LAB 5: Line Coding

This experiment investigates how a digital data stream can be encoded into a sequence of pulses for transmission through a baseband channel.

Learning Objectives

1. Examine various line coding methods used for digital baseband modulation in data communication applications using Matlab.
2. Recognize the advantages and drawbacks of each encoding technique.
3. Inspect spectral properties of those lines codes, i.e. their power spectral densities.

Background

Line coding denotes to the method of representing the bit stream (1's and 0's) in the form of voltage or current variations optimally adjusted for the specific properties of the used physical channel. So, the digital information must be converted into a physical signal and this physical signal is called a line code. Generally, line coders use the terms ***mark*** to mean binary one and ***space*** to mean binary zero.

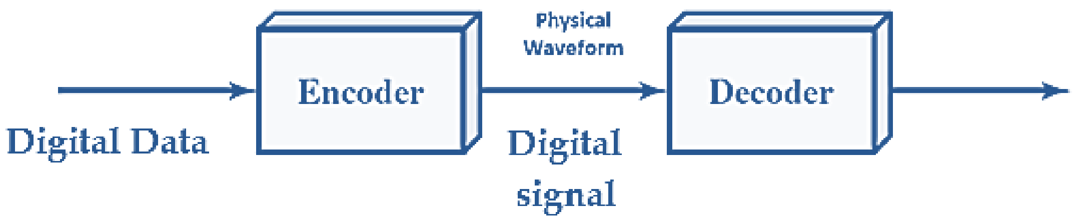


Figure 1: Encoding into digital signal

Line codes basic

Each line code is described by a **symbol mapping function** a_k and a **pulse shape** $p(t)$. So, the output of the line encoder is a waveform could be expressed as:

$$x(t) = \sum_{k=-\infty}^{+\infty} a_k p(t - kT_b)$$

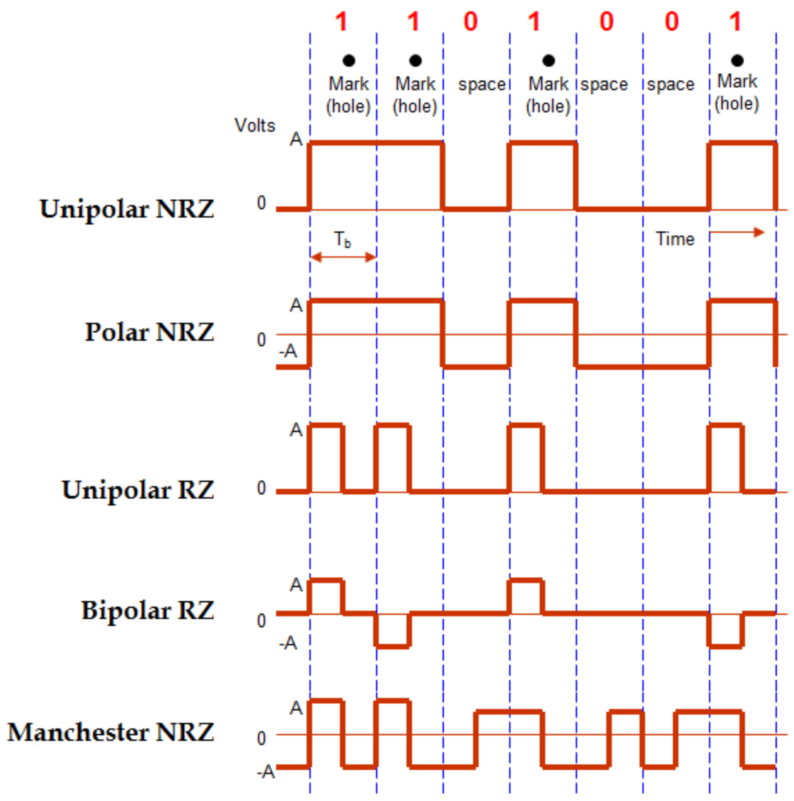
Where T_b is the bit period.

Categories of symbol mapping function a_k:

- **Unipolar**: all a_k elements have the same algebraic sign, that is, all positive or negative.
- **Polar**: one logic state is represented by a positive voltage level and the other by a negative voltage level.
- **Bipolar** (also known as alternate mark inversion AMI or pseudo-ternary): binary 0 is represented by no line signal, and a binary 1 is represented by a positive or negative pulse. The binary 1 pulses must alternate in polarity.

Categories of Pulse shape $p(t)$:

- **NRZ** (Non return-to-zero): the voltage level is constant during a bit interval; there is no transition (no return to a zero voltage level).
- **RZ** (Return to Zero): the voltage level is constant during first half of bit interval (i.e. positive or negative level) and its return to a zero voltage level in the second half of bit interval.
- **Manchester** (Biphase): there is a transition at the middle of each bit period. A low-to-high transition represents the logic 1, and a high-to-low transition represents the logic 0.



Signal element: a voltage pulse of constant amplitude. That part of a signal that occupies the shortest interval of a signaling code.

Evaluation of line codes

The selection of a proper line code can help in so many ways. The following metrics are consider for evaluating or comparing the various line coding techniques.

- **Self-synchronization**: We talk about the need to determine the beginning and end of each bit where a separate clock lead to synchronize the transmitter and receiver is not provided.
  - The ability to **recover timing** from the signal itself (i.e. that benefits in clock recovery at the receiver).
  - Long series of ones and zeros could cause a problem.
- **Error probability**:
  - The receiver needs to be able to distinguish the waveform associated with a mark from the waveform associated with a space, even if there is a considerable amount of noise and distortion in the channel.
- **Signal spectrum** is suitable for the channel.
  - In some cases **direct-current (dc) component** should be avoided if the channel has a DC blocking capacitance. With no dc component, ac coupling via transformer is possible; this provides excellent electrical isolation, reducing interference.
  - The transmission **bandwidth** should be minimized. A lack of high-frequency components means that less bandwidth is required for transmission.
  - A good signal design should concentrate the transmitted power in the middle of the transmission bandwidth. Because in practice, the transmission characteristics of the channel are worse near the band edges.

Practical Part

Task 1: UNRZ Line Code

Open Matlab, and create new script. Save this script as 'LineCodes.m'. and write the your commands inside it.

1. Use randsrc function to generate N data bits, and now let's N equal to 10.
2. Define the time base knowing that the data bit rate is 1Kbps and the sampling frequency is 10Ksps.
3. Now, open a new script and write: `function x = unrz(bits, bitrate,Fs)` here you define the function unrz that have three inputs (data bits, bit rate, and sampling frequency), while it has one output x which is the UNRZ encoded bits. Save the file with a name matched with the name of the function (i.e. unrz.m). to form the this line code follow these steps:
   - 3-1. Get number of bits N.
   - 3-2. Get number of samples per bit.
   - 3-3. Define the symbol mapping vector of UNRZ.
   - 3-4. Define the pulse shape vector of UNRZ.
   - 3-5. Set output initial value as an empty vector.
   - 3-6. Make a loop over all bits, then add to output - by concatenation - the vector that represents the line coded bit.
   - 3-7. Save your function.
4. Return back to your main file and call your function. Then display the output in a figure and check your resulted UNRZ coded data. (to make your figure more attractive, add a title that contains the Line code name and the original bits vector).
5. Calculate the signal mean value. What do you deduce?
6. In your opinion, Is it easy to recover symbol timing when a long string of 0s or 1s?

Task 2: other line codes

Copy the contains of unrz.m file to a new scripts and modify its names and its commands to form the polar NRZ, unipolar RZ, Bipolar RZ, and finally Manchester line coding. and then for each one repeat the questions 4, 5, and 6 in Task 1.

Task 3: Line codes spectrum

We want to compare the last performed line codes in the frequency domain. You have seen in the last lab that most of the power in a signal is concentrated in some finite band, and the effective bandwidth consists of that portion of the spectrum that contains most of the power. To make this concept precise, we need to define the power spectral density (PSD). In principle, the PSD refers to the power content of a signal as a function of frequency, so that it shows how much power is present over various frequency bands.

To estimate the power spectral density (PSD) of a signal 's' in Matlab you can use the following commands:

```
[pxx,f] = pwelch(s,1024,512,1024,Fs);
plot(f,10*log10(pxx),'r','Linewidth',2); hold on;
xlabel('Hz'); ylabel('dB');
```

1. Now, display the PSD of all line codes line codes studied in tasks 1 and 2 on the same figure.
2. By observing the PSD plot, identify the location on the frequency axis of the first spectral peaks $f_{p1}$and the first spectral nulls f_{n1}. Please note that we are only interested in spectral nulls for $f > 0$, while nulls at $f = 0$ means that there is no DC in the signal. The location of the peaks and nulls should be entered in the table below.
3. If the minimum required channel bandwidth for a given line code is determined by the location of the first spectral null, determine the bandwidth W for each line code.

Line Code	f_{p1} [Hz]	f_{n1} [Hz]	W [Hz]	DC ?
UNRZ				
URZ				
PRZ				
Manchester				
RZ-AMI				