DIGITAL COMMUNICATIONS AND ORTHOGONALITY

Matlab Case Study for Signals and Systems (Draft)

Introduction

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:

1. Design a digital pulse shape in both the time and frequency domain to satisfy several design criteria, such as minimizing bandwidth.
2. Explore properties of the autocorrelation function in both the time and frequency domain
3. Use the properties of orthogonality and convolution in the time domain to verify that your pulse shape can be used to send messages without inter-symbol interference.
4. Use the Nyquist Filtering criteria in the frequency domain to verify that your pulse shape can be used to send messages without inter-symbol interference.
5. Test your pulse shape using a Binary Pulse Amplitude Modulation scheme to transmit a custom message.

Orthogonality and the Inner Product

A short, intuitive 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, or to the concept of injective transformations. If you are familiar with these concepts, it may help to keep in mind what they tell you about the "loss of information" (or lack thereof) when performing certain operations.

[Visual aid here: Three panel "comic:" two signals added together in superposition, then separated out again]

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:

$$\langle f, g \rangle = \int_a^b f(x)g(x)dx$$

(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:

$$\int_a^b f(x)g(x)dx = 0$$

For instance, consider the two signals below: two sinc functions, one of which has been delayed slightly. The element-wise product of the two signals is shown to the right. The integral of this product is zero. Therefore, these two signals are orthogonal.

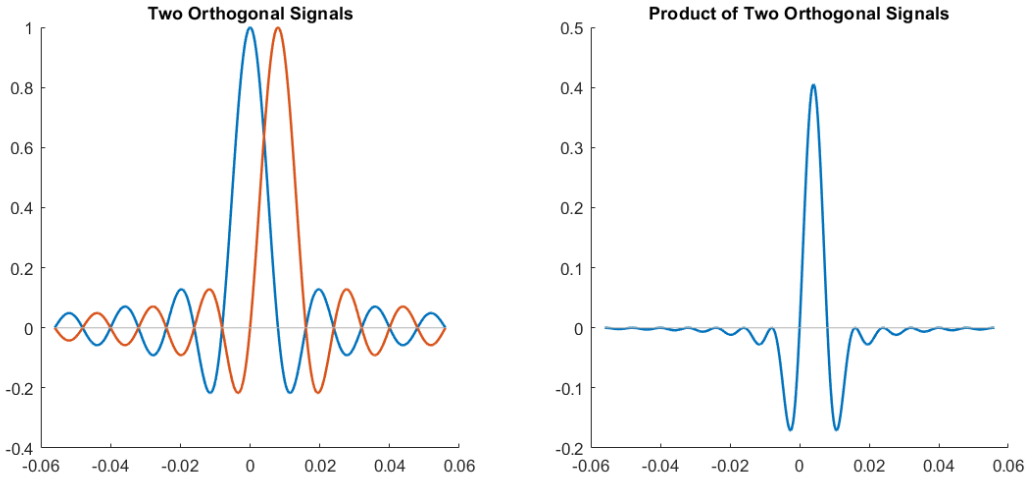


FIGURE 1: TWO ORTHOGONAL SIGNALS

Because these signals are orthogonal, we could transmit some linear combination of these two signals to a receiver, which could use a method called a "matched filter" to separate them out again and determine the coefficients of the linear combination used.

It might already have occurred to you that signals such as these which are orthogonal to time-delayed versions of themselves will be very useful for transmitting messages. We can transmit scalar multiples of such a signal repeatedly, at regular intervals, and no matter how much the signals overlap, our receiver will be able to tell them apart!

Pulse Amplitude Modulation

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 how it works:

Symbol Encoding

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.

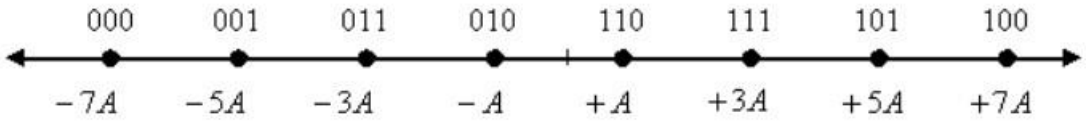


FIGURE 2: A SYMBOL MAP FOR A 3-BIT PULSE AMPLITUDE MODULATION SCHEME

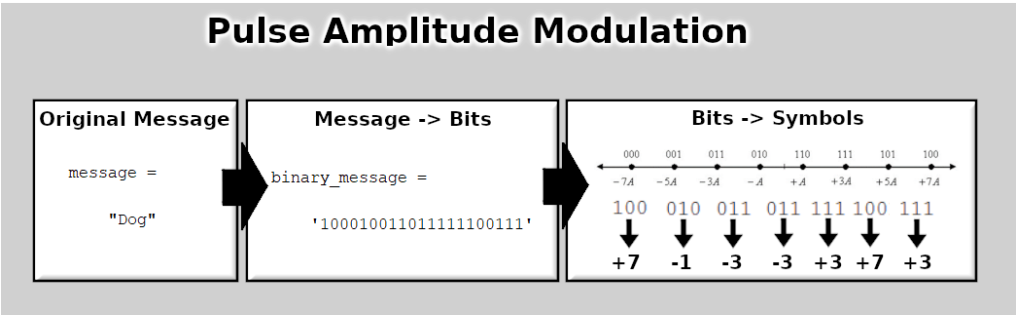


FIGURE 3: A MESSAGE IS CONVERTED INTO A SYMBOL SEQUENCE TO BE TRANSMITTED

Transmission

At regular periods of time (the symbol period) the transmitter multiplies the pulse shape by the symbol amplitude and sends it to the receiver. We can model this by convolving the pulse shape with the symbol sequence. What results is a superposition of several different pulses sent at uniform intervals: the transmitted signal.

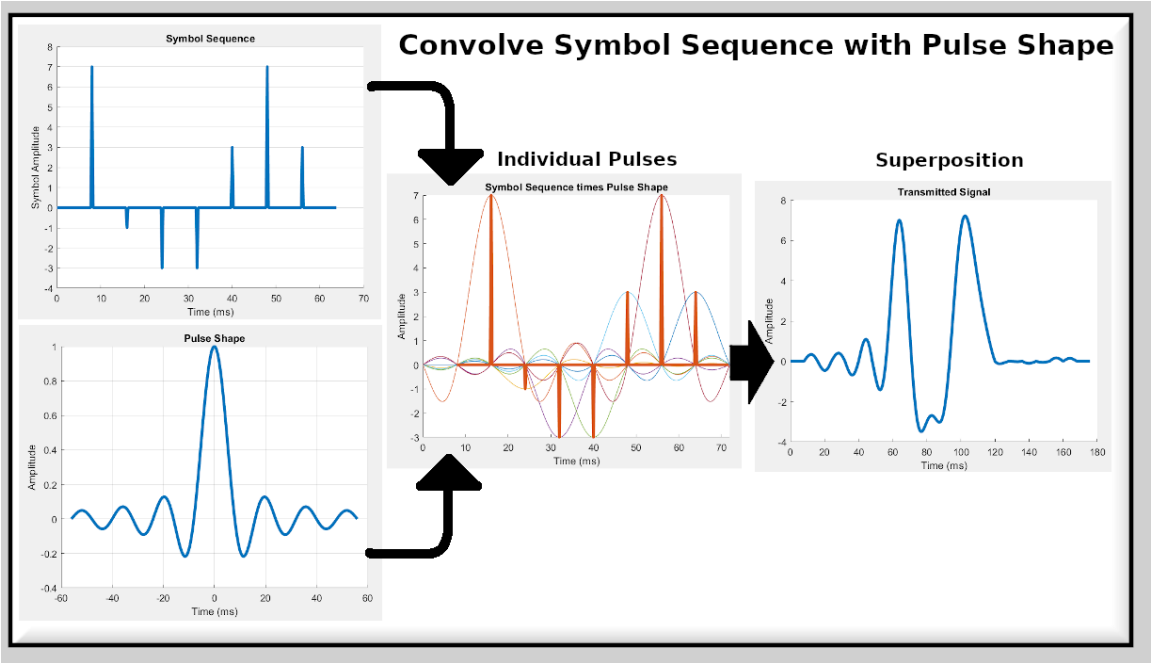


FIGURE 4: THE SYMBOL SEQUENCE IS ENCODED INTO THE PULSE SHAPE TO GET THE TRANSMITTED SIGNAL

Reception

Our transmitted signal is picked up by our receiver (give or take some random noise). The receiver then convolves the received signal with the original pulse shape, detecting regions of the signal that correlate to scalar multiples of the pulse shape. This is called a "matched filter." It works because the signal is strongly correlated with itself, and weakly correlated with versions of itself that have been delayed by an integer multiple of the symbol period. The matched filter separates out the past and future symbols from the current one, recovering the original symbol sequence. At this point, it can be mapped back to binary and the original message is recovered!

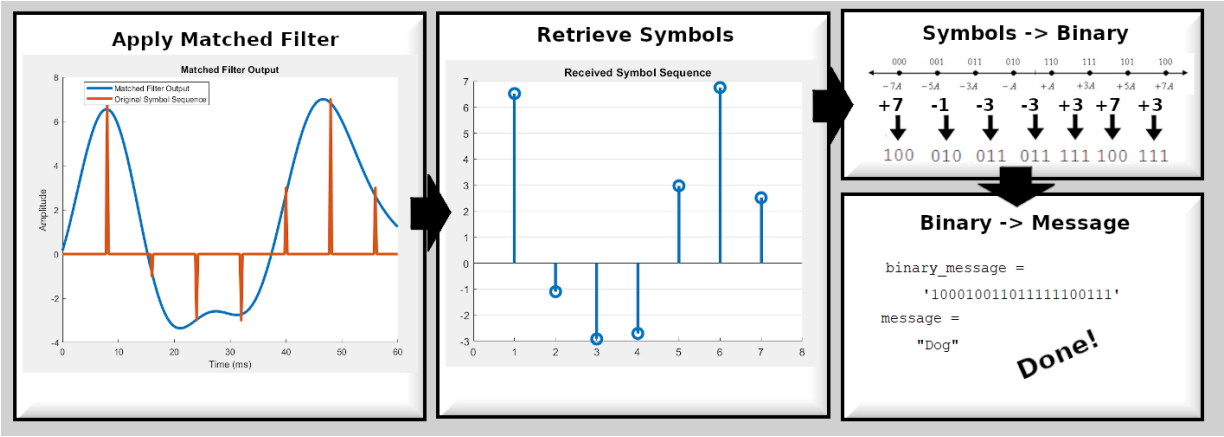


FIGURE 5: THE RECEIVER USES A MATCHED FILTER TO RETRIEVE THE ORIGINAL SYMBOL SEQUENCE AND RECOVER THE ORIGINAL MESSAGE

In this case study, 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. To send a one, we transmit our pulse shape normally. To send a zero, we transmit our pulse shape times negative one.

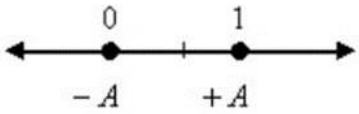


FIGURE 6: A SYMBOL MAP FOR A BINARY PULSE AMPLITUDE MODULATION SCHEME

Pulse Shaping: Desired Properties

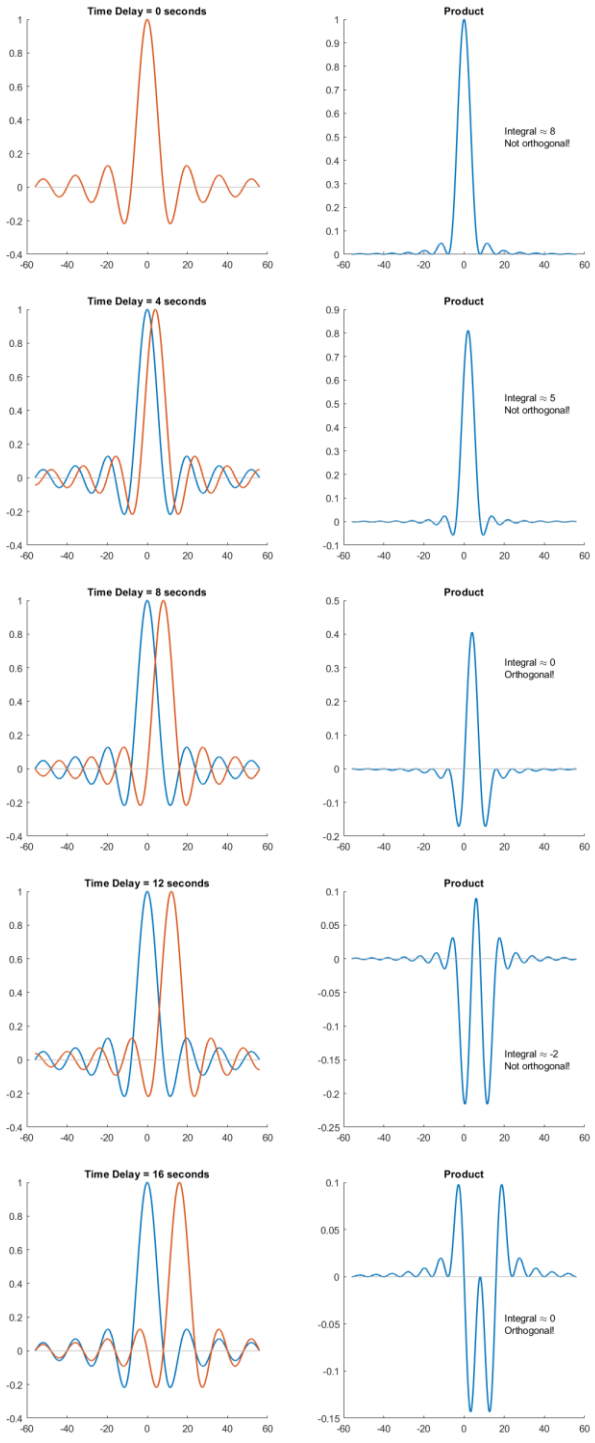
So what is this pulse shape that we are using to encode our message? There are lots of different signals we could use, but whatever we choose, we want it to have a few important properties:

- We want our pulse's Fourier transform to occupy a small frequency band (the "bandwidth), 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 and it won't overlap with past and future signals too much. 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. The narrower a signal gets in the frequency domain, the longer its duration becomes in the time domain. Part of your job in this case study will be devising a pulse shape that finds an acceptable trade-off between these design criteria.

Correlation and Orthogonality

So how do we test if a signal is orthogonal to a time-delayed version of itself? Recall that two signals are orthogonal if their inner product (the integral of the product of the two signals) is zero. We can imagine sliding two copies of our signal past one another over time. At each instant in time, we compute the inner product. If the inner product at that time offset is zero, (or close enough to zero for all practical purposes) the signal is orthogonal to itself at that time offset.



Does that sound complicated? In fact, you already know how to do this; the process we are describing is the convolution of the signal with itself! This operation is called the “auto-correlation.”

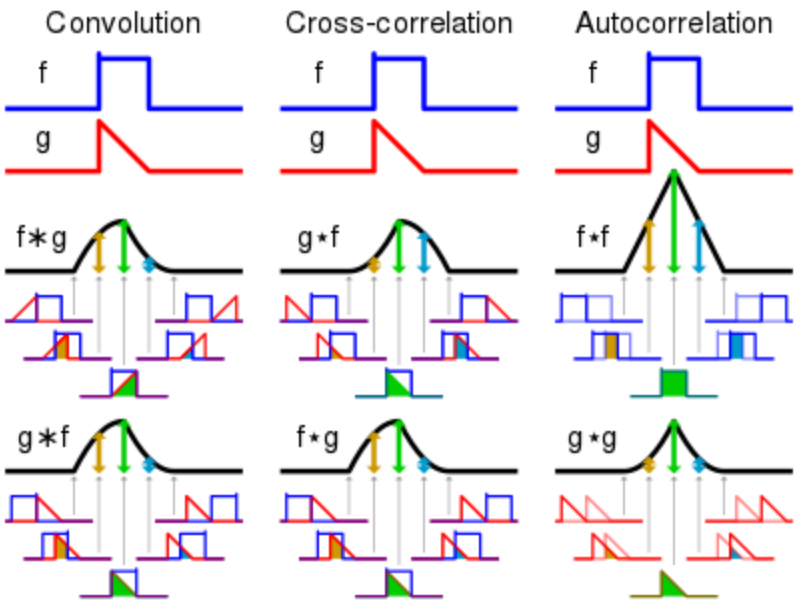


FIGURE 7: IMAGE SOURCE WIKIPEDIA COMMONS

To test if a signal is orthogonal to a time-delayed version of itself, take the convolution of the signal with a time-reversed version of itself (We do this because the convolution operation already time-reverses one of the signals. By doing so again we undo this process. If your signal is symmetric about the y-axis, this is unnecessary).

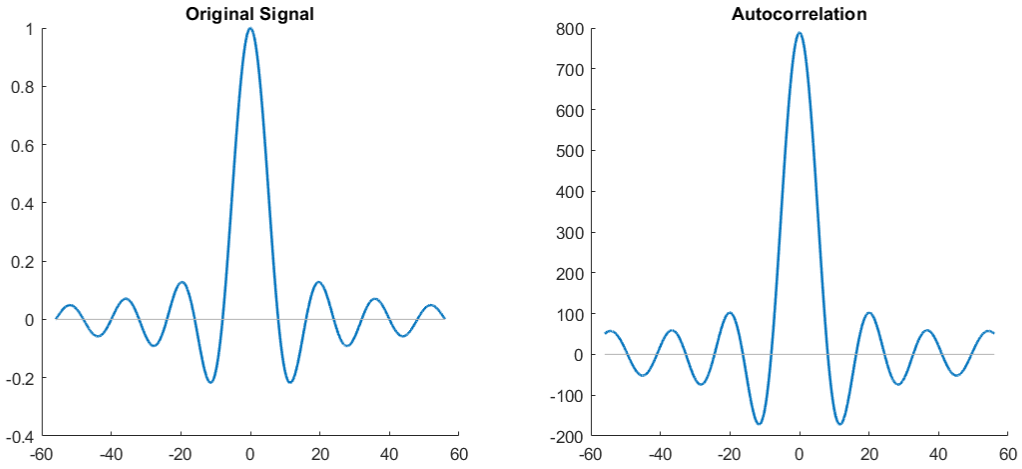


FIGURE 8: A SIGNAL AND ITS AUTOCORRELATION FUNCTION

At any value of t for which the autocorrelation function is zero, the signal is orthogonal to a copy of itself delayed by t. For instance, for the signal above, we can see it is orthogonal to versions of itself that have been delayed by integer multiples of 8 seconds.

Remember! Our motivation in this exploration of orthogonality is to find signals which will not interfere with each other, even if they overlap significantly. In this example, if we encode our symbols in this pulse shape and transmit it every 8 seconds (very slow!), our receiver will be able to recover the original symbols. If we use the wrong symbol frequency, say, one symbol every 6 seconds, this pulse

shape will not work; a pulse sent at t=0 will interfere with one sent at t=6, since the autocorrelation function at t=6N is nonzero for integer N.

The Nyquist Filtering Criteria

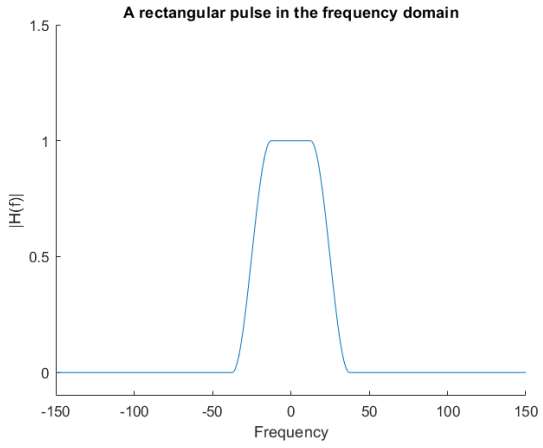
Examining our pulse shape in the time domain is not the only way to check if it has these useful orthogonality properties! We can verify many of the same conclusions by using the frequency domain representation of our pulse!

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:

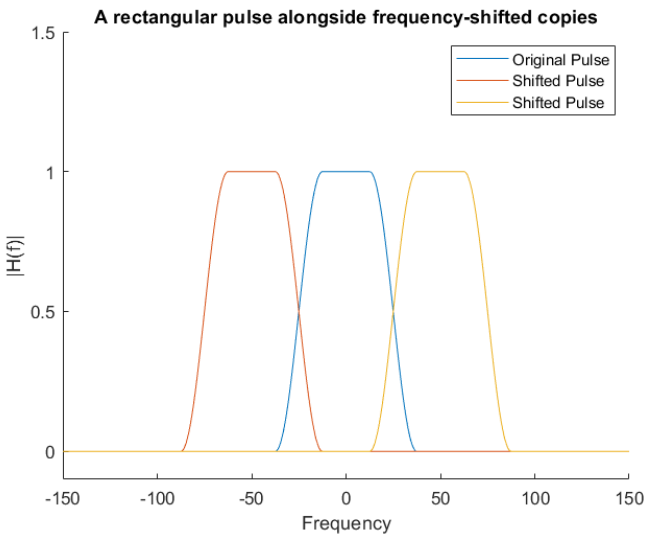
$$\frac{1}{T_s}\sum_{k=-\infty}^{+\infty} H\left(f-\frac{k}{T_s}\right)=1 \quad \forall f,$$

Essentially, this states that if we add the pulse shape to shifted versions of itself (where the shift is equal to the symbol frequency $1/T_s$) 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.

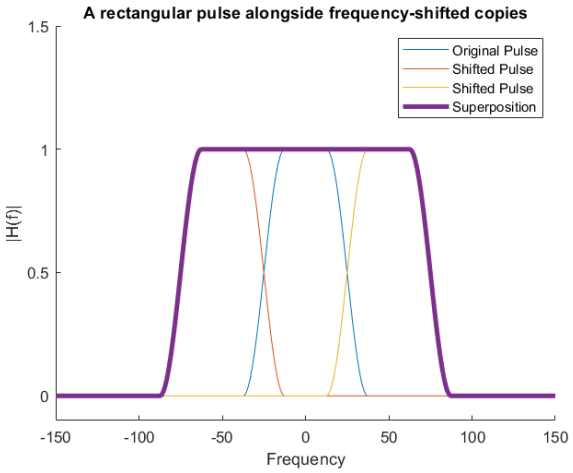
This might be hard to picture so here's a quick example. Consider a pulse shape with a Fourier transform that is mostly rectangular, with the edges rounded off. Let's say the symbol period is 1/50 seconds, so the symbol frequency is 50 Hz.



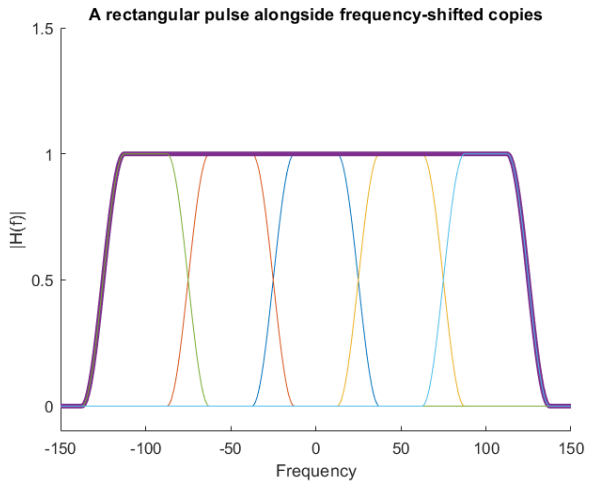
To test the Nyquist Filtering Criteria, we add versions of the pulse that have been shifted by 50 Hz in both directions. The actual criteria asks us to plot an infinite number of shifts, but we can plot just two - one shifted up 50 Hz, and one shifted down 50 Hz – and that will be enough to see the pattern.



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…



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



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. In this case study, we will be using discrete, finite time signals instead of continuous ones which can alter their Fourier transform. Use your best judgement on whether your pulse satisfies the Nyquist Filtering Criteria

Review

Wow, that was a lot! Let's quickly review everything before we hand the reigns to you. You've learned the following concepts:

- Orthogonality
  - Two signals are orthogonal if their inner product is zero.
  - If some set of signals are mutually orthogonal, then each linear combination of them is unique; the signals can be separated out again and the coefficients of the linear combination can be recovered.
- Correlation
  - The convolution of a signal with itself is called the autocorrelation. When performing the autocorrelation, we reverse one of the signals in time before convolving.
  - If the autocorrelation function is zero for some value of t_0, it means that the signals $f(t)$ and $f(t \pm t_o)$ are orthogonal.
- Digital Pulse Shaping
  - If the signals $f(t)$ and $f(t+NT_s)$ are orthogonal for symbol period T_s and any integer N, the signal $f(t)$ can be used to transmit symbols at a rate of $1/T_s$ without interference.

    - It is desirable to use a signal with a narrow bandwidth in the frequency domain and a short duration in the time domain. These properties are to some degree mutually exclusive; a trade-off between them must be found.
- The Nyquist Filtering Criteria
    - A signal *h(t)* with Fourier transform *H(f)* satisfies the Nyquist Stability Criteria for symbol period T_s if the sum of $|H(f+N/T_s)|$ for all integers N is a constant.
    - If a signal *h(t)* satisfies the Nyquist Stability Criteria, it can be used to transmit symbols at a rate of $1/T_s$ without interference.

Case Study

Ready to put all this to use? In this case study, you'll design your own pulse shape, explore its properties in both the time and frequency domain, and use it to transmit messages in a Pulse Amplitude Modulation scheme.

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 in the Pulse Shaping section, mainly:
    - Narrow frequency band
    - Short time duration
    - Orthogonal to versions of itself delayed by the symbol period
- Ensure you have both the time and frequency domain representations of your pulse.
- Determine the autocorrelation function of your pulse shape. 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 and comment on any interesting properties you notice. How might you use this property to send multiple signals simultaneously using the same pulse shape 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.