EE 422G - Signals and Systems Laboratory

Lab 6 Digital Transmission

Kevin D. Donohue
Department of Electrical and Computer Engineering
University of Kentucky
Lexington, KY 40506
February 19, 2013

Objectives:

- Understand how signal information can be encoded for transmission through base-band analog channels.
- Observe the impact of typical sources of corruption that create errors in the communication process, namely channel noise and limited bandwidth.
- Perform spectral estimation to analyze the relationship between encoded signal bandwidth requirements, channel noise, and bandwidth.

1. Background:

This lab considers encoding a digital data stream into analog signals for transmission through a base-band analog channel. Various line coding methods for digital base-band modulation will be implemented and the spectral properties of the codes will be examined. Simulation studies will be used to establish a relationship between bit error rates and signal corruption.

The classical model for corruption in communication channel is a band-limited filter (low-pass or band-pass filter) with additive Gaussian noise as shown in Fig. 1.

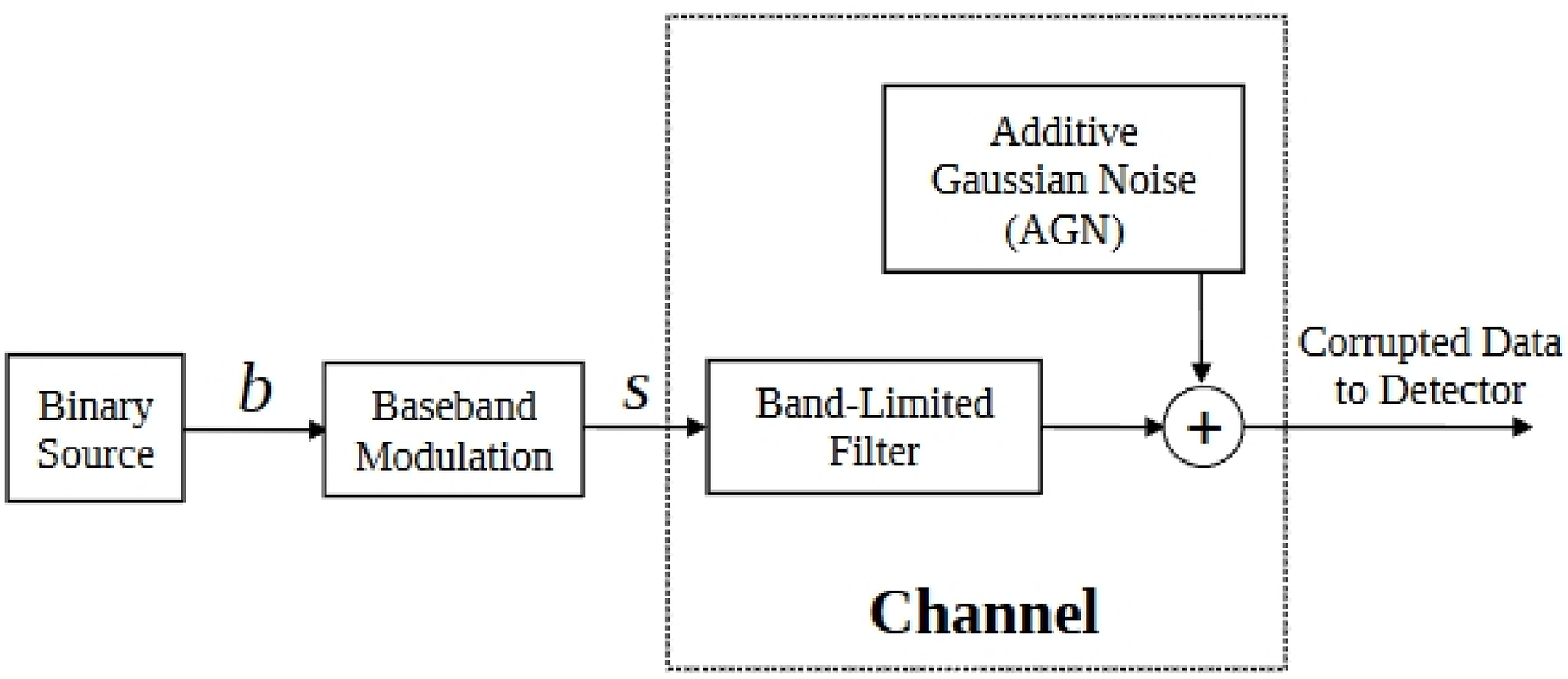

Figure 1. Block diagram of communication system showing the key components that interfere with the information transfer, such as the band-limited filtering and the additive white Gaussian noise.

Figure 1 illustrates the main components of a communication source and transmission. The information to be transmitted is converted into a binary sequence (i.e. ASCII codes for text or sequences of quantized signal amplitudes from a sampled analog waveform). The binary data source emits a series of 1's and 0's (bits) representing the communication system *source*. The *baseband modulation* operation codes each bit into a continuous waveform for sending the source sequence over a physical channel. The channel is a medium through which the signal propagates. Typical signals are formed by changes in voltages and currents over a wire, light energy over an optical fiber, or electromagnetic energy through the atmosphere. No medium is totally free of noise (random perturbations on transmitted signal), therefore white or colored Gaussian noise is added to the transmitted signal to simulate this corruption. Noise is an irreversible corruption (i.e. permanent loss of information). While the signals can be filtered to exploit signal redundancies and reduce the impact of noise, there will always be a level of uncertainty in the received signal. Parameters extracted from signals denoting 1s and 0s must be sufficiently separated (i.e. amplitude, frequency, phase …) to limit the impact of the ambiguities introduced by the system noise.

In addition to noise, channels have a limited bandwidth that can potentially distort the signals by reducing the frequency content of the signal non-uniformly over its spectrum. So if the coded waveforms are not bandlimited to a value less than channel bandwidth, the waveform will be distorted. Distortion is a deterministic change in the signal and can be reversed in some cases. If the distortion can be modeled as a linear filter, it can be reversed (the operation of undoing this distortion is referred to as deconvolution). In most cases of nonlinear distortion, such as clipping or quantization, the distortion cannot be reversed.

For the baseband channel, a low-pass filter is used to simulate its distortion. *Baseband* and *low-pass* essentially means the same thing when describing a signal. If signals are modulated with a high frequency oscillator (baseband spectrum shifted up on the frequency axis), such that they contained no significant DC energy, then the signals are considered passband (not baseband). This lab considers only baseband signals and channels.

In order to send information over the channel, the information must be encoded into a sequence of binary digits and these digits are converted into analog signals for transmission using a variety of signaling formats called *line codes*. To help examine their spectral properties, a Matlab function, *modulb*(), was written to create various line codes from bit sequences. The function syntax is:

```
>>  [y,t] = modulb(binary_sequence, Fd, Fs, line_code_name);
```

where *binary_sequence* is a vector of 1's and 0's denoting the source binary sequence, *Fd* corresponds to the binary data rate in bits per second (b/s), *line_code_name* is a special string indicating the particular line code to be used (see help file), and *Fs* is the sampling frequency of the line code waveform used by the simulation. Note that sampling rate *Fs* is used to simulate the analog signal, so this sampling rate needs to be much higher than

the bit rate (***at least*** by a factor of 10, 100 is preferred if it does not excessively slow down the system). The *modulb* function outputs the waveform as vector *y* and corresponding time axis *t*. The command supports the following codes: *'unipolar_nrz'*, *'bipolar_nrz'*, *'bipolar_rz'*, *'ami'*, *'manchester'*, *'miller'*, *'unipolar_nyquist'*, and *'bipolar_nyquist'*.

The type of line coding is selected to meet various system criteria such as power requirements, bit timings (additional transitions of the line code signals within the bit interval can help in timing recovery), bandwidth efficiency (excessive transitions may require more bandwidth than necessary), low frequency content (some channels block low frequency), error detection, and complexity. Figure 2 shows analog waveforms for various line coding examples.

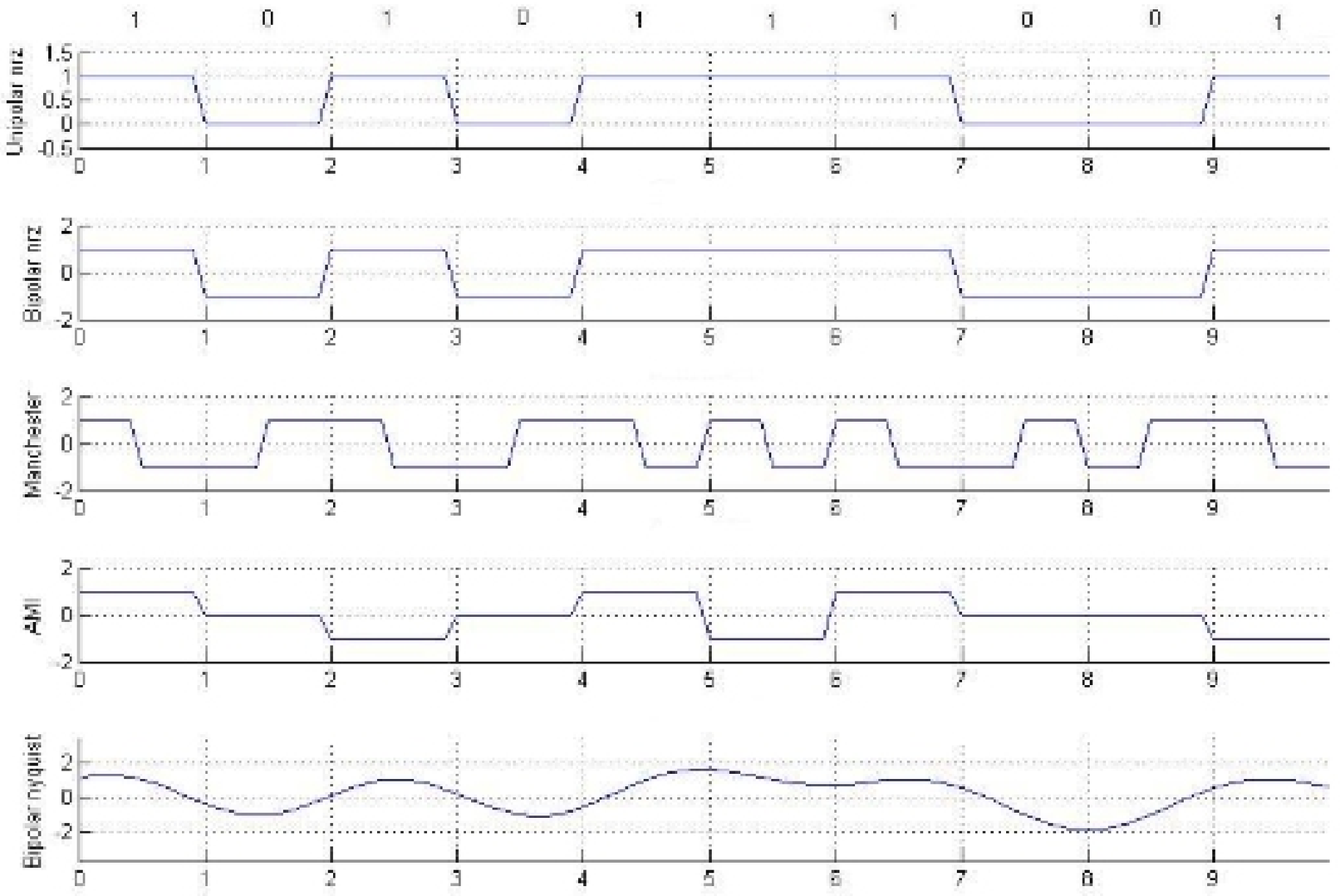

Figure 2. Line code waveform examples. Original binary sequence listed across the top is at a bit rate of 1, and applies to all codes.

2. Pre-Laboratory Assignment

1. Use the *modulb*() function to plot waveforms representing the binary sequence $seq=[0,1,0,0,1,1,0,0,0,1,1,1,0,0]$ using the following line codes at a bit rate of $R_d=1$ kb/s (choose a reasonable sampling rate for potting the waveforms):

   a) unipolar NRZ (on-off signaling, NRZ= non-return to zero)
   b) bipolar NRZ (binary antipodal signaling)
   c) bipolar RZ (binary antipodal signaling RZ = return to zero)