bandwidth of your system, and the signal to noise ratio, you know the capacity. Furthermore, it says that you can turn the wick up louder and louder on a fixed bandwidth and increase the capacity.

Shannon-Hartley Channel Coding Theorem:

$$C = B\log_2\left(1+S/N\right). \tag{1.12}$$

C is the capacity, B is the bandwidth in Hz and S/N is the unitless power ratio (not in dB but the actual powers in watts ratio). Sometimes this ratio is called the carrier to noise ratio in digital communications systems. I know you like this form better. But it was derived from the Shannon Noisy Channel Coding Theorem to explain, in information theoretic terms, Hartley’s 1928 result. It was the “Eddington eclipse proof of the general theory of relativity” for Shannon since Hartley’s result was “known to be right”. For completeness, we must say that our channels are not white; they are at best colored noise channels. But, the Shannon-Hartley result holds in a nice way.

$$C = \int_{f_1}^{f_2}\log_2\left(1+S(f)/N(f)\right)df. \tag{1.13}$$

This just says that the signal and the noise are functions of frequency and the capacity C is determined by this formula over the interval $[f_1, f_2]$.

In the PSK31 case, the channel is 60 Hz wide (at its -30 dB points) and we want the capacity to be greater than 31.25 bits per second. So we need

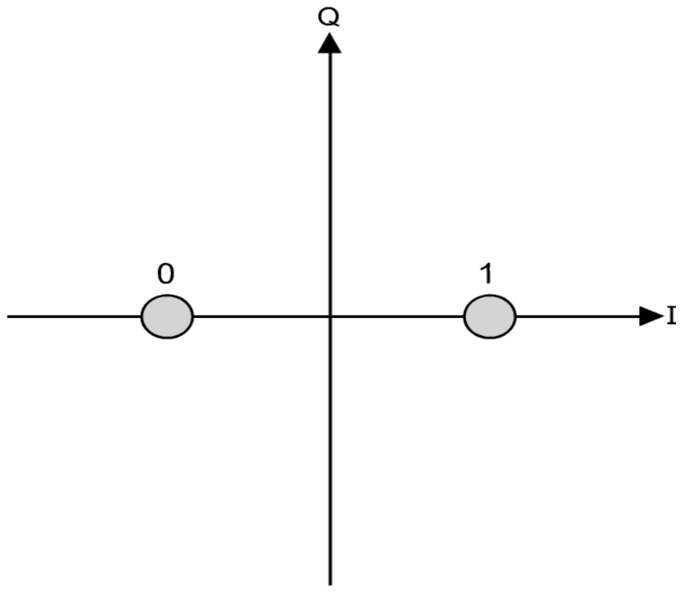
$$\frac{R}{B} < \log_2\left(1+S/N\right). \tag{1.14}$$

Since R is 31.25 bps and B is 60 Hz, this means S/N > .435 or E_b/N_0 > 0.8325 or in dB, we must be greater than -0.78 dB. I think most would agree we are not close. On the other hand, Peter’s design criteria were very strict about having the keyboard to keyboard character of the operation not be radically different from RTTY. He did not want forever feeling latency in the decoding. The penalty is increase SNR is required. But, the question for us is, are we doing a good job? If not, can we preserve the character and do better?

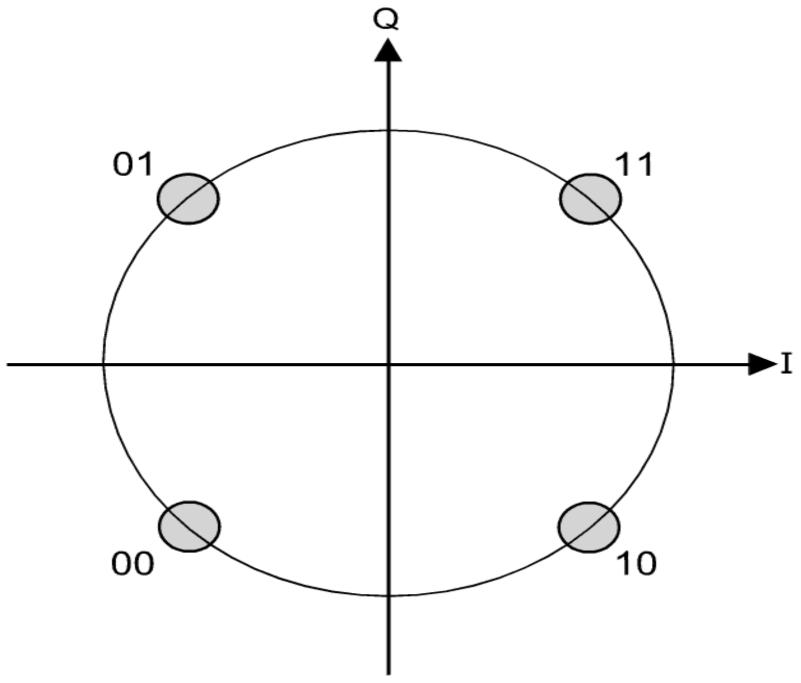
The QPSK mode is not going to cut it. Though it is done using soft decision decoding, it is still not enough to overcome the -4.5 dB penalty. But what if we encode in a different way and make use of a lot that has been learned in the last twenty years.

The problem with encoding the data, and then modulating, is that irrespective of the soft decision, it is still a code based on the probability of the Hamming metric of the digital paths given the channel. We can do better. Many will tell you that a code such as the one Peter used, where neither of the bits transmitted in a QPSK symbol is the actual data is better than the one where the data is sent in the clear with a coding bit accompanying. Ungerboeck showed us how to make this false (3,4).

The BPSK constellation is show in the next figure.



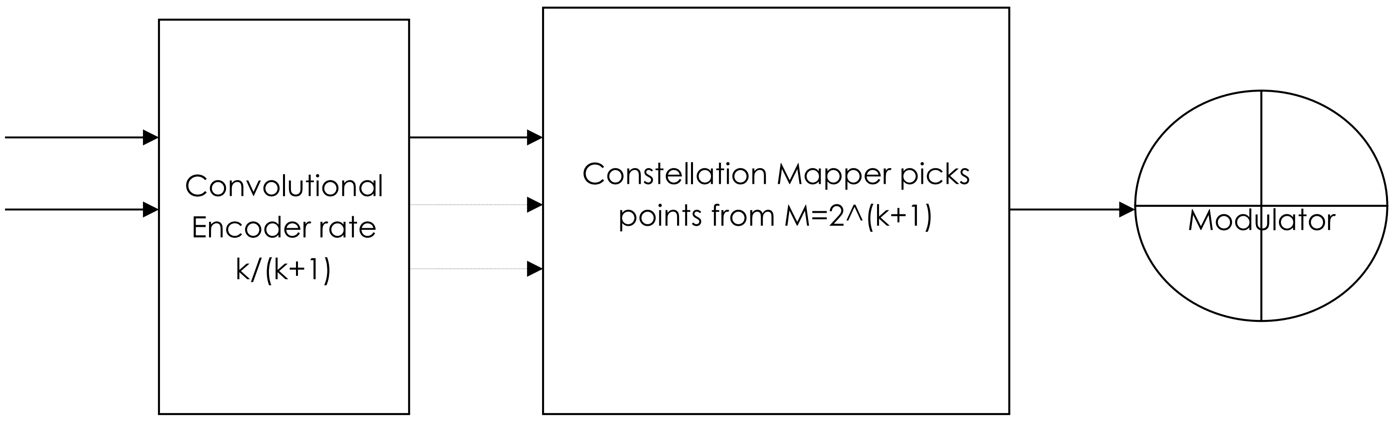
The points 0 and 1 are on opposite sides of the Q axis and are 2 units apart. Now let us consider the QPSK constellation.



We could also choose something similar to this (mathematically equivalent). The important feature to notice is that adjacent points encode only patterns that different in at most one bit. So if, on a noisy channel, you confuse one for a neighbor, you are at most one bit in error. Mathematically speaking, Peter's implementation is equivalent. As in the BPSK mode, 11 and 00 are on opposite sides and 00 means phase reversal and 11 means no phase shift. Just rotate this pretty picture 45 degrees to the right and all will be right with the world.

Trellis Coded Modulation:

Let us consider a different way of doing business. Ungerboeck said let us consider a way to live within our band limited channel, not increase the baud rate, but get actual coding gain. The famous set of ideas starts with the block diagram below. We take k data bits we wish to transmit and one of them goes uncoded to the constellation mapper. The convolutional encoder takes k-1 bits and produces k bits. Thus, k+1 bits feed the constellation mapper. Care is used in selecting how to amp the bits to constellation points. In one of our test cases, where we are taking binary data that was sent to the PSK31 binary modulator, that same data is now sent to the constellation mapper. But in addition to the


Convolutional Encoder rate k/(k+1)
Constellation Mapper picks points from M=2^(k+1)
Modulator

incoming data bit, we add a bit generated by the convolutional coder. This convolutional code can be quite weak, and in our case to keep it familiar to Ungerboeck fans, it can be a systematic code. In our case, we are going to have a weak uncoded bit and a coded version of it. The coded version will choose between vertical and horizontal members of the constellation and the uncoded, weaker data, will only be allowed to choose between members of the vertical or horizontal legs. Now the intuition becomes clearer. The weakest data is forced to choose between points that are farthest apart and the encoded data is used to choose between error prone close data.

In this practical trellis coded modulator, we will get ***a real 3 dB coding gain over the uncoded BPSK data.*** Practical implementations should be straightforward and easily added to several existing pieces of code in place of the conventional coded QPSK. The things to notice is that all zero data will produce the same kinds of revs as before by choice of mapping. It should fit easily into existing systems. Frank Brickle, AB2KT, who has been kind enough to let me pick on him in this paper, and I will be testing on the air with several other volunteers soon. Following this successful experiment, we will use a nonsystematic code and slightly more painful decoding but the increase in coding gain when you do not have these "parallel" paths from systematic codes is a function of the code length only. QRX for lift off.

Appendix A: Varicode

The Varicode Character Set

NUL 1010101011	DLE 1011110111
SOH 1011011011	DCI 1011110101
STX 1011101101	DC2 1110101101
ETX 1101110111	DC3 1110101111
EOT 1011101011	DC4 1101011011
ENQ 1101011111	NAK 1101101011
ACK 1011101111	SYN 1101101101
BEL 1011111101	ETB 1101010111
BS 1011111111	CAN 1101111011
HT 11101111	EM 1101111101
LF 11101	SUB 1110110111
VT 1101101111	ESC 1101010101
FF 1011011101	FS 1101011101
CR 11111	GS 1110111011
SO 1101110101	RS 1011111011
SI 1110101011	US 1101111111
SP 1	C 10101101
! 111111111	D 10110101
" 101011111	E 1110111

# 111110101	F 11011011
$ 111011011	G 11111101
% 1011010101	H 101010101
& 1010111011	I 1111111
101111111	J 111111101
(11111011	K 101111101
) 11110111	L 11010111
* 101101111	M 10111011
+ 111011111	N 11011101
, 1110101	O 10101011
- 110101	P 11010101
. 1010111	Q 111011101
/ 110101111	R 10101111
0 10110111	S 1101111
1 10111101	T 1101101
2 11101101	U 101010111
3 11111111	V 110110101
	W 101011101
4 101110111	X 101011101
5 101011011	Y 101110101

6 101101011	Z 101111011
7 110101101	[1010101101
8 110101011	\ 111110111
9 110110111	] 111101111
: 11110101	^ 111111011
; 110111101	_ 1010111111
< 111101101	. 101101101
= 1010101	/ 1011011111
> 111010111	a 1011
? 1010101111	b 1011111
@ 1010111101	c 101111
A 1111101	d 101101
B 11101011	e 11
f 111101	s 10111
g 1011011	t 101
h 101011	u 110111
i 1101	v 1111011
j 111101011	w 1101011
k 10111111	x 11011111
l 11011	y 1011101
m 111011	z 111010101

n 1111	{ 1010110111
o 111	| 110111011
p 1111111	} 1010110101
q 110111111	~ 1011010111
r 10101	DEL 1110110101

References:

1) Peter Martinez, G3PLX , "PSK31: A New Radio-Teletype Mode", **QEX**, July/August 1999.
2) Steve Ford, WB8IMY, "PSK31 – Has RTTY's Replacement Arrived?", **QST**, May 1999, pp. 41.
3) Gottried Ungerboeck, "Trellis-Coded Modulation with Redundant Signal Sets Part I: Introduction", **IEEE Communications Magazine**, vol. 25, no. 2, February 1987, pp. 5-11.
4) Gottried 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
5) Claude E. Shannon: *A Mathematical Theory of Communication*, Bell System Technical Journal, Vol 27, pp. 379–423, 623–656, 1948.
6) Claude E. Shannon and Warren Weaver: *The Mathematical Theory of Communication.* The University of Illinois Press, Urbana, Illinois, 1949.
7) R.V.L. Hartley, "The transmission of information", Bell System Technical Journal, vol. 3, pp, 535-564, July 1928.
8) Claude E. Shannon, "Communication in the Presence of Noise", **Proceedings of the IEEE** , Vol. 86, pp. 447-457.