Matched Filter and Maximum a Posteriori Detector for Amateur Radio Digital Modes

Rob Frohne, KL7NA
Walla Walla University
1595 Mojonnier Road
Walla Walla, WA 99362
Rob.Frohne@WallaWalla.Edu

Mark Priddy, KE7UFF
Walla Walla University

Mark.Priddy@WallaWalla.Edu

Abstract

This paper describes a method of demodulating a digital signal that takes into account the a priori probabilities of the letters being sent to calculate the a posteriori probabilities of the letters given what was received. The letter with the maximum probability can then be chosen as the detected letter. The probabilities that the correct letter was chosen, given the signal received, can be useful for features like probability squelch and color coding letters with their probabilities so that better estimates of words received can be made by the human operator. This method uses the redundancy built into our language to do error correction.

Keywords: MAP Detector, Matched Filter[i]

Introduction

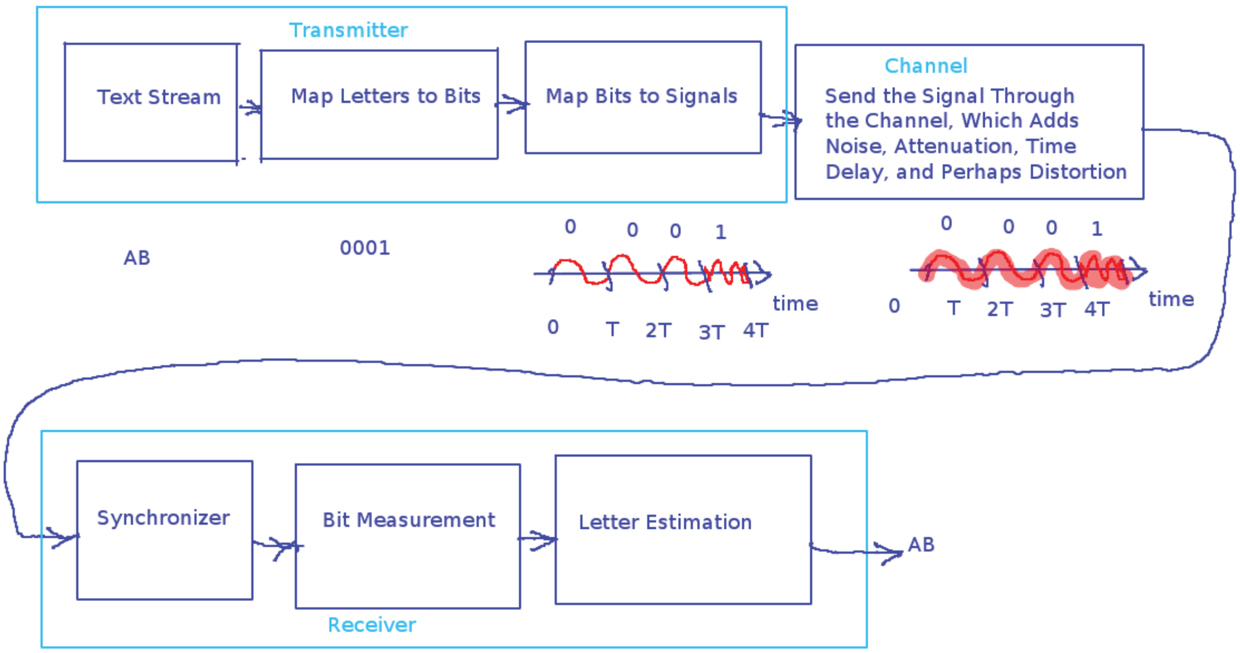
In 1948, Claude Shannon, the father of information theory, proposed the idea that if the probability of sending a zero wasn't equal to the probability of sending a one, then there was redundancy in every bit sent.v Redundancy can be used for error correction. The Maximum a Posteriori (MAP) detector uses the redundancy of the English language to do error correction. The MAP detector can do that for any digital mode that doesn't already take this redundancy into account.

A priori probabilities of sending various letters are the probabilities of sending those letters, before the message has been formed, but after the language has been chosen. For example, in English text, an "E" is more likely than an "X". The MAP detector uses the a priori probabilities, and the signal received, to form the a posteriori probabilities, the probabilities of the sent letters given whatever was received and the a priori probabilities. A Posteriori characterizes that kind of reasoning which goes from cause to effect, so an a posteriori detector is one that uses the signal that was received and the a priori probabilities to determine what was probably sent. A MAP detector is one that picks the signal with the most probability given the signal that was received and a priori probabilities. The thing that makes this interesting is taking into account the a priori probabilities of the symbols. Most detectors are Maximum Likelihood (ML) detectors, and assume that the a priori probabilities are equal, ignoring the redundancy.ix

If a message with a poor signal to noise ratio can be received in context, we are more likely to be able to correctly decode it. When we do this we are using the redundancy of the English language to correct the errors, doing MAP detection by intuition. We are using the a priori information we have that tells us that certain letter or bit combinations are impossible or at least unlikely, and that others are highly likely. For example, "TNX FXR THQ QSO" would probably interpreted as "TNX FOR THE QSO" or "TNX FER THE QSO" because those messages are common on the air, and the other message is very unlikely, in fact so unlikely it doesn't make sense.

A Digital Communications System

The block diagram of a digital communications system is shown in Illustration 1. Only the basic blocks are shown to make the details of the modems stand out. An example message is given to show how it works in a more concrete way.



Digital Communications System

Example modulation shown is frequency shift keying.

Illustration 1: Digital Communications System. The example shown is frequency shift keying. The mapping of letters to bits is given in Table 1: Mapping Symbols (Letters) to Bits.

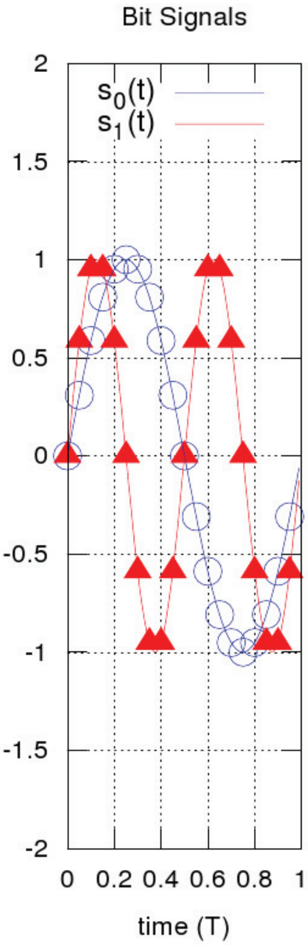
In the transmitter, the modem takes letters (from the keyboard usually) and maps them each to a binary string. In our specific example we will have four symbols, A, B, C and D that are mapped to two bits each. That mapping is shown in Table 1.

Most bit mappings have provisions for bit synchronization which allows a demodulator to know when the start of a new letter is, or perhaps the end of a letter, or both. For example, in RTTY, there are five bits that send the letter and three bits that do synchronization, one start bit and one and a half to two stop bits.[ii] [iii] In our example, if we started listening in a random place in the bit stream, we would have a fifty-fifty chance of being out of sync with the bit stream. For this example, let's ignore this problem. We will assume we start decoding with the first bit of the first letter and since we know the symbol length is two bits, we keep in sync.

Letter	Leading Bit	Trailing Bit
A	0	0
B	0	1
C	1	0
D	1	1

Table 1: Mapping Symbols (Letters) to Bits

Illustration 2: Example signals for each bit.



The next step in the transmitter is to map each bit in the bit stream to a signal, $s_0(t)$ for zero and $s_1(t)$ for one. These signals are sent out in a serial fashion, one right after the other in the same way the bits are arranged. The signals, $s_i(t)$, introduce redundancy and other properties (like having a frequency that will reflect from the ionosphere) necessary to overcome problems occurring in the channel to the receiver. The signals $s_0(t)$ and $s_1(t)$ need to be as different as possible, so that they won't easily be confused, by what happens to them in the channel, such as time delay, additive noise, multi-path propagation, fading, etc. Picking the signals $s_0(t)$ and $s_1(t)$ is important if designing a new protocol. In our case we want to make detectors for existing modes so they have already been picked, by the protocol designers.

Often, as in our example, but not always, the energy per bit in the signal $s_0(t)$ is the same as for $s_1(t)$. They are almost always non-zero for the bit duration time, T seconds, for both signals, because they must be interchangeable in order to send either a zero or a one in the specified time slot.[iv]

These signals are really functions of time that the transmitter sends out. They can be seen on an oscilloscope. For our example system, I picked the first two harmonics of the bit frequency.

$$\begin{aligned} s_0(t) &= \sin\left(\frac{2\pi t}{T}\right) \\ s_1(t) &= \sin\left(\frac{4\pi t}{T}\right) \end{aligned} \tag{1}$$

They are shown as they would be on an oscilloscope in Illustration 2: Example signals for each bit. This, however, is not the only way to think of these signals. Claude Shannon advocated another way of looking at these signals.[v] If these were run through an analog to digital converter, like the input of a computer sound card, it would yield a list of measured (or sampled) voltages, as shown in Illustration 2 with the triangles and circles. This list could be represented as a vector, or equivalently a point in space. If f_s is the sound card sampling frequency, then each signal would have $N = f_s T$ components, or values. In other words it would be a vector or a point in N dimensional space. Try to imagine N dimensional space by thinking of moving from one dimension to two and then to three and onward. To unify the vector and the point way of thinking, the vector from the origin to the point with the coordinates given by the sample values is the vector we represent the signal with.

When a transmission is made, it will have a number of these signals, $s_i(t)$ strung together. Then when it goes through the channel from the transmitter to the receiver, there is a time delay and noise is added to the signal. See Illustration 3.

At the receiver, the continuous signal will be sampled, producing the numbers corresponding to the values of the triangles inIllustration 3. The job of the receiver is to take those samples and figure out what letters were sent.

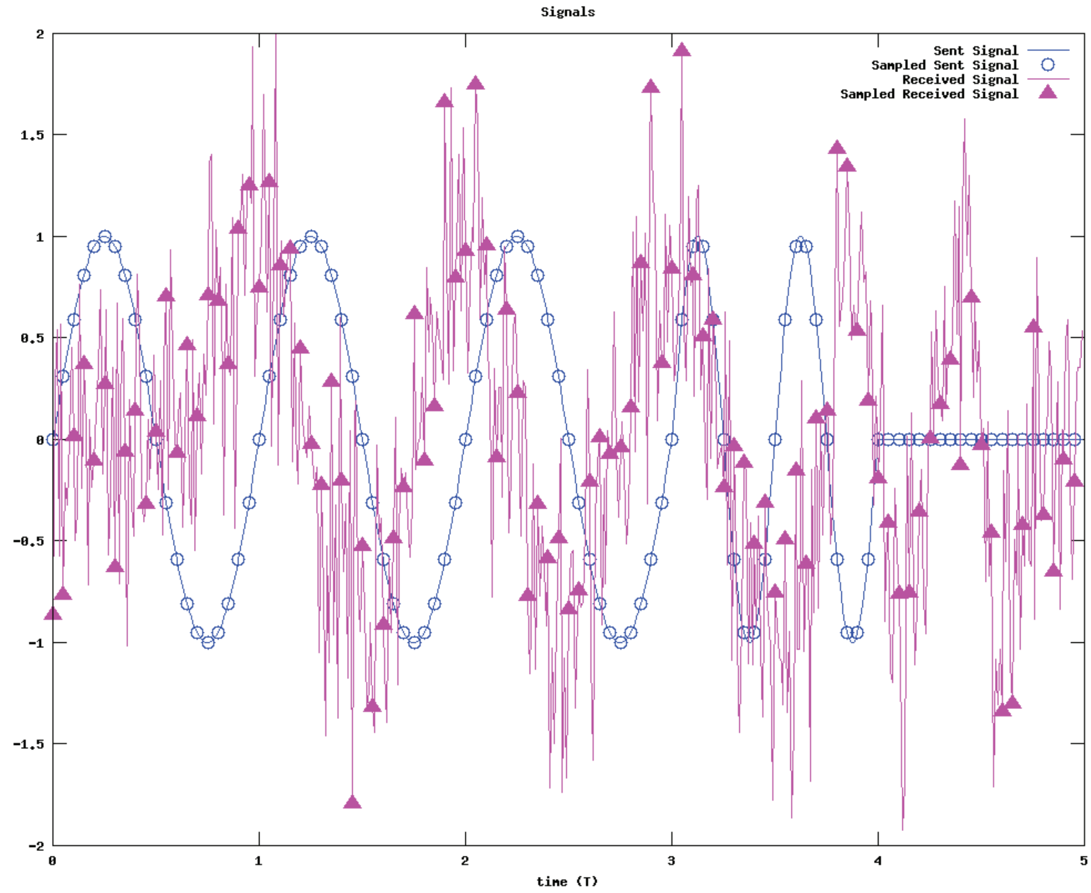


Illustration 3: Signal sent and signal received for the 0001 bit pattern.

Synchronization

There is first a problem of signal synchronization, and this problem occurs on every transmission. The delay in the channel is unknown to the receiver. So it must be measured. To do that, imagine taking the noisy received waveform of Illustration 3, putting it on a transparency and sliding it along $s_0(t)$ from Illustration 2, sliding it from the right toward the left until it appeared to match (trying to ignore the noise). This is roughly what is done to synchronize the signal at the receiver. To make this idea of matching more concrete, consider the signals to be vectors as Shannon recommended. Each different time shift (shifting by $1/f_s$ each time) is tried to find at what time shift, T_d, the vectors match best. Matching in this case, means how much are they in the same direction[vi]. To tell if two vectors are in the same direction, a dot or inner product is done, which is the sum of the products of like components of the two vectors. See the simple two dimensional example in Illustration 4. The inner product of a vector with a unit length vector (unit vector) is the component of the first vector in the direction of the unit vector. Note: the inner product with $s_1(t)$ is also needed, because there is no guarantee that the first bit will be a zero, as it was in this case.

Matched Filter

Once synchronization is achieved, the nth bit is received by finding the inner product of the received signal (shifted over n bit periods plus the delay, $nT + T_d$) with $s_0(t)$ and with $s_1(t)$ and determining which one is largest, (matches best).[vii] These inner products are mathematically equivalent to what is known as a "matched filter." A matched filter is an optimum linear filter, designed to pick out a specific signal in the presence of additive white Gaussian noise (AWGN).[viii] White noise is broadband. It is the same at every frequency. Gaussian noise follows the Normal or bell shaped curve, and Shannon has shown that for a given power level, it is the noisiest type of noise there is.v

Test Statistic

Let z_m be the inner product of the received signal with the synchronized $s_0(t)$, and likewise let z_p be the inner product with $s_1(t)$. We are really interested in the difference between these two, $z = z_p - z_m$. If z_p is larger z should be positive; if z_m is larger then the number should be negative.

Targets

In the case where there is no noise coming from the channel and a one is sent, then **z** will equal a target value, a_1. In the case where there is no noise coming from the channel and a zero is sent, then z will equal a different target value, a_0. If the energy in $s_0(t)$ is equal to the energy in $s_1(t)$, then $a_0 = - a_1$. This is because the inner product of a signal with itself is proportional to the energy in the signal.

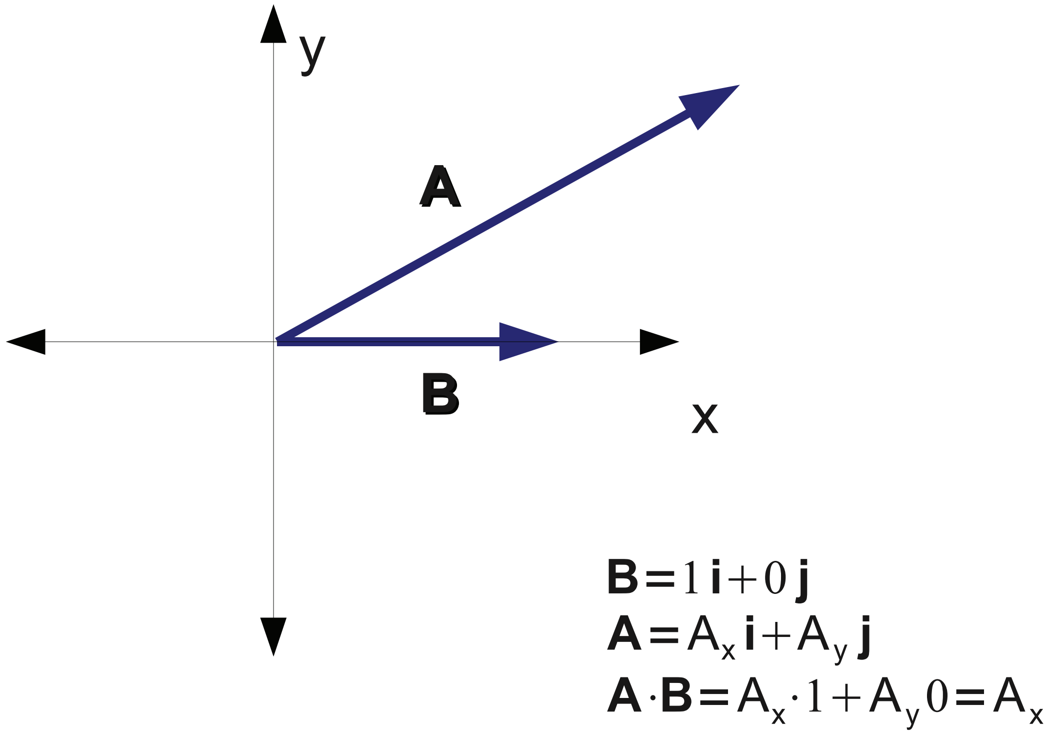

y
A
B
x
$\mathbf{B}=1\,\mathbf{i}+0\,\mathbf{j}$
$\mathbf{A}=A_x\,\mathbf{i}+A_y\,\mathbf{j}$
$\mathbf{A}\cdot\mathbf{B}=A_x\cdot 1+A_y 0=A_x$

Illustration 4: Vector inner product: If B is a unit vector, the inner product of A and B is the component of A along the direction of B.

If **B** is a unit length vector, the inner or dot product of **A** and **B** is the component of **A** along the direction of **B.**

$$\boldsymbol{A}\cdot\boldsymbol{B} = A_x B_x + A_y B_y \qquad (2)$$

Computing Probabilities

In order to do MAP detection, the probability that a symbol was sent must be calculated, given what was received. Knowing this probability is also useful because it gives an indication of the certainty of the bits received, indicating whether they are reliable or not. Printing could be squelched based on probabilities instead of, or in addition to audio levels. So to start out with we will describe how to calculate the probabilities for a single bit. Then we will extend those ideas to multiple bits so we can incorporate the a priori probabilities of the letters in an amateur radio QSO.

One Bit MAP Detector

The assumption that $s_0(t)$ was sent if it matches better than $s_1(t)$ implicitly assumes that sending a zero and sending a one are equally likely. This is the assumption that the ML[ix] detector makes. Many modem programs use it. Let's forgo this assumption, and see what happens. Let S_0 be the event that a zero bit was sent, and S_1 be the event that a one was actually sent. $P(S_0)$ is the probability that a zero was sent, and $P(S_1)$ is the probability that a one was sent. Let z_0 be a specific value we get for the random variable z by taking the difference of the inner products.

First we need to understand Bayes' Rule. Since the event $[(z=z_0)$ and $S_1]$ is the same as $[S_1$ and $(z=z_0)]$,

$$P((z=z_0)\cap S_1) \;=\; P(S_1\cap(z=z_0)) \tag{3}$$

where $\cap$ denotes "and," the intersection of the two events.

$$P((z=z_0)\cap S_1)=P((z=z_0)|S_1)P(S_1) \tag{4}$$

$$P(S_1\cap(z=z_0))=P(S_1|(z=z_0))P(z=z_0) \tag{5}$$

where the $P((z=z_0)|S_1)$ is the conditional probability of $(z=z_0)$ given S_1, etc. Equating both of the last two equations because of the one above them yields

$$\begin{gathered} P((z=z_0)|S_1)\,P(S_1)=P(S_1|(z=z_0))\,P(z=z_0) \\ P(S_1|(z=z_0))=\frac{P((z=z_0)|S_1)\,P(S_1)}{P(z=z_0)} \\ . \end{gathered} \tag{6}$$

Likewise, by using S0 instead of S1, we get

$$P(S_0|(z=z_0))=\frac{P((z=z_0)|S_0)\,P(S_0)}{P(z=z_0)} \tag{7}$$

This is known as Bayes' Rule. This is just what we need, because after a bit signal is received, we know $z=z_0$, and we want to compare the probability $P(S_1|(z=z_0))$ with $P(S_0|(z=z_0))$. The symbol with the largest probability of being sent, given what we received is the one we pick as our best guess for what was sent.

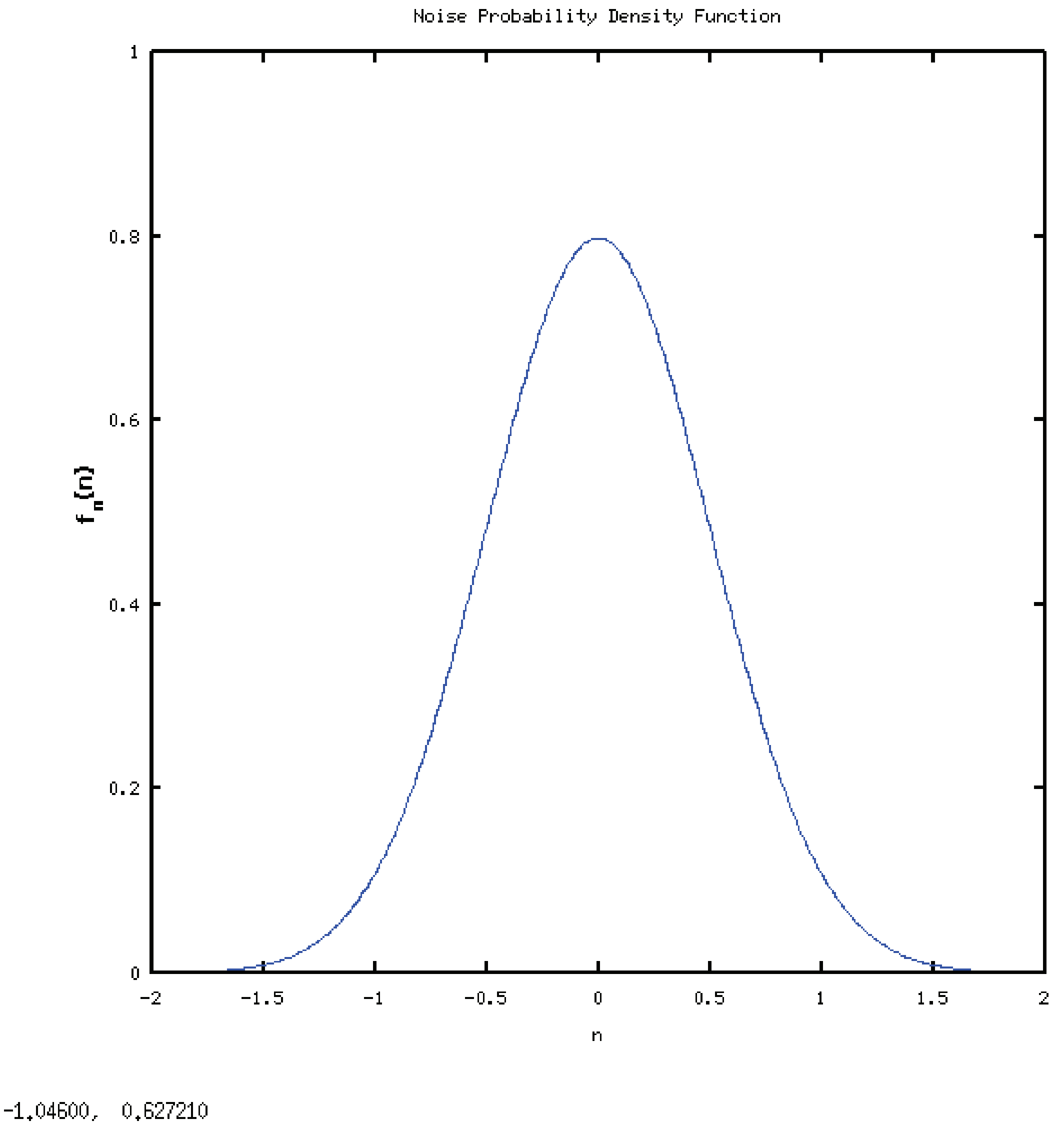


Illustration 5: The probability density function of the noise. A Normal curve with zero mean and in our example, standard deviation of 0.5.

The terms on the right hand side of (6) and (7) are functions of the noise. For the illustrations, we assume the noise is AWGN, and the receiver is linear. In this case, the noise, n, will be a random variable that is normally distributed with zero mean and variance proportional to the power in the noise. This is a not a bad assumption in most cases, but if it is, the actual probability density function of the noise can be estimated from samples. Illustration 5 shows the normally distributed noise probability density function with variance 0.25.

The random variable z is composed of the sum of two other random variables, the noise, n, and the target value, a, which takes on values a_0 with probability, $P(S_0)$, and a_1 with probability, $P(S_1)$.

$$z = a+n \tag{8}$$

This implies the conditional probability density function for the the test statistic, z given S_0

$$f_{z|S_0}(z|S_0)=f_n(z-a_0) \tag{9}$$

and given S_1

$$f_{z|S_1}(z|S_1)=f_n(z-a_1) \tag{10}$$

is just the probability density function of the noise shifted so it is centered on the target values. For a small region Δz wide, around z_0,

$$\begin{aligned} P((z=z_0)|S_0)&=f_{z|S_0}(z|S_0)\Delta z \\ P((z=z_0)|S_1)&=f_{z|S_1}(z|S_1)\Delta z \end{aligned} \tag{11}$$

.

The $P(z=z_0)$ can be found like this.

$$\begin{aligned} P(z=z_0)&=P[((z=z_0)\cap S_0)\cup((z=z_0)\cap S_1)] \\ &=P[((z=z_0)\cap S_0)]+P[((z=z_0)\cap S_1)] \\ &=P[((z=z_0)|S_0)]P(S_0)+P[((z=z_0)|S_1)]P(S_1) \end{aligned} \tag{12}$$

Finally, (6) and (7) along with(10), (11) and (12) yield the final formula for calculating the conditional probabilities.

$$P(S_0|(z=z_0)) = \frac{f_n(z_0-a_0)P(S_0)}{f_n(z_0-a_0)P(S_0)+f_n(z_0-a_1)P(S_1)} \tag{13}$$

and

$$P(S_1|(z=z_0)) = \frac{f_n(z_0-a_1)P(S_1)}{f_n(z_0-a_0)P(S_0)+f_n(z_0-a_1)P(S_1)} \tag{14}$$

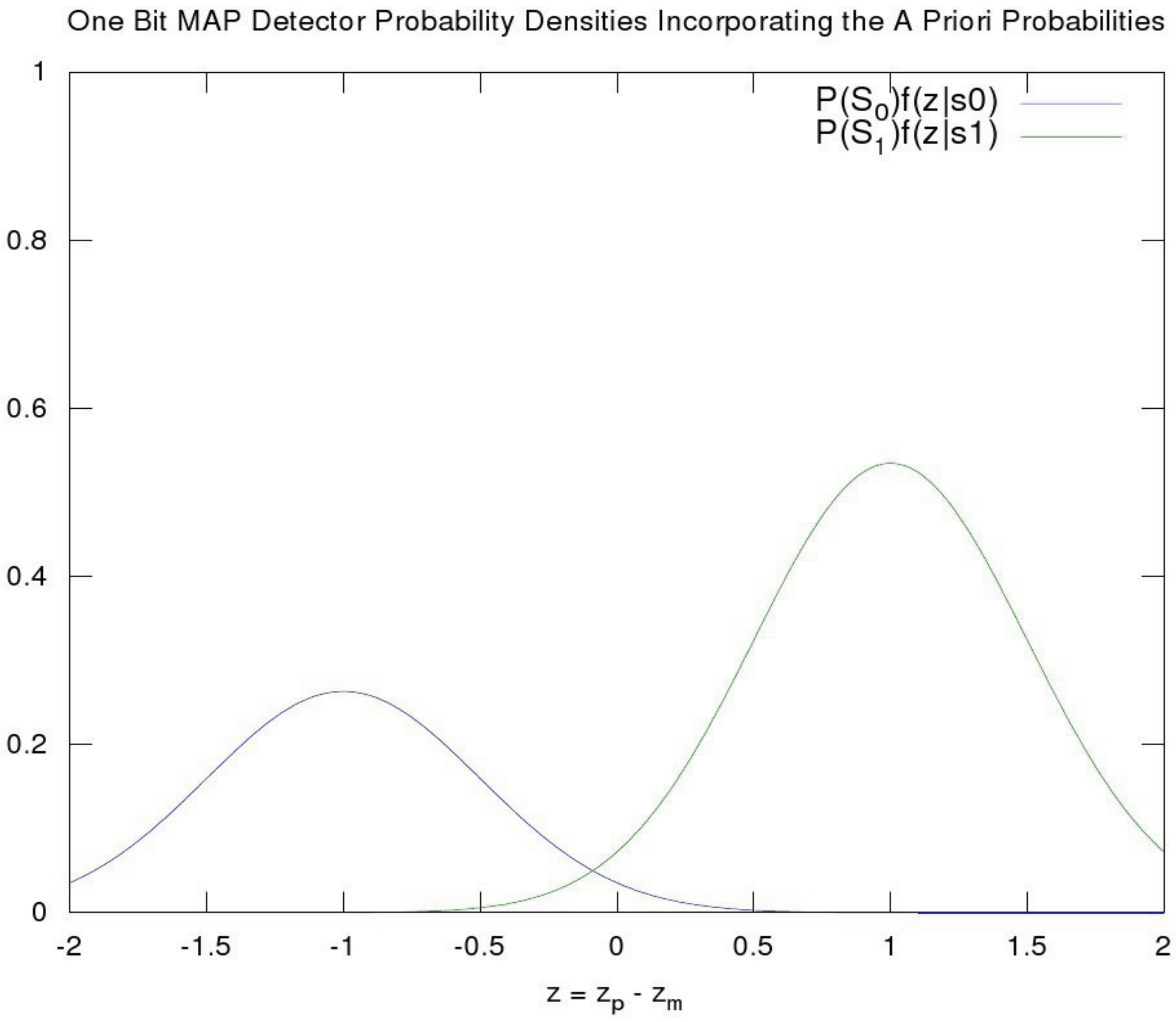


Illustration 6: For a particular z value, the symbol with the highest probability is chosen.

The $P(S_1|(z=z_0))$ and the $P(S_0|(z=z_0))$ both have the same denominator (scaling factor), so when comparing them, the denominator can be ignored, while simply looking for the one (the i value) with the largest $f_n(z_0-a_i)P(S_i)$. The matched filter yields $z_{0.;}$ Illustration 6 is used to pick the curve that is highest at that z_0 value. The best guess for the bit is the bit corresponding to that curve.

In Illustration 6, $a_0 = -1$, $a_1=1$, $P(S_0) = 0.33$, $P(S_1) = 0.67$, and the variance is 0.25. In this case the signal to noise ratio is 6 dB. Note that the cross over point is slightly less than z=0, which would have been the cross over point if one and zero had equal probabilities. This is the benefit of the MAP detector over the ML detector which assumes each bit has equal a priori probability. Note that the big gains for the MAP detector come when the bits are far from equally likely. The probability of making the mistake of saying the sent bit was a zero, when it was really a one, coming from using the ML detector instead of the MAP detector is the area under the green curve between where the curves cross and z = 0, and in this case it looks to be between one and two percent. That means about two bits in a hundred will be incorrectly interpreted with a ML detector that would have been right with a MAP detector. This may not seem very bad, however, in order to get a letter right, all the bits have to be correct, and if there are eight bits per letter, that means roughly one in ten letters will be incorrect with ML that would be fine with MAP detection.

This is all great, but our explanation has limited us to taking one bit at a time. Most digital modes have assigned roughly equal probabilities to the zero and the one. In that case, if z is greater than zero, a one is chosen, and if it is less than zero, a zero is chosen, and a ML and a MAP detector give the same results. In order to really gain an advantage with the MAP detector, the probabilities must differ a lot, and in order for that to happen, bits are grouped together, so that some combinations are more unlikely. It is natural to use the letters themselves for this grouping, because they have a number of bits and the a priori probabilities are easier to find. The next section shows what happens with two bit letters and how to generalize the ideas of the one bit MAP detector to a larger number of bits.

Multiple Bit MAP Detector

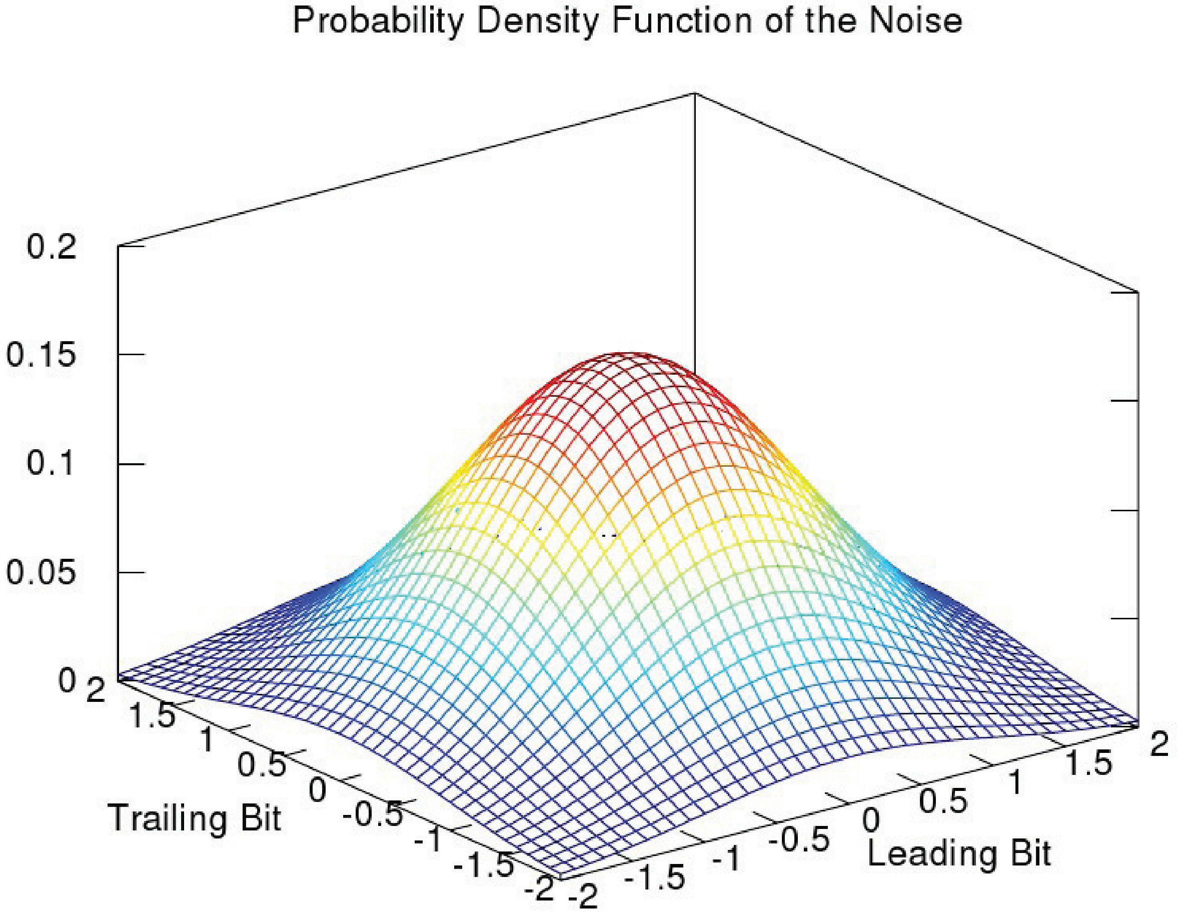


Illustration 7: The noise probability density function for two bits is dimensional with zero mean. In this example it is normal with standard deviation 0.5.

In order to extend the ideas of the one bit MAP detector to more bits, again the vector concept is essential. For each bit, the z value is stored into a vector called **z** until it contains the number of bits grouped together. For this example it will be two, as it is much easier to visualize than when more dimensions are added. So **z** is a point on a two dimensional plane. One dimension relates to the leading bit and the other to the trailing bit. There will be four target values analogous to a_0 and a_1, and they can be made into vectors as well, $\mathbf{a_0}$, $\mathbf{a_1}$,$\mathbf{a_2}$ and $\mathbf{a_3}$. The noise probability density function is now two dimensional because errors can occur in either bit direction, as seen in Illustration 7. Surrounding these target points, $\mathbf{a_i}$ are two dimensional conditional probability density functions which are just the noise probability density function centered on the appropriate target points and scaled with the probability of the individual bit patterns as seen in Illustration 8. This figure is analogous to Illustration 6. The mathematical steps to arrive at this follow those above, with the appropriate vector substitutions. The results are:

$$P(S_i|(z=\boldsymbol{z_0})) \;=\; \frac{f_n(\boldsymbol{z_0}-\boldsymbol{a_1})P(S_i)}{\sum\limits_{symbols\ j} f_n(\boldsymbol{z_0}-\boldsymbol{a_j})P(S_j)} \qquad (15)$$

Just as before, P(S$_i$|(**z=z$_0$**)) is evaluated for each i and the largest is picked as our best guess for what symbol was actually sent, given what was received.

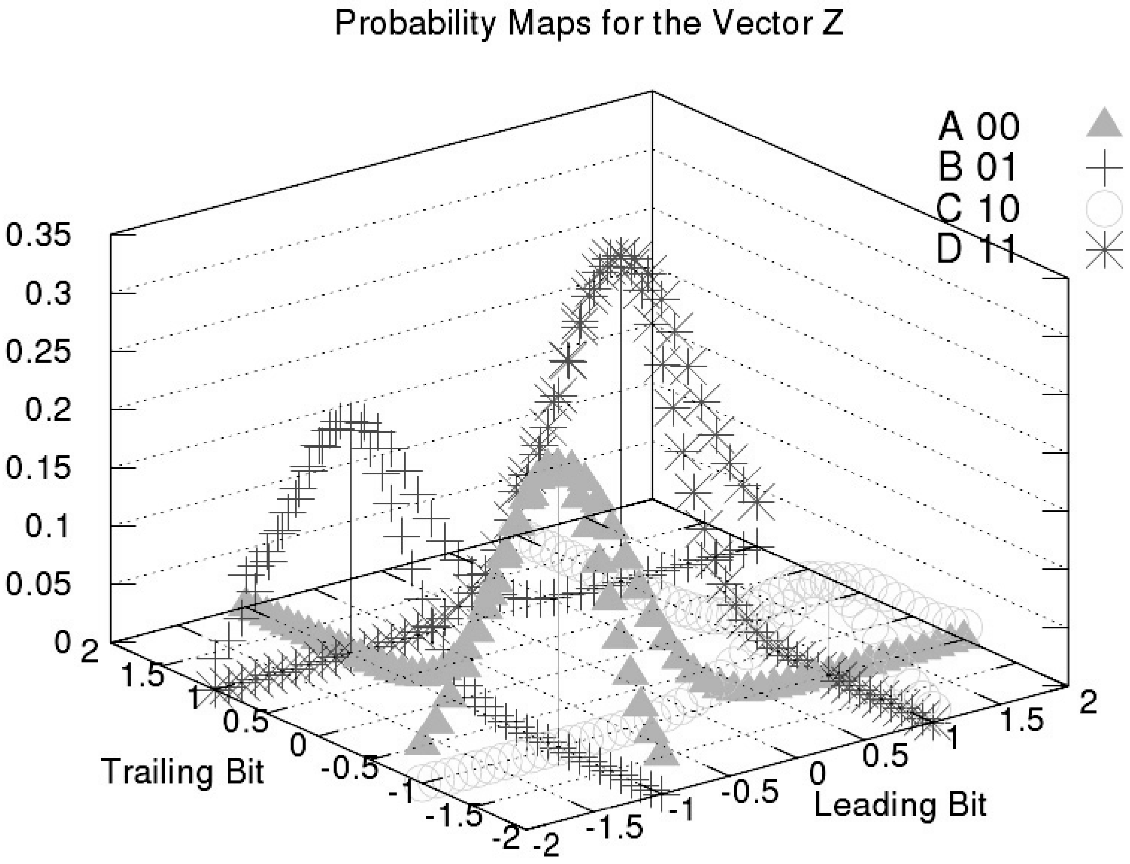


Illustration 8: Each target point is hit according to the probability distribution about it. The example shown here has P(00) of 0.3, P(01) of 0.25, P(10) of 0.1 and P(11) of 0.35.

Other Considerations

In the real world there are always complications to be addressed, and there are those here as well. Some digital schemes have variable length symbols. Examples include Morse Code and PSK31. Those have to be converted to fixed length, by appending letters together, and then new a priori probabilities, $P(S_i)$, need to be computed based on the probabilities for the individual letters. In general these modulation types have tried to take into account the relative probabilities of the letters by making the symbols longer for infrequent letters, and shorter for more frequent ones, so there is less to be gained by applying a MAP detector to them.

Sometimes the actual signals $s_0(t)$ and $s_1(t)$ for the bits are not completely known. For example, what was the sign, phase and frequency of the signals sent? How are the target values, $\mathbf{a}_i$ found? What is the actual noise probability density function? How are the a priori probabilities of the letters, $P(S_i)$, for real ham QSOs obtained? Usually they are all measured from the signals and the noise received, by averaging or some other statistical method.

What are the effects of automatic gain control (AGC) in the receiver? What about automatic frequency control (AFC)? They could change the target values, $\mathbf{a}_i$, and effect the noise characteristics. It is evident that software defined receivers have the advantage, because the information about AGC and AFC and other settings can be used to better demodulate the incoming signals.

Conclusion

Using the mathematical notion of vectors, matched filters were found to be easy to implement. They are the optimum linear filters for digital detection in the presence of additive white Gaussian noise. The MAP detector uses the output of the matched filters and the redundancy in the symbols to choose the symbol received as the one with the highest probability given the a priori probabilities of the symbols and the signal received. It is based on Bayes' Rule, and applying it is fairly simple. The probabilities of being correct that are already calculated when demodulating with a MAP detector are also useful to the human when interpreting the message and as a squelch indicator.

It is hoped that this paper will serve to expose the ideas of MAP detection and matched filters to more experimenters, so that amateur software demodulators integrated with software defined radios will be improved beyond the state of the art today. The authors are playing with these ideas in an RTTY demodulator, using the open source program Fldigi as a platform.

[i] C. E. Shannon, "A Mathematical Theory of Communication," Bell System Technical Journal, Vol. XXVII, July, 1948.

[ii] Donald Hill, AA5AU, http://www.aa5au.com/gettingstarted/rtty_diddles_technical.htm.

[iii] Robert Glassey, G0VTQ, http://www.rttyinfo.com/rttyinfo/sta/Baudot~1.htm.

[iv] Sometimes, as in RTTY, the stop bits are not specified as being an integer number of bit times which adds complications to detection.

[v] Claude E. Shannon, "Communication in the Presence of Noise," Proceedings of the I.R.E. January, 1948.

[vi] The reason we want the same direction (and not same length) for the vectors is that the channel changes the amplitude of the signal (and thus the length of the vector), but its general direction remains the same, assuming the noise isn't too loud.

[vii] This assumes the probability of a zero is the same as the probability of a one.

[viii] Hwei Hsu, "Analog and Digital Communications, Second Edition," McGraw-Hill, 2003.

[ix] http://en.wikipedia.org/wiki/Maximum_a_posteriori_estimation