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.

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 is 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 taking 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 bits – ones and zeros – and then convert those bits 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.

For example, if we use the symbol map shown above and we want to send the bit sequence “110101”, we would break it up into two symbols: “110” maps to +A and “101” maps to +5A. We would send our pulse twice, first multiplying its amplitude by A and then by 5A. The receiver can then tell the pulses apart, recover the symbols, convert them into bits, and reconstruct our original message!

In this lab, we will be using a *binary* Pulse Amplitude Modulation scheme. This means the symbols map contains only two symbols: one to send a 1, and one to send a 0.

A close up of a clock

Description automatically generated

# Pulse Shaping in a nutshell

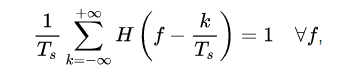
So what is this pulse shape that we are using to encode our message? There are lots of different shapes we could use, but whatever we choose, we want it to have a few important properties:

* 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 versions of itself that have been delayed by an integer multiple of the symbol period, 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 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 shifted versions of itself (where the shift is equal to the symbol frequency) 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.

This might be hard to picture so here’s a quick example. Consider a pulse shape with a Fourier transform that is mostly rectangular, with the edges rounded off. The symbol period is 1/50 seconds, so the symbol frequency is 50 Hz.

A screenshot of a cell phone

Description automatically generated

To test the Nyquist Filtering Criteria, we add versions of the pulse that have been shifted by 50 Hz in both directions. The actual criteria asks us to plot an infinite number of shifts, but we can plot just two - one shifted up 50 Hz, and one shifted down 50 Hz – and that will be enough to see the pattern.

A screenshot of a cell phone

Description automatically generated

Note that because our pulse is wider than 50 Hz, there is some overlap between the two plots. However, when we take the superposition of all three plots…

A screenshot of a cell phone

Description automatically generated

They add up to a constant value! We could continue adding frequency shifts and see if the superposition continues to be constant:

A picture containing game

Description automatically generated

But for our purposes this is enough. This pulse shape satisfies the Nyquist Filtering Criteria.

Some warnings:

* We only needed to plot one pair of frequency shifts to verify the criteria because our pulse was narrow. If your pulse’s frequency domain is wider than twice the symbol frequency, you will need to plot additional iterations.
* Your pulse may not add up exactly to a constant, especially if your sample rate is not much larger than your symbol rate or if your pulse includes hard edges. Use your best judgement on whether your pulse satisfies the Nyquist Filtering Criteria

# Case Study

Using the *pulse\_shaping.m* script as a template, complete the following tasks. The MATLAB script includes comments and some starter code to assist with each task.

* Devise a pulse shape in both the time domain and the frequency domain based on the design criteria described above, mainly:
  + Narrow frequency band
  + Short time duration
  + Orthogonal to versions of itself delayed by the symbol period
* Ensure you have time and frequency representations of your pulse.
* 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 time-delayed versions of itself.
* Determine the frequency representation of the autocorrelation function and compare it to the frequency representation of the pulse shape itself. What do you notice? How can this be explained by properties of convolution?
* Verify that your pulse shape satisfies the Nyquist Filtering Criteria.
* Test your pulse shape using the encode() and decode() functions. Does it successfully transmit messages?

# Extension

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

* Use the encode() function and a custom message to create a binary Pulse Amplitude Modulated signal. Trade your output signal, pulse shape, and information about the sample rate and symbol rate with a classmate. Use the decode() function to recover the other student’s message. Comment on how effectively their pulse shape meets the desired criteria:
  + Short duration
  + Narrow frequency bandwidth
  + Orthogonal to time-delayed versions of itself
  + Meets the Nyquist Filtering Criteria
* Multiply your pulse shape with a high frequency cosine 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?
* Examine the encode() and decode() functions. In your own words, explain how and why they work.

# What to Turn In

* Your completed pulse\_shaping.m script
* A writeup of your observations and notes
* Any additional functions you wrote for this project. Do not include the encode.m and decode.m functions.