**EECS 121: Coding for Digital Communication & Beyond** **Fall 2013**

# Lecture 5 — September 17

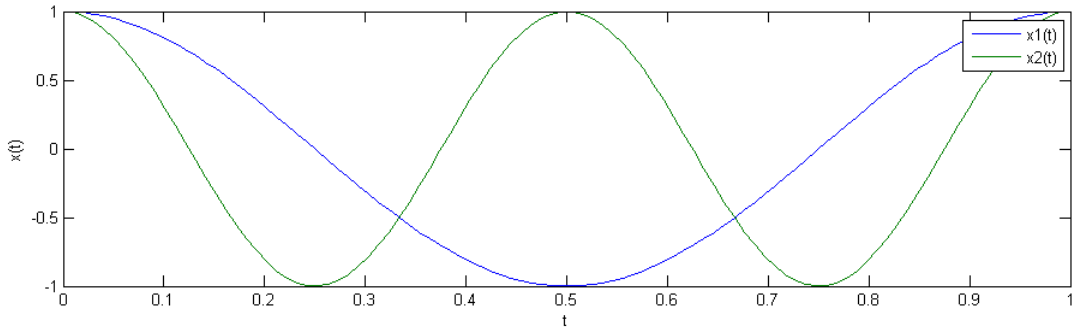
*Lecturer: Prof. Anant Sahai* *Scribe: Preetum Nakkiran*

## 5.1 Orthogonal Signaling

Consider the familiar model of a sender and receiver communicating over a channel with AWGN. Say the sender communicates by transmitting a signal x(t) over a period of time. Depending on the message, the sender transmits different signals x(t). In OOK, for example, the sender may transmit x(t) = cos(t) for a one and x(t) = 0 for a zero. In order to transmit more than one bit over the same duration T, the sender must choose between more than 2 signals. In particular, if the sender transmits one of $2^k$ different signals, then the receiver can theoretically recover $k$ bits. In **orthogonal signaling**, the choice of x(t) is made from a family of orthogonal functions (which have nice properties for decoding and noise analysis).

Let us consider a concrete example that will be explored in detail in the following sections. The sender transmits a waveform $x(t)$ over a period of time $T$. In order to send $k$ bits, the sender picks $x(t)$ as one of $2^k$ functions from the orthogonal cosine family:
$\{x_i(t) = cos(2\pi f_c it), i \in \mathbb{N}\}$ where $f_c = \frac{1}{T}$ (the fundamental period)
For example:
$x_1(t) = cos(2\pi f_c t)$
$x_2(t) = cos(2\pi f_c 2t)$
$\vdots$
$x_{2^k}(t) = cos(2\pi f_c (2^k)t)$

Examples of $x_1(t)$ and $x_2(t)$ for $T = 1$:

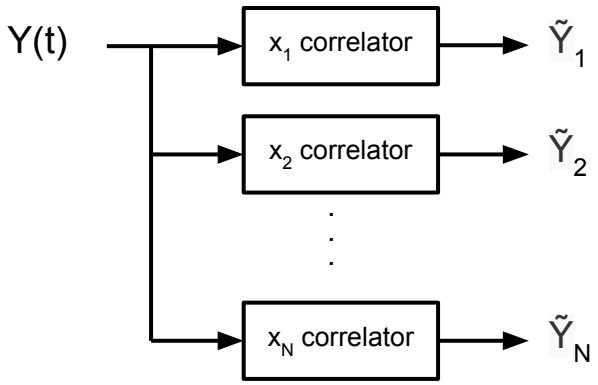



Notice that each signal completes an integer number of periods over time $T$. It is useful to think of these signals as vectors in the vector space of functions over $[0, T]$, with the

traditional inner-product: $< f, g >= \int_0^T f(t)g(t)dt$. Notice that the cosine family is indeed orthogonal under the inner-product, and the energy in each signal is a constant:

$$||x_i||^2 = \int_0^T \cos^2(\frac{2\pi}{T}it)dt = \frac{T}{2}$$

Now consider the receiver. The received signal $y(t)$ is corrupted by AWGN: $y(t) = x_i(t)+n(t)$. In order to recover the signal, the receiver first correlates $y(t)$ with all potentially sent signals $x_i(t)$




The output of the correlations are a set of scalars $\widetilde{Y}_i$, representing the component of the received signal in the direction of $x_i(t)$:

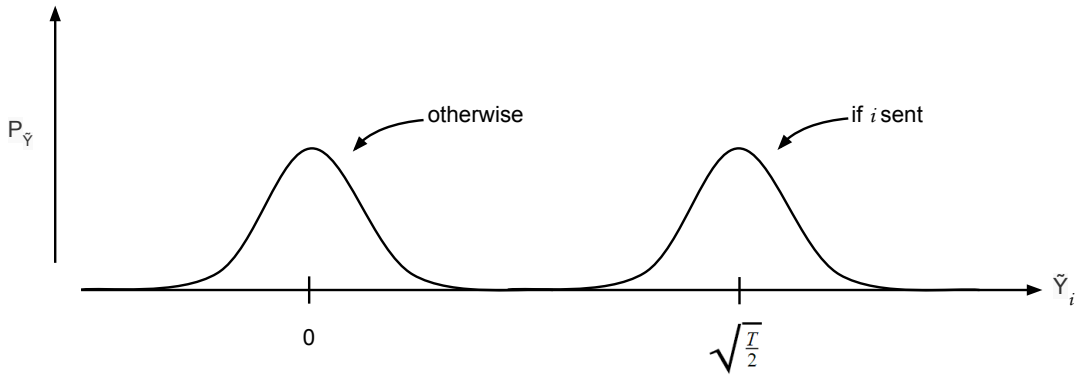
$$\widetilde{Y}_i = < y(t), x_i^{norm}(t) > = < y(t), \frac{1}{||x_i||}x_i(t) > = < y(t), \sqrt{\frac{2}{T}}cos(2\pi f_c it) >$$

The process of correlation is essentially a projection onto each orthogonal signal in the input family. Notice that the correlation is with respect to a normalized signal, so it is exactly a projection. Let us look at the distribution of the $\widetilde{Y}_i$s, if a signal $x_j(t)$ was sent:

$$\begin{aligned}\widetilde{Y}_i &= < y(t), x_i^{norm}(t) > \\ &= < x_j(t) + n(t), x_i^{norm}(t) > \\ &= < x_j(t), x_i^{norm}(t) > + < n(t), x_i^{norm}(t) > \\ &= < x_j(t), x_i^{norm}(t) > + N(0, \sigma^2||x_i^{norm}||^2)\end{aligned}$$

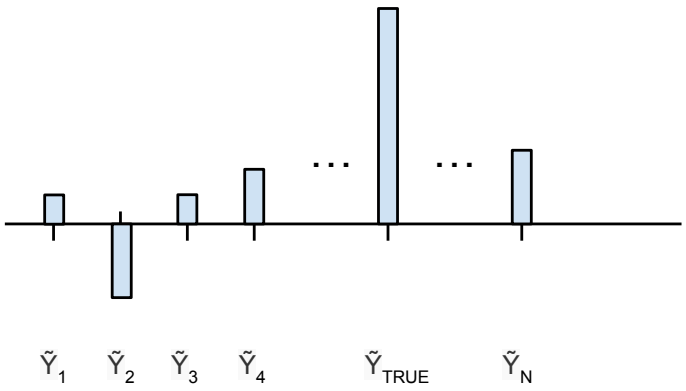
$$\widetilde{Y}_i = \begin{cases} \sqrt{\frac{T}{2}} + N_i \text{ if i was sent} \\ N_i \text{ otherwise} \end{cases}$$

Where $N_i \sim N(0, \sigma^2)$. Notice that the $N_i$s are iid, since $x_i^{norm} \perp x_j^{norm}$ for $i \neq j$.




We expect the correlation $\widetilde{Y}_i$ to be around $\sqrt{\frac{T}{2}}$ if $x_i$ was sent, and around zero otherwise. Now we must choose a decision rule for the decoder, to recover the original message. Given these properties of $\widetilde{Y}_i$, a natural rule suggests itself: pick the max $\widetilde{Y}_i$, and decode the signal as $i$. (This is in fact equivalent to the ML decision rule).

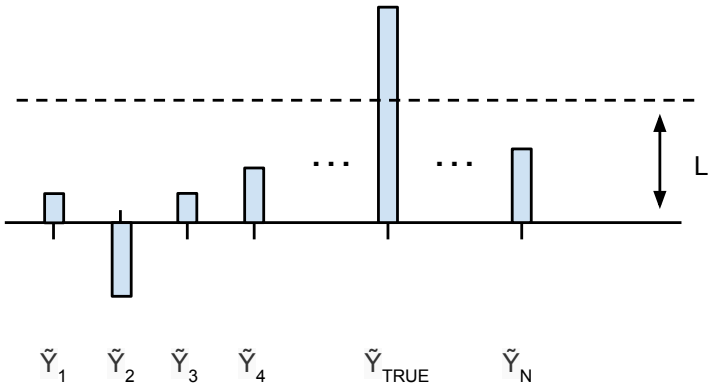
Now let us consider what would happen if the sender wishes to pack in more bits in his message (over a fixed duration $T$). Let us visualize the sequence of $\widetilde{Y}_i$s as bars on a line, with the height of the $i^{th}$ bar corresponding to $\widetilde{Y}_i$. For proper decoding, the max $\widetilde{Y}_i$ must always correspond to the sent signal $x_i$.




However, we can intuitively see that as we try to pack in more signals (bars in the diagram), the probability of noise causing one bar to randomly jump above the received signal increases. More exactly, the probability of decoding error ($\widetilde{Y}_j > \widetilde{Y}_{true}$ for some $j$) increases with increasing number of bits $k$. In order to curb this error probability, it is clear that we must increase the expected value of $\widetilde{Y}_{true} = \sqrt{\frac{T}{2}}$, by increasing the energy in the signals (in our case, by increasing $T$). Intuitively, this should put more distance between the true signal and the noise. But by how much do we need to increase the transmitting energy for each additional bit we wish to send? Let us investigate this formally.

## 5.2 Noise Analysis

To ease analysis, let us use a simpler (but almost equivalent) decoding rule: Set a threshold L. If exactly one $\widetilde{Y}_i > L$, then decode to $i$. Otherwise, "give up" (claim an erasure).




With this rule, there are two possible (independent) error events:
1) Noise pushes the true message below the threshold. ($\widetilde{Y}_{true} < L$)
2) Noise pushes some other $\widetilde{Y}$ above the threshold: ($\exists \widetilde{Y}_{false} > L$)

Given we want to send $k$ bits, we want to choose parameters $L$ (the correlation threshold) and $T$ ($\propto$ signaling energy) such that the probability of both these events tend to 0.

Let us first look at Error Event 2, and try to find an appropriate threshold $L$ that will give a sufficiently small probability of error (that the noise pushes some other $\widetilde{Y}_i$ above the threshold):

$$\begin{aligned} P(\exists \widetilde{Y}_{false} > L) &< 2^k P(Y_0 > L) \qquad \text{by identically-distributed and union bound} \\ &= 2^k Q(\frac{L}{\sigma}) \\ &< 2^k e^{-\frac{L^2}{2\sigma^2}} \\ &= e^{k(\ln 2) - \frac{L^2}{2\sigma^2}} \\ &= e^{-k(\frac{L^2}{2k\sigma^2} - \ln 2)} \end{aligned}$$

We want to keep the term $\frac{L^2}{2k\sigma^2} - \ln 2$ positive, so let us set the threshold $L$ such that $L^2 = (\ln 2 + \epsilon)2k\sigma^2$:

$$\begin{aligned} P(\exists \widetilde{Y}_{false} > L) &< e^{-k(\frac{(\ln 2+\epsilon)2k\sigma^2}{2k\sigma^2} - \ln 2)} \\ &= e^{-k\epsilon} \end{aligned}$$

With such a threshold, the probability of error will go to 0 with increasing $k$.

Now that we have a threshold that suppresses the noise, let us see what energy our signals must be to pass the threshold.

We want to keep the probability of Error Event 1 small. Recall that this is the probability that the true signal is below the threshold:

$$\begin{aligned} P(\widetilde{Y}_{true} < L) &= P(\sqrt{\frac{T}{2}} + N_i < L) \\ &= P(N_i < L - \sqrt{\frac{T}{2}}) \\ &= Q\left(\frac{\sqrt{\frac{T}{2}} - L}{\sigma}\right) \\ &< e^{-\left(\frac{\sqrt{\frac{T}{2}}-L}{\sigma}\right)^2/2} \end{aligned}$$

If we set $T$ to keep the term $\sqrt{\frac{T}{2}} - L$ large, then this error probability will also go to 0. We want: $\sqrt{\frac{T}{2}} > L \implies \frac{T}{2} > L^2$. Using our threshold $L$ from the previous section, we need: $\frac{T}{2} > (\ln 2 + \epsilon)2k\sigma^2$. So let us set $T = (\ln 2 + 2\epsilon)4k\sigma^2$. With these choices of parameters, the probability of Error Event 1 will be:

$$\begin{aligned} P(\widetilde{Y}_{true} < L) &< e^{-\left(\frac{\sqrt{\frac{T}{2}}-L}{\sigma}\right)^2/2} \\ &= e^{-\left(\frac{\sqrt{(\ln 2+2\epsilon)2k\sigma^2}-\sqrt{(\ln 2+\epsilon)2k\sigma^2}}{\sigma}\right)^2/2} \\ &= e^{-\left(\sqrt{2k}(\sqrt{\ln 2+2\epsilon}-\sqrt{\ln 2+\epsilon})\right)^2/2} \\ &= e^{-\alpha k} \qquad \text{with } \alpha = (\sqrt{\ln 2 + 2\epsilon} - \sqrt{\ln 2 + \epsilon})^2 > 0 \end{aligned}$$

Which also tends to 0 as $k$ increases.

To summarize, if we wish to send $k$ bits across a channel, we can send a signal with energy [1]

$$E = ||x_i||^2 = \frac{T}{2} = (\ln 2 + 2\epsilon)2k\sigma^2$$

And be able to set an appropriate threshold that will decode the received signal with error probability tending to 0. Notice that the energy $E$ is proportional to the number of bits $k$.

## 5.3 Implications

Let us take a step back and see what we have managed to do. In OOK signaling, where each bit is sent separately, the only way to decrease the probability of error is to increase the energy-per-bit used.

Here, however, we have a scheme where the energy-per-bit $\frac{E}{k}$ remains a **constant**, and yet the probability of error tends to 0 as the number of bits sent increases! Somehow, packing in more bits lets us decrease the probability of error for all the bits, while using the same energy-per-bit! [2]

Intuitively, orthogonal signaling protects bits by sending them all together, instead of splitting them apart (as OOK does). When bits are sent together, we can afford to send the message at a higher energy, since we are sending more bits. This higher energy signal is then easier to distinguish from the noise. (Increasing the energy in an OOK message, however, cannot be done without increasing the energy-per-bit). This scheme is especially effective because all the messages are orthogonal ("far apart"), and the noise decays too rapidly to corrupt them (push them closer together).

This idea of bits gaining protection from traveling together is a general theme of coding, and will arise in different forms in many other coding schemes. Orthogonal signaling is a simple example of this, but not a very practical one: Using our described scheme (with T = 1 sec), sending 30 bits-per-second would require $2^{30}$ frequencies, or a bandwidth of $\approx 1$ GHz. In the next lectures, we will look at ways to improve bandwidth usage by relaxing the constraint of perfect orthogonality.

[1] Although our derivation seemed to use loose bounds, the final result is actually very accurate. In fact, for noise energy $\sigma^2 = \frac{\mathcal{N}_0}{2}$, there is a general result that the energy-per-bit required is $\frac{E}{k} = (\ln 2)\mathcal{N}_0$ – in exact agreement with our results!

[2] Notice that our analysis did not depend on the specific form of the orthogonal signals – in fact it holds for any orthogonal basis with a constant energy per signal $E = ||x_i||^2$. For example, pulse-position-modulation (PPM) is another form of orthogonal signaling, which uses a basis of $2^k$ shifted square pulses.