Filtering and Noise Effects on Digital Communications

Modules: Audio Oscillator, Sequence Generator, Baseband Channel Filters, Digital Utilities, Multiplier, Quadrature Utilities, Utilities, Adder, Tuneable LPF, 100-kHz Channel Filters, Noise Generator, Integrate & Dump, Wideband True RMS Meter

0 Pre-Laboratory Reading

Both filtering and noise can degrade the quality of digital communications. In this experiment, the effect of filtering and the effect of noise will be investigated separately. In practice, filtering and noise can both degrade the same system; but it is easier to investigate these two effects separately.

0.1 Filtering

You have already seen that filtering is essential in many communication applications. It is essential to have a filter at the output of a multiplier. If the difference-frequency term is required, then the sum-frequency term must be blocked. (In some applications, of course, those roles are reversed, and it is the sum-frequency term that is required.) In a superheterodyne receiver, the best performance can be obtained only if, before the multiplier, there is a filter that blocks the image band. Sampling works best with an anti-aliasing filter. These are just a few of the many communications-technology applications that require filtering.

In general, we want a filter to have a frequency response whose magnitude is constant within the passband and whose angle is a linear function of frequency within the passband. With such an ideal filter, the desired signal, which presumably lies within the passband, will experience no distortion in passing through the filter. Of course, the signal will be delayed, as always, by a filter; but a constant and relatively small delay is usually not a problem.

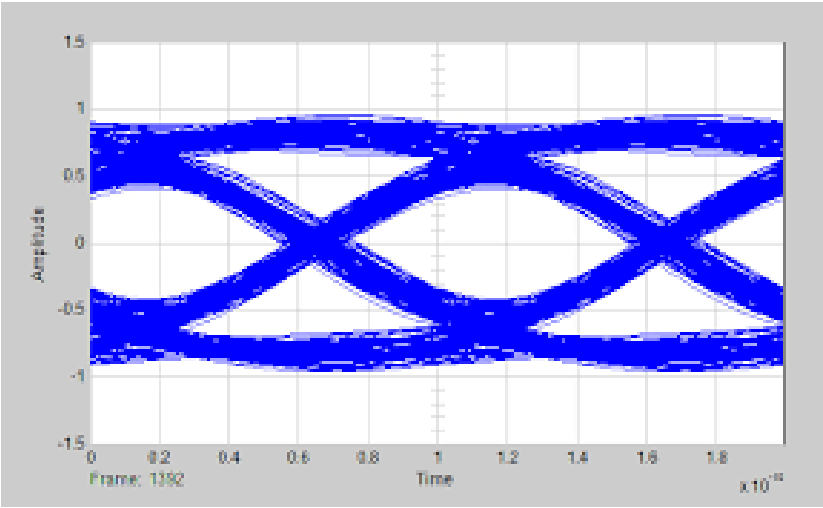
In practice, it is impossible to design a useful filter (one that blocks some part of the spectrum) that does not distort, at least to some extent, the desired signal lying within the passband. The frequency response of a practical filter will have a magnitude that is not perfectly constant and/or an angle this is somewhat nonlinear within the passband.

In addition, the channel that a signal passes through, such as a wireless channel between transmitter and receiver, will always have a limited bandwidth. Often we want to push the maximum possible bit rate (number of bits per second) through the channel. Under these circumstances, some part of the desired signal will typically lie outside the passband of the channel.

Imperfect filters and limited bandwidth will, therefore, cause degradation to a practical data communication system. This degradation will manifest itself as occasional (or perhaps even frequent) errors when the receiver tries to identify the arriving bits. Sometimes a positive bit will be mistaken for a negative bit and vice versa.

One method of investigating the effect of filtering on data communications is to create an example data stream, pass it through the channel (or filter), and observe both the input and output together on the oscilloscope. The effect of filtering is made visible in this way. We probably want to use continuous capture with this method, so that changes in the input and/or output signals are immediately reflected on the oscilloscope display. For a stable display, this implies that the input data stream must be periodic. A disadvantage of this method is that we must supply the (periodic) input data stream. This is fine for testing a system in the laboratory, prior to the deployment of the system into the field. But it is impractical to test a deployed, operating system (channel) in the field with this method. For one thing, we would have to take the system down in order to do our test. For another thing, the distance between channel input and output might be large, making it difficult or impossible to feed both channel input and output to the same oscilloscope.

An alternative method, which is immensely popular, is the eye diagram (also called the eye pattern). An eye diagram is created on an oscilloscope while using only one signal, the channel output. The eye diagram uses the persistence mode of an oscilloscope. The persistence mode is similar to the continuous-capture mode in the sense that the oscilloscope displays multiple traces of the signal obtained at different times. But instead of each new trace *replacing* the last trace, in persistence mode each new trace *augments* the old traces. In other words, in persistence mode, the old traces do not disappear, rather each new trace is added into the mix. It's like a painter adding new brush strokes to a canvas: the previous brush stroke does not disappear with the arrival of a new brush stroke. The figure below shows an example of a filtered data stream appearing on an oscilloscope display in persistence mode.



For the eye-diagram method to work, the oscilloscope must use, as an external trigger, the clock that defines the bit timing. This causes bit boundaries to line up, from one trace to the next. Of course, in one bit interval of one (filtered) trace, the bit might be positive; but the corresponding bit of the next trace might be negative.

When we view an eye diagram, we are looking for separation between a positive-bit trace and a negative-bit trace in any given (apparent) bit interval. (The bits will have arrived at different times, of course, but by an *apparent* bit interval we mean a horizontal section of the oscilloscope display that contains the bit interval for each of several different traces.) If the separation between positive-bit traces and negative-bit traces is present, we call this an "eye". The 0-V horizontal axis of the oscilloscope display will pass through a row of eyes. In the figure above, for example, there is one whole eye in the middle of the display, and there is a (more-than-half) partial eye to the left of this central eye and a (less-than-half) partial eye to the right of the central eye. If you imagine a vertical line passing through the center of an eye, this (imaginary) vertical line represents the center of the (apparent) bit interval.

A receiver typically judges whether a bit is positive or negative by testing if the (filtered) signal is positive or negative at the center of the bit interval. The eye diagram cannot be used by a receiver to identify individual bits. The receiver will make a decision about a received bit by examining only the portion of the received signal associated with that particular bit.

Eye diagrams are useful because they give us a good idea about whether a receiver will be able to correctly identify bits from the received (filtered) signal. In the figure above, the eyes are wide open, meaning that there is good separation, in an eye's vertical center-line, between the positive-bit traces and the negative-bit traces. We would expect no bit errors in this case.

The advantage of the eye-diagram method is that only one signal, the channel output signal, is required for the measurement. This means, for example, that the eye-diagram method can be applied to an operational channel without interrupting its work. We use the bit stream that is already passing through the channel. This bit stream need not be periodic. We only need to tap into the data stream. (In other words, we use a copy of the data stream that is being delivered by the channel to the end user.) Operational data channels often have built-in taps for just this purpose.

0.2 Noise

Noise is random. It typically has a mean of zero. (It is, at times, positive and, at other times, negative; but it has an average of 0 V.) When noise is present in a communication system, it can typically be modeled as an additive term. (In other words, the receiver sees the signal *plus* noise.) In the absence of noise, it is common for a receiver to identify a bit by testing whether the arriving signal is positive or negative at the center of the bit interval. However, in the presence of noise, a better approach is to do an integration that tends to *average-out* the effect of the noise.

1 Filtering and Eye Diagram

In this part of the experiment, you will use a filter to represent the filtering effect of a data communication channel. For this purpose, you will use the Baseband Channel Filters module, set for Filter 3.

Obtain a quick view of the magnitude of the transfer function for this filter. You can do this by connecting the Noise Generator module to the input of the Baseband Channel Filters module and then observing the spectrum of the filter output. Make sure to select Filter 3 on the front panel of the Baseband Channel Filters module.

Channel A: Baseband Channel Filters module (Filter 3) output, showing $|H(f)|$

Estimate the bandwidth of this low-pass filter.

On the PCB of the Sequence Generator module there is a pair of switches in a dual in-line package (DIP). These switches determine the sequences. Set both switches to the up position.

For this part of the experiment, you will use an analog clock, instead of a TTL clock, to control the timing on the Sequence Generator. The analog clock will be a sinusoid produced by the Audio Oscillator. Connect the analog output of the Audio Oscillator to the analog clock input of the Sequence Generator. Place a copy of this analog clock (the Audio Oscillator output) on the input of the Frequency Counter, so that you can monitor the clock frequency. Initially set the analog clock frequency to approximately 2.0 kHz.

You will use the sequence at the analog Y output port of the Sequence Generator; this is a bipolar signal. Connect this bipolar signal to the input of the chosen filter (Baseband Channel Filters module, Filter 3). Place a copy of this bipolar signal on Channel A. Connect the filter output to Channel B. In order to get a stable oscilloscope display, you will need to use the TTL sync output of the Sequence Generator module as an external trigger source. The analog clock should be set for approximately 2.0 kHz. Set the horizontal scale of the oscilloscope to show a relatively small number of bits.

Channel A: analog Y output of Sequence Generator (2.0-kHz clock)
Channel B: filter output

For the chosen filter and with a clock rate of 2 kHz (that is, a bit rate of 2000 bits/sec), it should be possible to recognize the bits on the filter output. There is distortion present on the filter output. Abrupt signal transitions have become gradual signal transitions. There is also delay, as there will be with any filter. By taking the delay into account, however, it should be possible for you to recognize the original bits in the filter output.

You should now increase the analog clock (the bit rate) from 2.0 kHz to approximately 3.0 kHz, then 4.0 kHz, then 5.0 kHz.

Channel A: analog Y output of Sequence Generator (3.0-kHz clock)
Channel B: filter output

Channel A: analog Y output of Sequence Generator (4.0-kHz clock)
Channel B: filter output

Channel A: analog Y output of Sequence Generator (5.0-kHz clock)
Channel B: filter output

By the time you get to a clock of 5 kHz, you should find that it is difficult to say, from the oscilloscope view currently available to you, that a receiver will be able to correctly identify the bits.

A better approach is the eye diagram. To get an eye diagram, make the following changes to your test arrangement. Turn off the Channel A display; you won't need it for an eye diagram. For the external trigger, use the analog clock that sets the bit rate (instead of the TTL synch output). Set the horizontal scale to 200 μs/div. Select the persistence mode.

In the persistence mode, a large set of oscilloscope traces are displayed simultaneously, not just the most recent trace. Sometimes in persistence mode, it will be necessary to reset the persistence display. For example, after changing a parameter in the test arrangement, such as the clock frequency, you will want to reset the persistence display. This is easily done by changing back to (regular) oscilloscope mode and then immediately back to persistence mode again.

Set the analog clock frequency to approximately 4.0 kHz (so that the bit rate is 4000 bits/sec). Reset the persistence display.

Channel B: filter output (4.0 kHz clock)

The 0-V horizontal axis should pass through a row of "eyes". These eyes should be wide open.

A receiver can make a decision about a bit by examining the center of the bit interval and noting the sign (positive or negative) of the waveform at that point. The persistence mode shows overlapping waveforms (traces). If all waveforms avoid the center of the eye, this suggests that the receiver will be able to tell the difference between a positive bit and a negative bit. For the present case, a 4.0-kHz bit rate passing through the chosen filter, the eyes should be open and a receiver should not be making any errors.

Change the analog clock frequency to approximately 5.0 kHz. Reset the persistence display.

Channel B: filter output (5.0 kHz clock)

Change the analog clock frequency to approximately 6.0 kHz. Reset the persistence display.

Channel B: filter output (6.0 kHz clock)

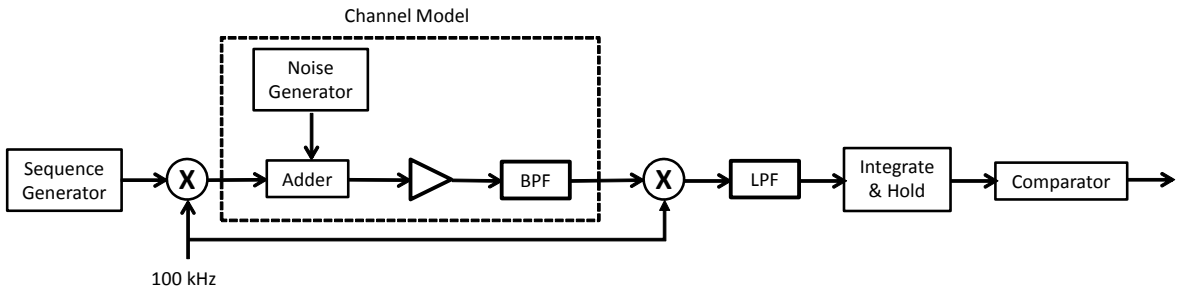
By the time you get to a clock of 6 kHz, you should find that the eyes are closed or nearly so. This filter cannot support a bit rate of 6000 bits/sec.

We often want to know the upper limit on bit rate that can be supported by a data communications channel. It is easier to discover this maximum bit rate using an eye diagram than by using an oscilloscope in the usual way (mode). The other advantage of an eye diagram is that only the channel output needs to be displayed on the oscilloscope. This means, for example, that the eye diagram can be used with whatever bits are coming through the channel. An eye diagram can be obtained while the channel is hard at work, delivering actual data in an operating system.

2 BPSK Detection in the Presence of Noise

You will assembly an end-to-end digital communications system. This system will include a data source, a BPSK modulator, a channel model, and a synchronous detector (using a stolen carrier). The channel model will include additive noise, so that you can see the effect of noise. The output of the synchronous detector will be noisy. The detector output will be sent to an Integrate & Hold device and a Comparator. The combination of the Integrate & Hold device and the Comparator should produce a "clean" (but delayed) copy of the original bit stream.

The paragraphs below explain how to build and adjust this system. The figure below illustrates what you are working toward:



The Sequence Generator will provide a stream of bipolar voltages (representing a stream of bits). Use a TTL clock with frequency $(100/48)$ kHz to clock the Sequence Generator. You may create this TTL clock by connecting a $(100/12)$-kHz TTL clock (labeled as the 8.3-kHz TTL clock on the Master Signals panel) to the input of a divide-by-4 function on the Digital Utilities module.

On the PCB of the Sequence Generator module there is a pair of switches in a dual in-line package (DIP). These switches determine the sequences. Set both switches to the up position. You will use the sequence at the analog Y output port; this is a bipolar signal. Connect this bipolar signal to Channel A. In order to get a stable oscilloscope display, you will need to use the TTL sync output of the Sequence Generator module as an external trigger source.

 Channel A: analog Y output of Sequence Generator

Create a PSK carrier by connecting the analog Y output of the Sequence Generator to one input of the Multiplier and a 100-kHz sinusoid (Master Signals) to the other input.

You will construct a channel model from the following modules: Adder, Noise Generator, Buffer Amplifier, and 100-kHz Channel Filters. The output of the modulator (that is, the Multiplier) will go to one input of the Adder. The other input will come from the Noise Generator, set for a noise level of $+22$ dB. The Adder is actually a weighted summer with an inversion. The Adder will be followed by a Buffer Amplifier, having a negative gain, so that the net gain through the channel model will be positive. Connect the output of the Buffer Amplifier to the 100-kHz Channel Filters module, set for filter 3, which is a band-pass filter having a center frequency of 100 kHz (despite what the TIMS manual page says about this module).

In practice, transmitted signals pick up noise as they pass to the receiver, and this noise typically has zero mean and a flat (white) spectrum. The Noise Generator produces zero-mean noise with an approximately flat spectrum, up to several hundred kilohertz.

You will adjust the signal and noise levels using the gain of the Buffer Amplifier and the two weighting factors (gains) of the Adder. Connect the output of the channel model (the output of the band-pass filter) to the RMS Meter. For this experiment, we want the signal and the noise to each contribute 1.0 V rms to the channel model output. The next two paragraphs explain how to accomplish this.

First, you will set the noise level. Temporarily disconnect the BPSK signal from the Adder, so that only the noise is present in the channel model. Set the Adder gain knob associated with the noise to a mid-range position where the line on that knob is vertical. Then adjust the Buffer Amplifier gain so that the RMS Meter at the channel model output indicates 1.0 V rms. It is important that you make no further changes to either the Buffer Amplifier gain or to the Adder noise-path gain, so that the noise level at the channel model output remains 1.0 V rms.

Second, reconnect the BPSK signal to the Adder and temporarily disconnect the noise from the Adder. Now only the signal should be present in the channel model. By adjusting the Adder's signal-path gain, set the output of the channel model equal to 1.0 V rms. After that is accomplished, reconnect the noise to the Adder. You have now set the signal and noise contributions to the channel model output.

We want to observe the spectrum of the channel model output, which will be the input to the receiver. First, temporarily disconnect the noise from the Adder, so that only signal is present in the system.


Channel A: channel model output (BPSK signal only)

Second, reconnect the noise to the Adder.


Channel A: channel model output (BPSK signal and noise)

You should observe that the noise has a noticeable effect on the spectrum.

You will use synchronous detection here, with a stolen 100-kHz sinusoid as the local oscillator and a multiplier from the Quadrature Utilities module. The output of this multiplier will go to a Tuneable LPF, set for a bandwidth of approximately 26 kHz. (The Tuneable LPF's clock output has a frequency equal to 100 times the bandwidth.)

Connect the analog Y output of the Sequence Generator to Channel A. Connect the synchronous detector output (the output of the Tuneable LPF) to Channel B. Use the TTL sync output of the Sequence Generator module as an external trigger source. The Channel A signal should be stable. However, there is noise as well as signal on Channel B. Since noise is not periodic, it is not possible to make the Channel B waveform appear stable. Therefore, you will want to stop signal capture in order to study the oscilloscope display.


Channel A: analog Y output of Sequence Generator
Channel B: synchronous detector output

Let us suppose that you are tasked with making decisions about the bits as viewed at the output of the synchronous detector. We assume here that you don't know the original bits. Imagine that you can't see the Channel A signal; instead, you can only see the Channel B waveform. Yet you must guess, based on the Channel B waveform, whether the bit corresponding to the each bit interval is positive or a negative. (It is assumed here that you know where the bit boundaries lie.) A simple way to do this is to select one instant in time for each bit interval and guess "positive" if the Channel B waveform is positive at that designated instant in time and guess "negative" otherwise. We might select the exact center of the bit interval as the instant in time at which to

guess the bit. In the absence of noise, this is a reasonable solution. However, as you can see from the oscilloscope display, this is not a good solution when significant noise is present. You will probably be able to see some cases in which the above decision process leads to an incorrect decision on a bit. If you can't see this, you should try changing the time scale so that more bits are displayed.

A better solution, given the presence of noise, is to send the output of the synchronous detector to a combination of an Integrate & Hold device followed by a Comparator. There are two Integrate & Hold devices on the Integrate & Dump module. For the input/output port pair I&D 1, you will set a dial on the PCB to the Integrate & Hold function. (You could use I&D 2 instead.) You will supply a $(100/48)$-kHz TTL clock to the TTL clock input of the Integrate & Dump module. The period of this clock equals, in this case, the bit period. With this arrangement, there is an integration of the input waveform over each individual bit period. The result of the integration is negated and then held for one bit period. The output signal looks like a cleaner version of the input signal except that there is also a negation and a delay of one bit period.

The Comparator is on the Utilities module. Its Ref (Reference) input should be connected to the ground port on the Variable DC panel. The output from the Integrate & Hold device should be connected to the analog input of the Comparator, and the analog output of the Comparator should be used as the receiver output. With the Comparator's Ref input at ground, a positive voltage on the input produces a negative voltage on the analog output and a negative voltage on the input produces a positive voltage on the TTL output.

The cascade of the Integrate & Hold device with the Comparator device has the effect of a decision-making function. Although there is an inherent negation in each of these devices, the cascade has positive net gain, as the two negations (one in each device) cancel each other. There will be a delay of one bit period in the cascade output, which is a result of the integration.

Channel A: analog Y output of Sequence Generator
Channel B: receiver output (analog output of Comparator)

You should find that the combination of the Integrate & Hold device followed by the Comparator produces a reliable decision on the bits, despite the presence of noise. As mentioned above, there is a one-bit delay associated with the Integrate & Hold.