Digital Communications and Orthogonality

Matlab Case Study for Signals and Systems (Draft)

In the modern age we are surrounded by invisible signals. Radio transmissions, wifi signals, and cell phone towers are constantly bouncing messages back and forth. But how are these messages sent, and how do they avoid interfering with one another? How can we improve their efficiency and accuracy? To answer all these questions, we require a solid understanding of pulse shaping, orthogonality, and signal correlation.

In this case study, you will explore the concepts of pulse shaping and orthogonality, and how they relate to the modulation, transmission, and reception of digital signals. You will use this knowledge to design a pulse shape that satisfies the Nyquist Filtering Criteria and examine its properties in both the time and frequency domain using the Fast Fourier Transform algorithm.

Finally, you’ll use your pulse shape to encode an ASCII text message in a binary Pulse Amplitude Modulation (PAM) scheme. You’ll share your pulse shape and encoded signal with another student, who will use that information to receive your original message.

# Orthogonality in a Nutshell

A short definition of orthogonality is that any set of *n* signals are orthogonal to one another if, when added together as a linear combination, they can then be separated out again. This idea has some similarity to the concept of linear independence of vectors.

You may recognize the term “orthogonal” from physics or calculus. If so, you’ll recall that we can test if two vectors are orthogonal by taking their inner product, or dot product. If it is zero, the vectors are orthogonal. Good news! We have a similar operation, also called the inner product, that we can perform on signals. The inner product of two signals over an interval *a, b* is defined as:

(We sometimes add a weighting function *w(x)* to this definition, but for this case study *w(x) = 1* and so we can disregard this detail.)

Just as orthogonal vectors have an inner product of zero, so do orthogonal signals.

Definition:

Two signals f(x) and g(x) are orthogonal over the interval [a, b] if and only if:

It’s critical that we use orthogonal signals in digital communications, as our signals will often overlap and if they aren’t orthogonal, we can’t be certain our receiver will be able to tell them apart.

# Pulse Amplitude Modulation in a nutshell

There are many different ways to encode information, but for this case study we will be using a method called Pulse Amplitude Modulation, or PAM. PAM works by using a basic pulse shape and transmitting it over and over again, varying the amplitude each time. The receiver can detect these changes in amplitude and decode them to receive the original message.

Here’s the basic idea: We convert the message we want to send into discrete packets, called symbols. Each symbol represents some sequence of bits. For example, in the diagram below, each symbol represents some 3-bit sequence, and is equivalent to some amplitude, represented here as a scalar multiple of some constant A.

A picture containing object, clock

Description automatically generated

At regular periods of time (the symbol period) the transmitter multiplies the pulse shape by the symbol amplitude and sends it to the receiver. Because the pulse shape is orthogonal to a time-delayed version of itself, (more on that later) the receiver can separate out each symbol and convert them back into binary, and the original message can be reconstructed.

# Pulse Shaping in a nutshell

So what is this pulse shape that we are using to encode our message? There are lots of different shapes we could use, but whatever we choose, we want it to have a few important properties:

* We want our pulse’s Fourier transform to occupy a small frequency band, so that it will not interfere with (or be interfered with by) other transmissions in the area.
* We want our pulse to occupy a short time duration, so that we can send it in a reasonable amount of time. If our pulse is too long, we might have to shave some off it off and lose some accuracy.
* We want our pulse to be orthogonal to delayed versions of itself, so that if they overlap our receiver can still tell the difference between them.

However, as you’ll discover in the case study, the first two two goals are to some degree mutually exclusive. Part of your job in this case study will be devising a pulse shape that finds an acceptable trade-off between them.

# The Nyquist Filtering Criteria

The Nyquist Filtering Theorem says that we can determine whether a pulse is orthogonal to a time-delayed version of itself by examining the pulse’s Fourier transform. The mathematical definition of the Nyquist Filtering Criteria is show below:

Essentially, this states that if we add the pulse shape to a time-delayed version of itself, (where the delay is an integer multiple of the symbol period) we get a constant value. If this condition is satisfied, our pulse shape satisfies the criteria, and we can use it to send messages without worrying about past and future symbols interfering with the current one.

# Case Study

Using the *pulse\_shaping.m* script as a template, complete the following tasks:

* Devise a pulse shape in either the time domain or the frequency domain. Using the fft() and ifft() functions, ensure that you have representations of your pulse shape in both domains.
* Determine the autocorrelation function of your pulse shape. The autocorrelation is the convolution of the signal with itself. Verify that the autocorrelation function has a value of 0 or nearly 0 for all integer multiples of the symbol period. Show via MATLAB code or explain mathematically why this is equivalent to the idea that the pulse shape is orthogonal to

# Extension

If you’re interested in learning more, you can try these additional challenges.

* Convolve your pulse shape with a high frequency sine wave. Compare the frequency spectrum of your pulse before and after. How might you use this property to send multiple signals while ensuring they don’t overlap in the frequency domain?

Explain the two extremes of pulse shapes:

1. a rectangular pulse shape. Problem: infinite in frequency domain, so uses a lot of bandwidth
2. a function which is rectangular in the frequency domain. This results in a sinc function in the time domain, which has lots of ringing. Although it is infinitely long in the time domain, devices typically cut off the time domain signal after several symbol periods. But with a sinc function, one needs a lot of time before and after the center of the pulse to capture enough of it.

For both of these pulse shapes, because the slope of the pulse is high, time synchronization is a problem -- a little bit of time error leads to a big error in the receiver’s ability to separate one pulse from the next.

A receiver separates two orthogonal symbols by using correlation and/or filtering

* Show an example of this via Matlab functions
* The functions could show a M-PAM system (for simplicity, wouldn’t use QAM)
* Explain the code / link to the code so they can use these functions.

This project is to design your own pulse shape.

The Nyquist sampling theorem gives a method involving the design of a pulse shape p(t), which has the property that it is orthogonal to all delayed versions of itself that are multiples of T\_s, the symbol period. So the receiver can perfectly separate one symbol from the others sent before and after itself.

The steps might be:

1. Create a shape in the frequency domain for R\_p(f), the power spectral density of p(t) (the Fourier transform of the autocorrelation function of p(t), if they haven’t learned the PSD yet), that meets the Nyquist filtering theorem criteria (which is about symmetry, it shouldn’t be too complicated).
2. Show that R\_p(f) meets the Nyquist Filtering Theorem. That is, that …+ R\_p ( f – 2/T\_s ) + R\_p ( f – 1/T\_s ) + R\_p ( f ) + R\_p ( f + 1/T\_s ) + R\_p ( f + 2/T\_s ) + … is equal to a constant.
3. Take the square root of R\_p(f) in the frequency domain
4. Take the inverse Fourier transform to get p(t)
5. Find the autocorrelation of p(t)
6. Show that this autocorrelation is equal to 0 at m\*T\_s, where T\_s is the symbol period.
7. Run your pulse shape through the transmitter and receiver to show that it works