| EE 121: Coding for Digital Communications and Beyond | Fall 2013 |
|---|---|
| Lecture 3 — September 5 | |
| *Lecturer: Prof. Anant Sahai* | *Scribe: Rishi Sharma* |

This lecture covers:

- Axes on which communications problems vary
- Communicating one-bit in continuous time

## 3.1 Degrees of Freedom

Communication is about transmitting bits from one place to another. This transmission can be done through space (voice from one cell phone to another) or through time (accessing the data on a hard drive at some point in the future). Communications problems have 4 major axes on which they can vary:

- Random Noise vs. Adversarial Noise
- One Bit vs. Many Bits
- Discrete Time vs. Continuous Time
- Discrete Alphabet vs. Continuous Alphabet

It is important to recognize that any combination can make for an interesting and/or practical communications problem.

**Ex 1.** Consider Paul Revere's ride during the American Revolution. Revere tells a friend to go to the Old North Church and prepare signal lanterns to communicate how the British are attacking. He is told to light "one (lantern) if by land, two if by sea." Here the friend is communicating **one** (very important) **bit** across a channel with **random noise** in **discrete time** with a **discrete alphabet**.

**Ex 2.** For a more familiar example, consider Reed-Solomon codes, in which we transmit points in a finite field in such a way that our receiver can recover from malicious errors: transfer **many bits** across a channel with **adversarial noise** in **discrete time** using a **discrete alphabet** (discrete number of points in finite field).

It is often easy to see how the engineering challenges change when dealing with random vs. adversarial noise or communicating one bit vs. many bits. However, it is not as intuitive to see how problems change from discrete time to continuous time, so the remainder of the lecture notes will explore **communicating one bit in continuous time**.

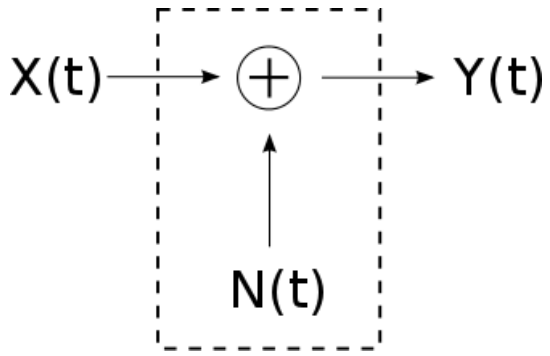


**Figure 3.1.** Additive noise model where X, Y, and N are signals in continuous time.

## 3.2 Communicating One Bit in Continuous Time

When communicating in continuous time, we must map something fundamentally discrete (one bit) to a symbol that evolves through time. In order to explore the intricacies of continuous time communication, we use the illustrative example of **OOK (On-Off Keying) through a continuous time channel with additive noise**. We will encode the bit value 0 as $X_0(t)$ and the bit value 1 as $X_1(t)$ where

$$X_0(t) = 0 \qquad X_1(t) = A\sin(2\pi f_c t)$$

These states can be thought of as voltages of 0 or $A$ that are subsequently modulated by a sinusoid and transmitted across a noisy channel (see figure 3.2). The output $Y$ will be a sum of the input signal and the noise $N$

$$Y_0(t) = N(t) \qquad Y_1(t) = A\sin(2\pi f_c t) + N(t)$$

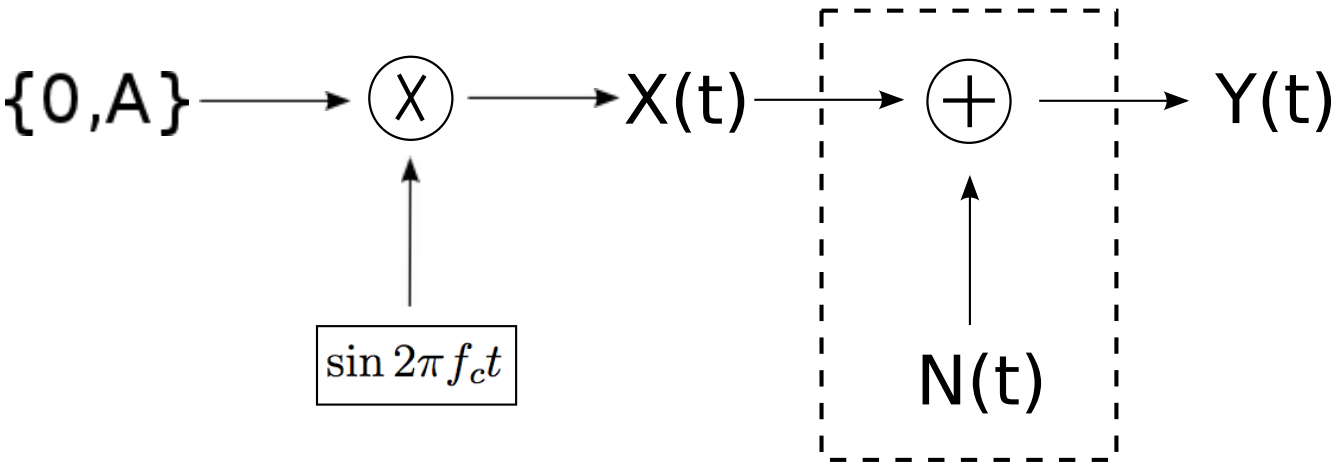


**Figure 3.2.** Illustration of the OOK modulation scheme chosen.

So how do we decode the continuous signal we receive, $y(t)$? We will explore a few ideas.

### 3.2.1 Decoding Idea 1: Modulate Back to Baseband

Without the noise, this encoding scheme is essentially amplitude modulation (review notes from EE20/120), so one idea is to look to AM decoders for inspiration. When decoding AM, we modulate the output signal $y(t)$ with the same frequency that was used to encode it, shifting the spectrum of the output back to baseband. This allows us to recover a noisy version of the original voltage (0 or $A$) using a low-pass-filter.

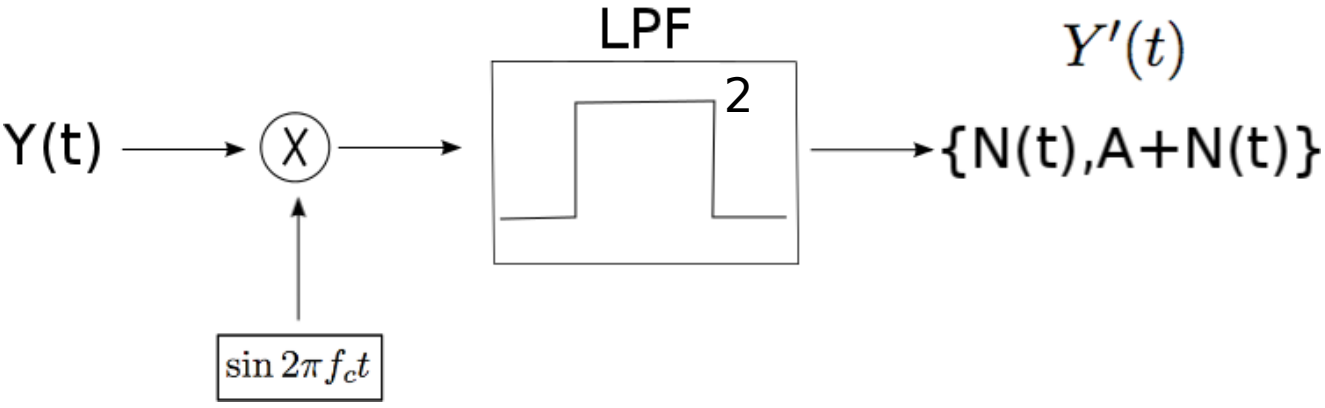


**Figure 3.3.** If $N(t) = 0$ (i.e. communication through a noiseless channel), $Y'(t) = 0$ or $Y'(t) = A$.

Decoding results in the continuous time output $Y'(t)$. This forces us to consider a reality of continuous time communication: **analog processing is hard**. The general strategy used to circumvent this problem is to discretize our signals. It is much easier to use a computer and decode the transmission digitally, so we sample our output $Y'(t)$ to get $\vec{Y}'$.

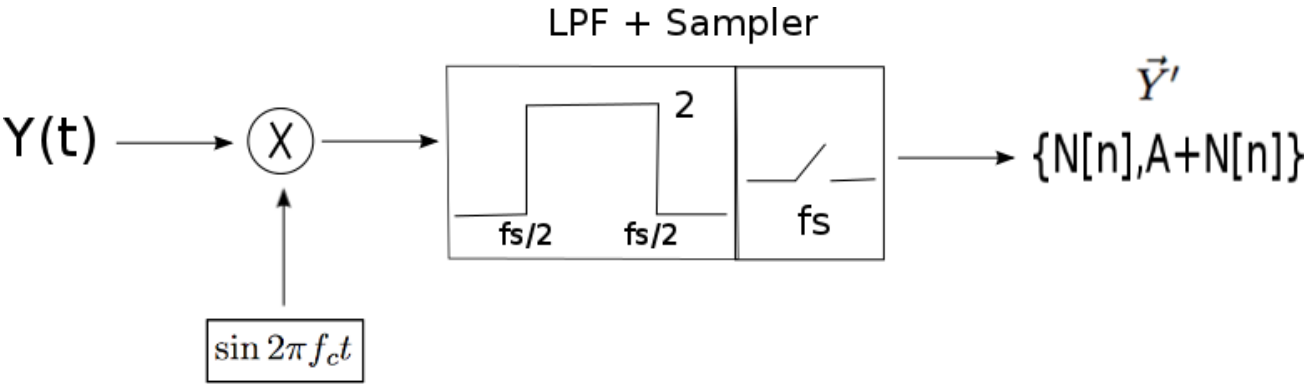


**Figure 3.4.** The number of samples $\vec{y}_i \in \vec{Y}'$ collected depends on the sampling frequency $f_s$ and the duration of time $L$ for which samples are collected. Note that the sampling frequency $f_s$ is twice the cutoff frequency of the low-pass-filter.

In the last stage of the process, we hope to determine the bit that was sent from the series of discrete samples $\vec{Y}'$. Let us denote $\vec{Y}'_0$ and $\vec{Y}'_1$ to be $\vec{Y}'$ if 0 or 1 were sent respectively.

Observe

$$\vec{Y}_0' = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} + \begin{bmatrix} n_0 \\ n_1 \\ \vdots \\ n_m \end{bmatrix} \qquad \vec{Y}_1' = \begin{bmatrix} A \\ A \\ \vdots \\ A \end{bmatrix} + \begin{bmatrix} n_0 \\ n_1 \\ \vdots \\ n_m \end{bmatrix}$$

Where $n_i$ is the $i^{th}$ sample of the noise $N(t)$.

Assuming the noise is random and "reasonable" (i.e. the noise is such that 0's are probably still closer to 0 than $A$ and 1's are probably still closer to $A$ than 0), a natural thresholding technique would be to average the values in $\vec{Y}$ and put a threshold at $\frac{A}{2}$. If the average is greater than $\frac{A}{2}$, then we decode a 1, otherwise a 0.

It is important to note that by collecting infinite samples of $Y'(t)$, your probability of error will go to zero (under the assumption of "reasonable" noise). Though we will have succeeded in decoding the bit, this solution is impractical. In the real world, we are interested in communicating many bits across a channel, and waiting infinitely long for one bit is simply not feasible. The astute reader will also have noticed that by making the sampling frequency arbitrarily high, we also can collect almost infinite samples and thus decode the bit with probability of error approaching zero. This turns out not to be the case, however, for reasons having to do with the interaction of sampling and noise. Before we can discuss this interaction we must address a topic we have been avoiding so far: the nature of noise in continuous time.

### 3.2.2 Modeling Noise in Continuous Time

In order to accurately model our noise variable $N(t)$, let us philosophically consider two basic and powerful properties of noise (which can also be experimentally verified):

1. Noise in nature is not adversarial. In other words, noise hates all signals equally and is not "out to get" any particular signal you are transmitting. An equivalent way of stating this property is that noise hates all frequencies equally, which suggests that $N(t)$ **have an equal power within any frequency band of a fixed width**. Noise which satisfies this property is known as **white noise**

2. Noise is the sum of many small noise sources. We can better understand this property by investigating the major source of noise in our communication model. The noise in our system can predominantly be explained by thermal fluctuations from the circuitry that is producing, modulating, and decoding our signal. Fundamentally, these thermal fluctuations are coming from a random number of electrons constantly dancing through our circuit in every direction. Each of these electrons can be modeled as an **I.I.D random voltage** that represents a **small noise source contributing to the overall**

**noise**, thus invoking the **Central Limit Theorem**, which tells us the distribution of the noise will be **Gaussian**. Moreover, we can plausibly assert that the **noise will have zero** mean because voltages arise as a differential value between two points, and the random dance of electrons should not prefer one point any more than another.

Thus, we've convinced ourselves that our noise $N(t)$ can be modeled as white Gaussian noise with zero mean. This class of communications channels we are analyzing are known as **additive white Gaussian noise** (AWGN) channels.

### 3.2.3 The Relationship Between Noise and Sampling

Thus far we have learned that noise can be modeled as a zero mean Gaussian, so let us now suppose that it has some fixed variance $\sigma^2$. As was noted earlier, we can make the probability of error in our transmission arbitrarily low by collecting infinite samples of the received signal and, because the noise has zero mean, average the samples to easily decode the transmitted message. One way of collecting infinite samples is to make the sampling frequency infinitely high. Disregarding all physical limitations in our ability to sample arbitrarily fast, this notion that a higher sampling frequency will make our transmission more robust should violate our intuition of the Shannon-Nyquist sampling theorem, which tells us that we can perfectly reconstruct a signal by sampling at twice the speed of the highest frequency in our transmitted signal (Nyquist rate). Increasing the sampling frequency beyond the Nyquist rate should not provide any added benefit in reconstruction, so we have reason to suspect that our noise model is flawed. It is also empirically found that increasing the sampling frequency seemed to result in a noisier signal.

It for this reason that we model the variance of the noise to be proportional to the sampling frequency, thus making the distribution of the sampled noise $N_i \sim N(0, \sigma^2 f_s)$ (where $N_i$ is the $i^{tth}$ sample of the noise). This view ensures that we can no longer improve our communication channel by sampling the output arbitrarily fast. This conclusion, however, obfuscates what is actually happening as our sampling frequency goes to infinity. In reality the noise variance is independent of sampling frequency. This is because the value of a signal $N(t)$ at time $nT_s$ (where the sampling period $T_s = \frac{1}{f_s}$) is not simply equal to $N(nT_s)$ in real life sampling, for the reason that, in order to sample, a capacitor in an analog-to-digital converter circuit must be charged up by the input signal and the voltage read off to determine the sample value at that time. As we make the sampling frequency faster, the capacitor has less time to charge up, **thus making the signal values weaker**. Though the signal is weaker, the noise from random electron movement remains the same, and so our **signal-to-noise-ratio in each sample decreases, and because we are collecting more samples, the overall SNR is unchanged** (recall the signal-to-noise ratio is the ratio of signal energy to noise energy). It is inconvenient to have the signal values dependent on sampling frequency, so in real life circuits there is a gain stage after the sampling block to normalize the signal energy across all possible sampling frequencies. The result is that the noise is gained along

with the signal, and the noise variance increases. This is how we arrive at our noise model $N_i \sim N(0, \sigma^2 f_s)$ where the variance in the noise increases with sampling frequency.

### 3.2.4 Decoding Idea 2: Matched Filtering

Now that we are endowed with a sufficient understanding of noise, we can more thoroughly analyze another method for our receiver to decode $y(t)$. **Matched filtering** encompasses a general class of filters which **correlate** noisy received signals with their noiseless counterparts to determine which signal was sent. In our case, we can correlate the output $Y(t)$ with $X_1(t) = \sin(2\pi f_c t)$ to compute how "similar" they are. If they are found to be sufficiently "similar" (according to some threshold), then we decode a 1. Otherwise we decode a 0.

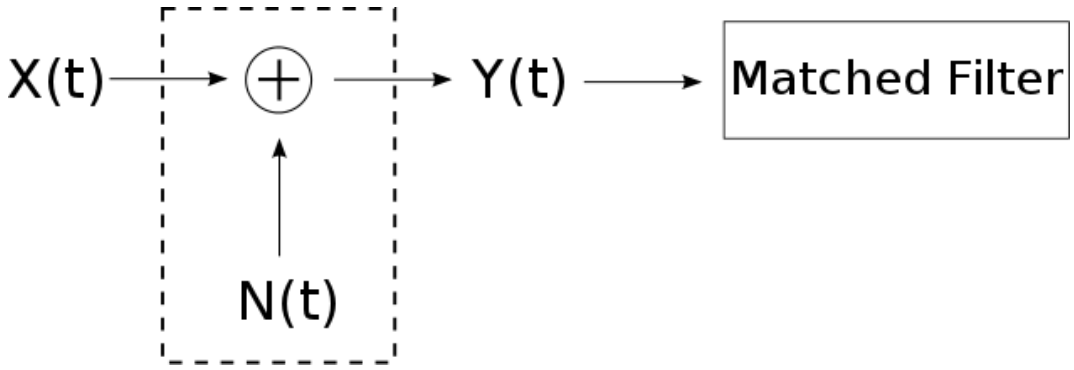


**Figure 3.5.** The signal-to-noise ratio at the output of the matched filter will be $SNR_{max}$.

In the past such comparisons were implemented as analog filters, which look fairly different from the digital implementations of today. The principal, however, is the same: the matched filter should satisfy the **SNR maximization property**, which states that the output of the matched filter should maximize signal-to-noise-ratio. It can be proven in the analog case (and is done so in most texts on communications) that in an AWGN channel.

$$SNR_{max} = \frac{2}{\mathcal{N}_0}||x(t)||^2 \qquad \text{where } \frac{\mathcal{N}_0}{2} = \sigma^2 \text{ (i.e. variance of noise)}$$

We will derive a simpler, digital matched filter which instead performs the inner product $\langle \vec{Y}, \vec{h} \rangle$ with some $\vec{h}$ that will allow us to achieve $SNR_{max}$. Note that $\vec{y} = \vec{x} + \vec{n}$ is composed of both a signal and noise vector and that the inner product defined on our space will simply reduce to the dot product because $\vec{y}$ and $\vec{h}$ will be real:

The energy in our signal

$$\text{Signal Energy} = |\langle \vec{x}, \vec{h} \rangle|^2$$

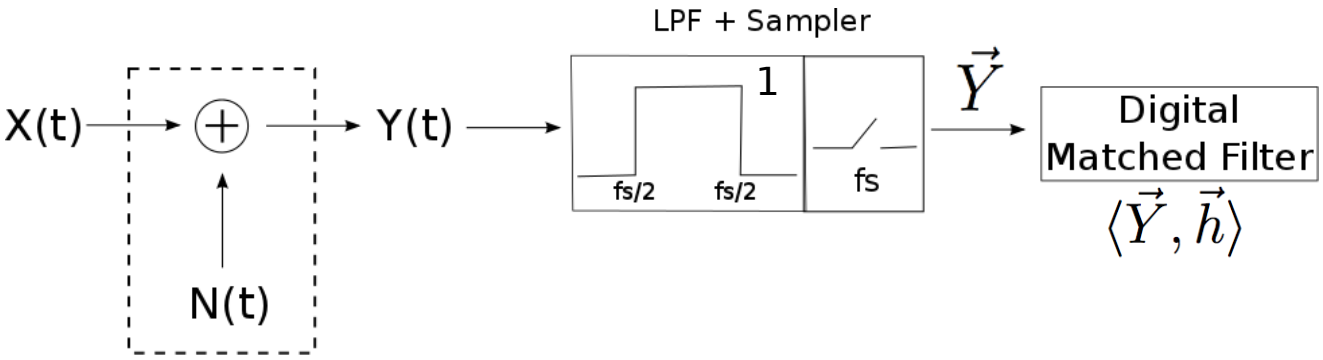


**Figure 3.6.** The digital matched filter used in our example computes $\langle \vec{Y}, \vec{h} \rangle$ where $h$ is chosen to be $x_1$. Then we compares this value to a threshold to determine the bit communicated.

The expected noise energy at the output of the matched filter

$$\begin{aligned}
\text{Noise Energy} &= \mathbb{E}\left[|\langle \vec{N}, \vec{h} \rangle|^2\right] \\
&= \mathbb{E}\left[(n_1 h_1 + n_2 h_2 + \cdots + n_m h_m)^2\right] \\
&= \mathbb{E}\left[n_1^2 h_1^2 + n_2^2 h_2^2 \cdots + n_m^2 h_m^2 + n_1 h_1 n_2 h_2 + n_1 h_1 n_3 h_3 + \cdots + n_{m-1} h_{m-1} n_m h_m\right]
\end{aligned}$$

where $n_i$ are samples of a Gaussian with mean 0, so $\mathbb{E}[n_i] = 0$. By linearity of expectations we can separate all terms and see that all of them containing a $\mathbb{E}[n_i]$ go to zero. We are left with the expression:

$$\begin{aligned}
\text{Noise Energy} &= \mathbb{E}[n_1^2] h_1^2 + \mathbb{E}[n_2^2] h_2^2 + \cdots + \mathbb{E}[n_m^2] h_m^2 \\
&= \mathbb{E}[N^2] ||h||^2 \\
&= \frac{\mathcal{N}_0}{2} ||h||^2
\end{aligned}$$

The signal-to-noise ratio, defined as the ratio of the signal power to the noise power, equals

$$SNR = \frac{2}{\mathcal{N}_0} \cdot \frac{|\langle \vec{x}, \vec{h} \rangle|^2}{||h||^2}$$

The Cauchy-Schwarz Inequality states that

$$\langle \vec{x}, \vec{h} \rangle^2 \leq ||x||^2 ||h||^2$$

with equality if and only if $\vec{h}$ is parallel to $\vec{x}$, which will happen if we choose $\vec{h} = k\vec{x}$, where $k$ is some arbitrary nonzero constant. Thus we get the result

$$SNR = \frac{2}{\mathcal{N}_0} \cdot \frac{||x||^2 ||x||^2}{||x||^2} = \frac{2}{\mathcal{N}_0} ||x||^2 = SNR_{max}$$

This choice of $\vec{h}$ assumes we know what $x(t)$ will look like when sampled at $f_s$ (namely, $\vec{x}$), so we will have to keep track of this vector in order for the matched filter to work. By projecting $\vec{Y}$ onto $\vec{X}$, we extract a real number $\langle \vec{y}, \vec{x} \rangle$ which maximizes signal-to-noise-ratio. We can now apply a threshold to decode.

### 3.2.5 Setting a Threshold

To determine how to set the threshold on our matched filter, we should examine the distribution of $\langle \vec{Y}, \vec{x_1} \rangle$. If a 0 bit is sent,

$$\begin{aligned} \langle \vec{Y_0}, \vec{x_1} \rangle &= \langle \vec{N}, \vec{x_1} \rangle \\ &= [N_1, \cdots, N_m] \cdot [x_1, \cdots, x_m]^\top \\ &= \sum_{i=1}^{m} N_i x_i \end{aligned}$$

where each $N_i x_i \sim N(0, x_i^2 \sigma^2)$ because all $x_i$ are fixed constants. Remember that when summing multiple Gaussian random variables each with mean $\mu_i$ and variance $\sigma_i^2$, the resulting distribution is also Guassian with mean $\sum_i \mu_i$ and variance $\sum_i \sigma_i^2$. Thus, the distribution of $\langle \vec{Y_0}, \vec{x_1} \rangle \sim N(0, \sum_{i=1}^{m} x_i^2 \sigma^2) = N(0, ||x||^2 \sigma^2) = ||x|| \times N(0, \sigma^2)$.

$$\begin{aligned} \langle \vec{Y_1}, \vec{x_1} \rangle &= \langle \vec{x_1} + \vec{N}, \vec{x_1} \rangle \\ &= [x_1, \cdots, x_m] \cdot [x_1, \cdots, x_m]^\top + [N_1, \cdots, N_m] \cdot [x_1, \cdots, x_m]^\top \\ &= ||x||^2 + \sum_{i=1}^{m} N_i x_i \end{aligned}$$

and thus $\langle \vec{Y_1}, \vec{x_1} \rangle \sim N(||x||^2, \sigma^2 ||x||^2) = ||x|| \times N(||x||, \sigma^2)$.

Because both Gaussians are scaled by the same constant, we can disregard it and find that we have two Gaussian distributions with identical standard deviations, where one is shifted to $||x||$. We set our decoding threshold at $\frac{||x||}{2}$ to minimize the probability of error.

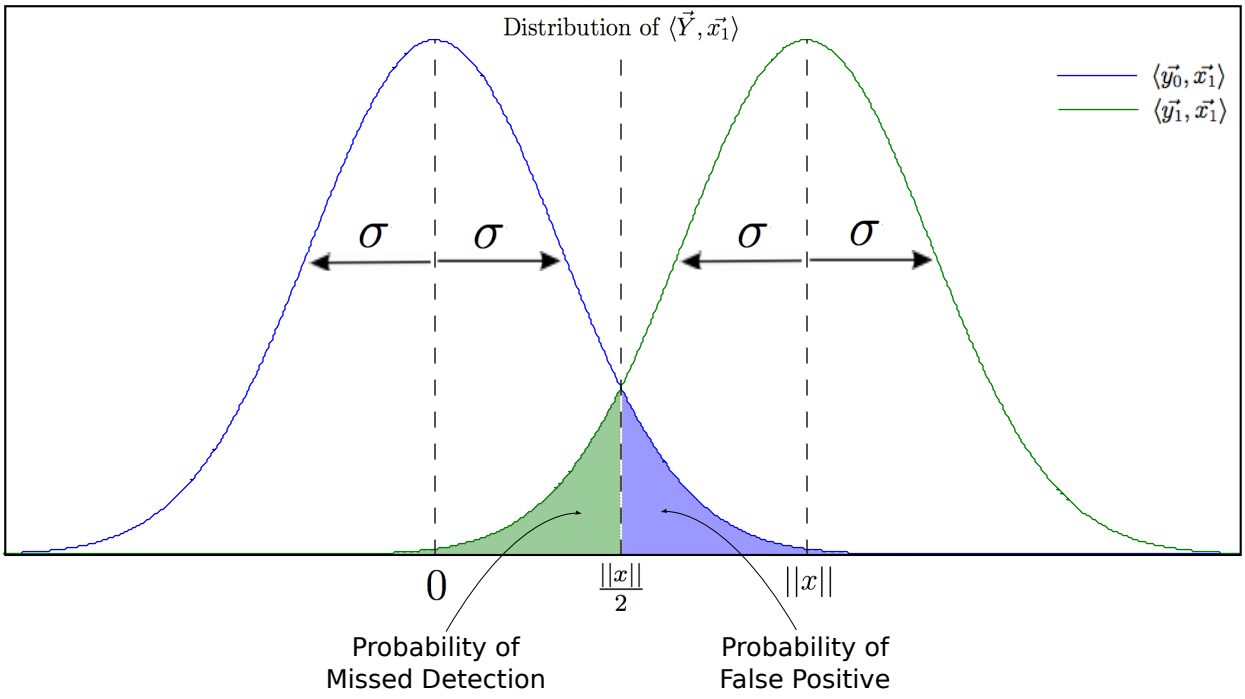


**Figure 3.7.** The point where two Gaussians with identical standard deviation $\sigma$ cross is at the point halfway between them. The analysis of AWGN in continuous time turned out to be remarkably similar to our analysis of Gaussian noise in discrete time.