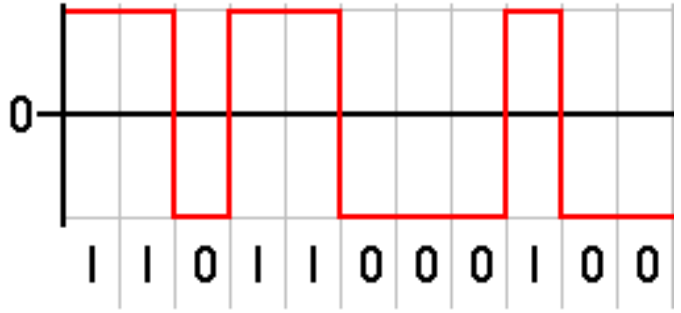
Lab No. 8
Data Communication and Networks

Line Coding

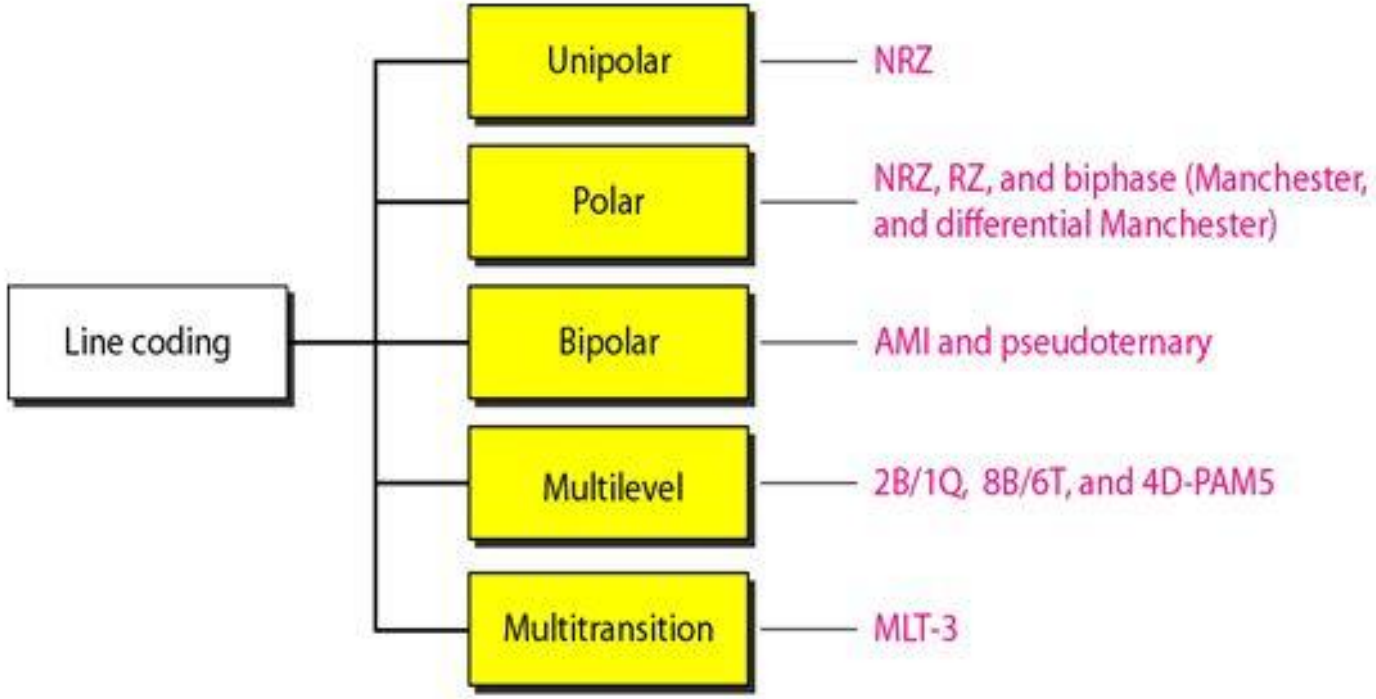
Line Coding

- In telecommunication, a line code is a code chosen for use within a communications system for transmitting a digital signal down a line.
- Line coding is often used for digital data transport.



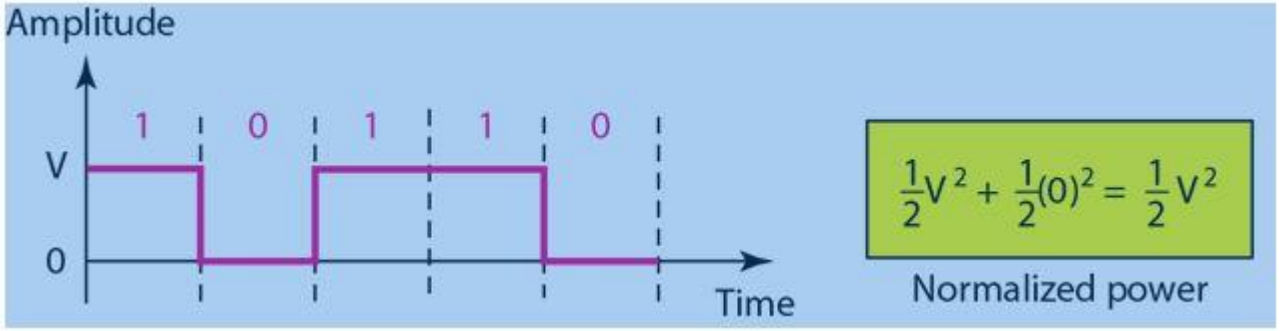
- Line coding consists of representing the digital signal to be transported, by a waveform that is optimally tuned for the specific properties of the physical channel (and of the receiving equipment).

Different Line-Coding Techniques



Unipolar Scheme:

- In a unipolar scheme, all the signal levels are on one side of the time axis, either above or below.
- NRZ (Non-Return-to-Zero): Traditionally, a unipolar scheme was designed as a non-return-to- zero (NRZ) scheme in which the positive voltage defines bit 1 and the zero voltage defines bit O. It is called NRZ because the signal does not return to zero at the middle of the bit. The following figure shows a unipolar NRZ scheme.



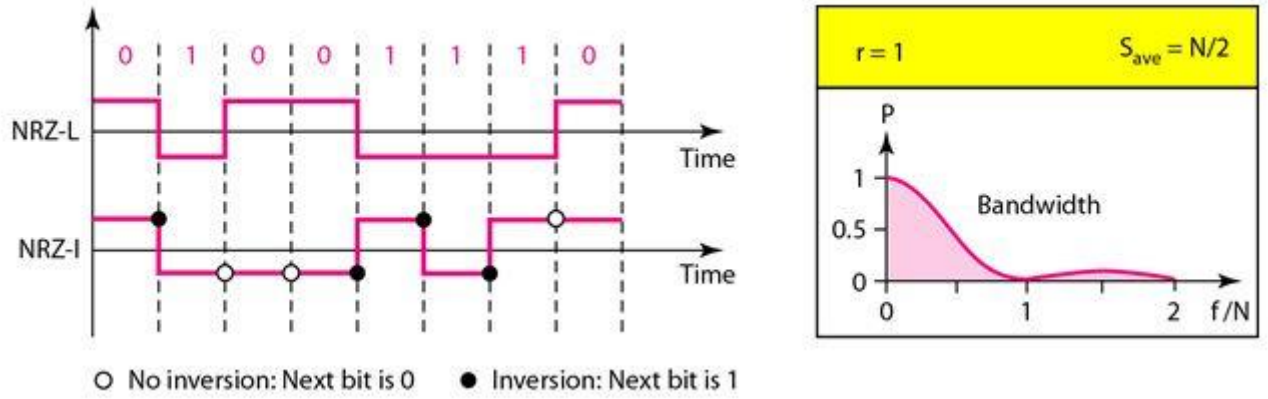
Polar Schemes

- In polar schemes, the voltages are on the both sides of the time axis. For example, the voltage level for 0 can be positive and the voltage level for I can be negative.

Non-Return-to-Zero (NRZ):

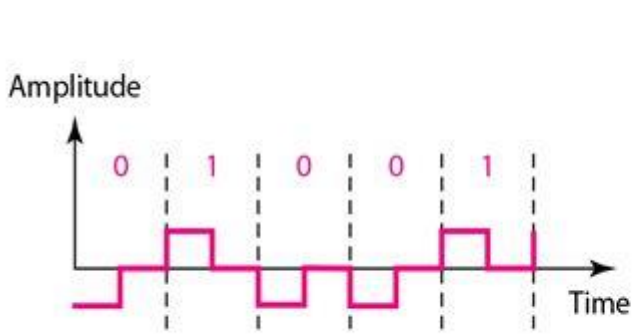
- In polar NRZ encoding, we use two levels of voltage amplitude. We can have two versions of polar NRZ: NRZ-Land NRZ-I, as shown in the following Figure. The figure also shows the value of r, the average baud rate, and the bandwidth.

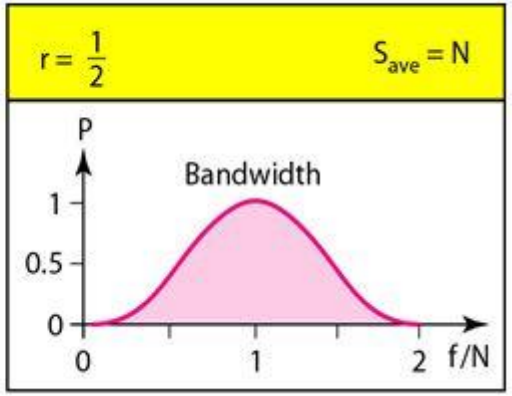
- In the first variation, NRZ-L (NRZ-Level), the level of the voltage determines the value of the bit. In the second variation, NRZ-I (NRZ-Invert), the change or lack of change in the level of the voltage determines the value of the bit. If there is no change, the bit is 0; if there is a change, the bit is 1.



Return to Zero (RZ):

- The main problem with NRZ encoding occurs when the sender and receiver clocks are not synchronized. The receiver does not know when one bit has ended and the next bit is starting. One solution is the return-to-zero (RZ) scheme, which uses three values: positive, negative, and zero. In RZ, the signal changes not between bits but during the bit. In the following figure, we see that the signal goes to 0 in the middle of each bit. It remains there until the beginning of the next bit.

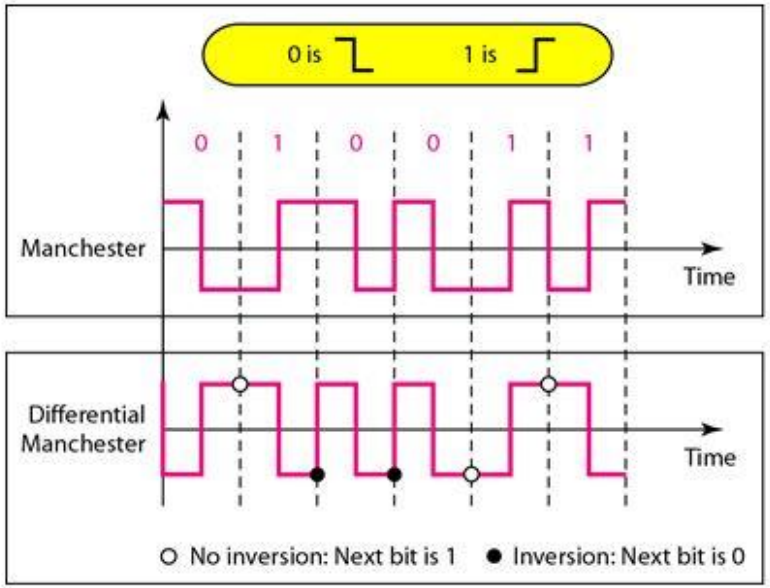


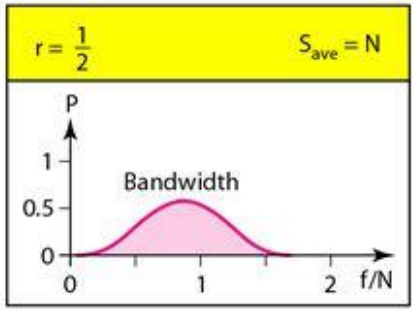


Biphase Manchester and Differential Manchester:

- The idea of RZ (transition at the middle of the bit) and the idea of NRZ-L are combined into the Manchester scheme.
- In Manchester encoding, the duration of the bit is divided into two halves. The voltage remains at one level during the first half and moves to the other level in the second half. The transition at the middle of the bit provides synchronization.
- Differential Manchester, on the other hand, combines the ideas of RZ and NRZ-I. There is always a transition at the middle of the bit, but the bit values are determined at the beginning of the bit. If the next bit is 0, there is a transition; if the next bit is 1, there is none. The following figure shows both Manchester and differential Manchester encoding.

- The Manchester scheme overcomes several problems associated with NRZ-L, and differential Manchester overcomes several problems associated with NRZ-I. First, there is no baseline wandering. There is no DC component because each bit has a positive and negative voltage contribution. The only drawback is the signal rate. The signal rate for Manchester and differential Manchester is double that for NRZ. The reason is that there is always one transition at the middle of the bit and maybe one transition at the end of each bit.





Line Coding in MATLAB

Function For Biphase Manchester

```
function [t,x] = manchester(bits, bitrate)
% MANCHESTER Encode bit string using Manchester code.
%   [T, X] = MANCHESTER(BITS, BITRATE) encodes BITS array using Manchester
%   code with given BITRATE. Outputs are time T and encoded signal
%   values X.
 T = length(bits)/bitrate; % full time of bit sequence
n = 200;
N = n*length(bits);
dt = T/N;
t = 0:dt:T;
x = zeros(1,length(t)); % output signal
```

Function For Biphase Manchester (Cont…)

```
for i = 0:length(bits)-1
 if bits(i+1) == 1
   x(i*n+1:(i+0.5)*n) = 1;
   x((i+0.5)*n+1:(i+1)*n) = -1;
 else
   x(i*n+1:(i+0.5)*n) = -1;
   x((i+0.5)*n+1:(i+1)*n) = 1;
 end
end
```

Function For Polar RZ

```
function [t,x] = prz(bits, bitrate)
% PRZ Encode bit string using polar RZ code.
%   [T, X] = PRZ(BITS, BITRATE) encodes BITS array using polar RZ
%   code with given BITRATE. Outputs are time T and encoded signal
%   values X.

T = length(bits)/bitrate; % full time of bit sequence
n = 200;
N = n*length(bits);
dt = T/N;
t = 0:dt:T;
x = zeros(1,length(t)); % output signal
```

Function For Polar RZ (cont…)

```
for i = 0:length(bits)-1
  if bits(i+1) == 1
    x(i*n+1:(i+0.5)*n) = 1;
    x((i+0.5)*n+1:(i+1)*n) = 0;
  else
    x(i*n+1:(i+0.5)*n) = -1;
    x((i+0.5)*n+1:(i+1)*n) = 0;
  end
end
```

Function For Unipolar NRZ

```
function [t,x] = unrz(bits, bitrate)
% UNRZ Encode bit string using unipolar NRZ code.
%   [T, X] = UNRZ(BITS, BITRATE) encodes BITS array using unipolar NRZ
%   code with given BITRATE. Outputs are time T and encoded signal
%   values X.

T = length(bits)/bitrate; % full time of bit sequence
n = 200;
N = n*length(bits);
dt = T/N;
t = 0:dt:T;
x = zeros(1,length(t)); % output signal
```

Function For Unipolar NRZ (Cont…)

```
for i = 0:length(bits)-1
  if bits(i+1) == 1
    x(i*n+1:(i+1)*n) = 1;
  else
    x(i*n+1:(i+1)*n) = 0;
  end
end
```

Function For Unipolar RZ

```
function [t,x] = urz(bits, bitrate)
% URZ Encode bit string using unipolar RZ code.
%   [T, X] = URZ(BITS, BITRATE) encodes BITS array using unipolar RZ
%   code with given BITRATE. Outputs are time T and encoded signal
%   values X.
T = length(bits)/bitrate; % full time of bit sequence
n = 200;
N = n*length(bits);
dt = T/N;
t = 0:dt:T;
x = zeros(1,length(t)); % output signal
```

Function For Unipolar RZ (Cont…)

```
for i = 0:length(bits)-1
  if bits(i+1) == 1
    x(i*n+1:(i+0.5)*n) = 1;
    x((i+0.5)*n+1:(i+1)*n) = 0;
  else
    x(i*n+1:(i+1)*n) = 0;
  end
end
```

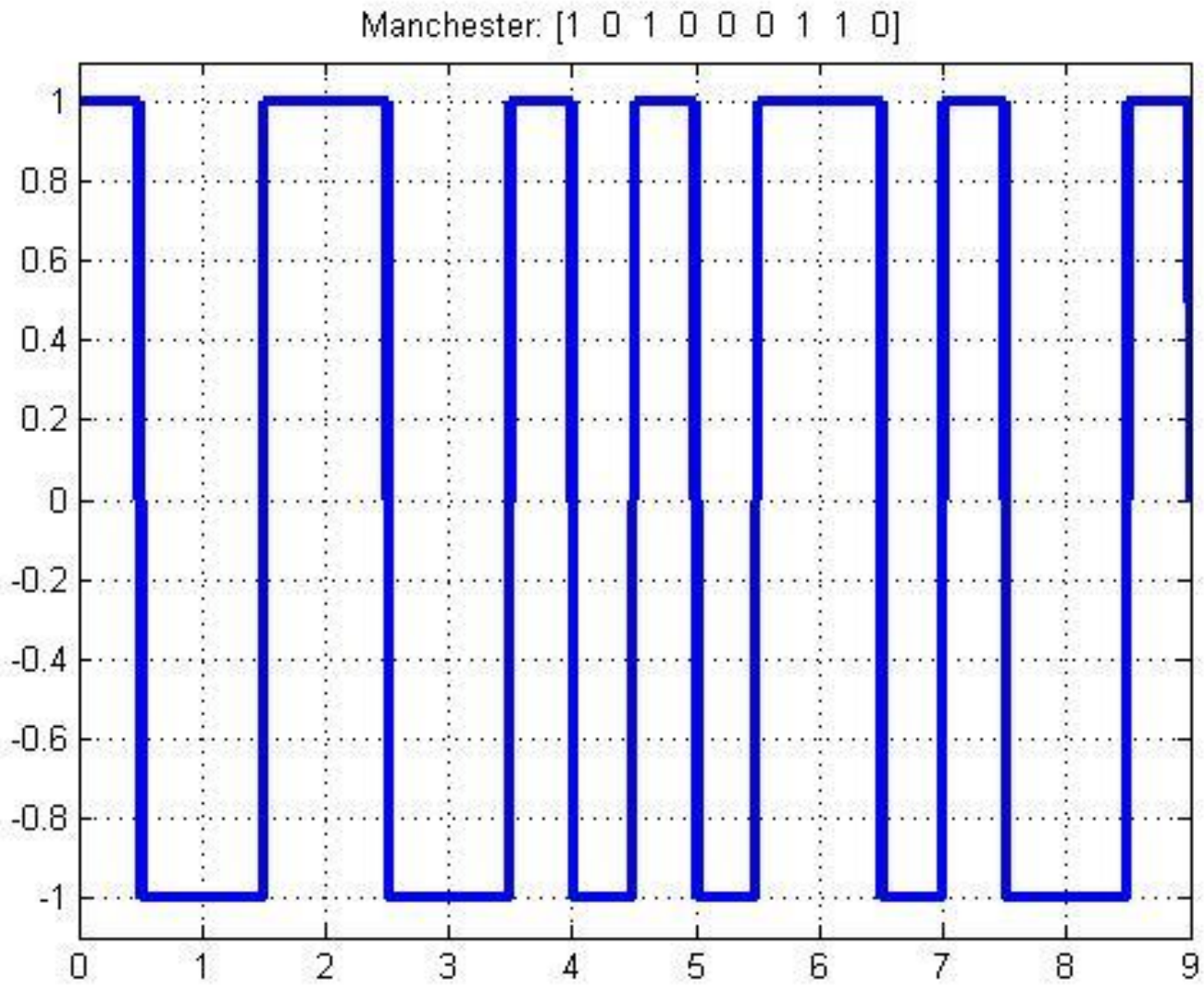
Calling the functions

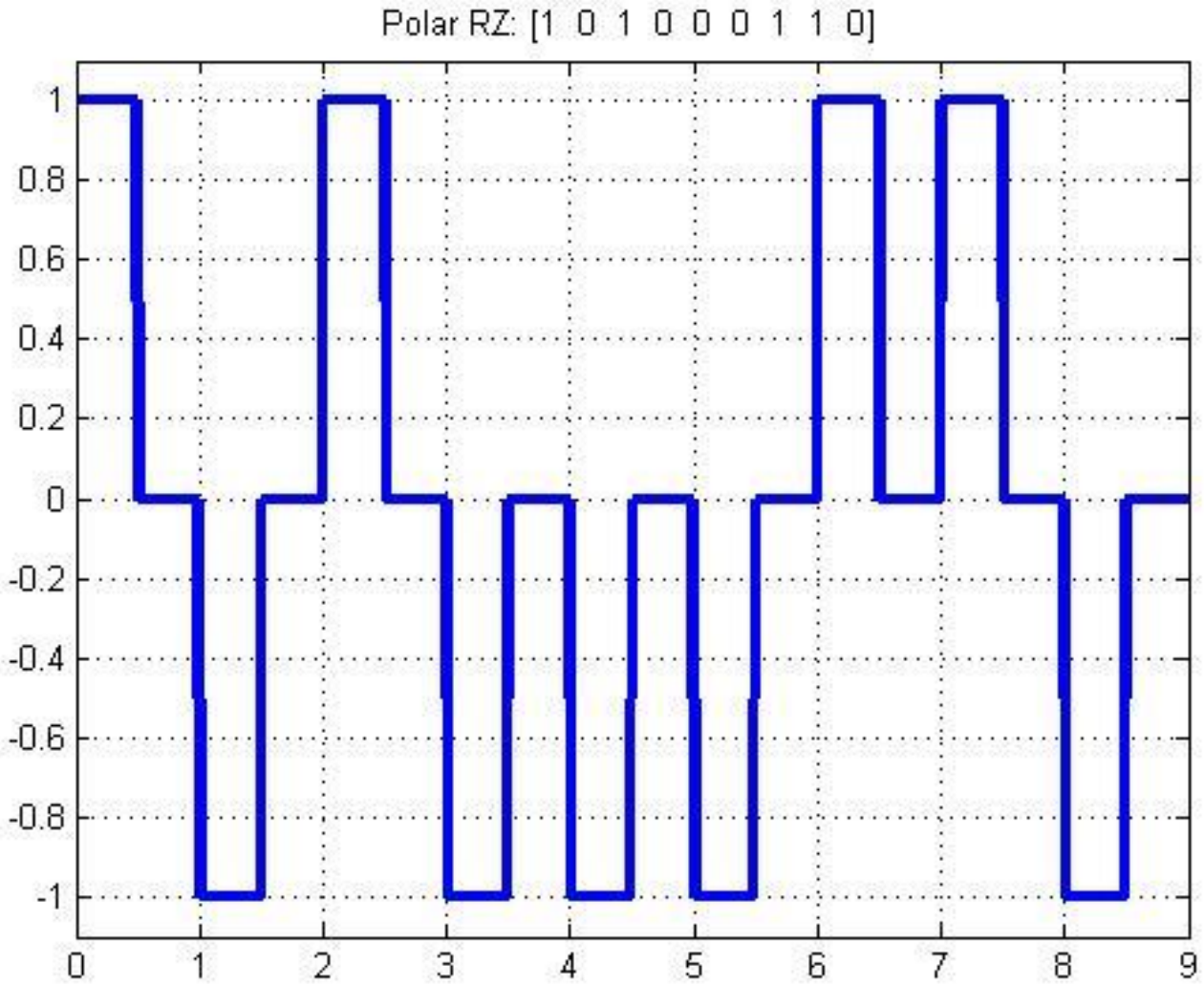
```
% Demo of using different line codings
bits = [1 0 1 0 0 0 1 1 0];
bitrate = 1; % bits per second
figure;
[t,s] = unrz(bits,bitrate);
plot(t,s,'LineWidth',3);
axis([0 t(end) -0.1 1.1])
grid on;
title(['Unipolar NRZ: [' num2str(bits) ']']);
figure;
[t,s] = urz(bits,bitrate);
plot(t,s,'LineWidth',3);
axis([0 t(end) -0.1 1.1])
grid on;
title(['Unipolar RZ: [' num2str(bits) ']']);
```

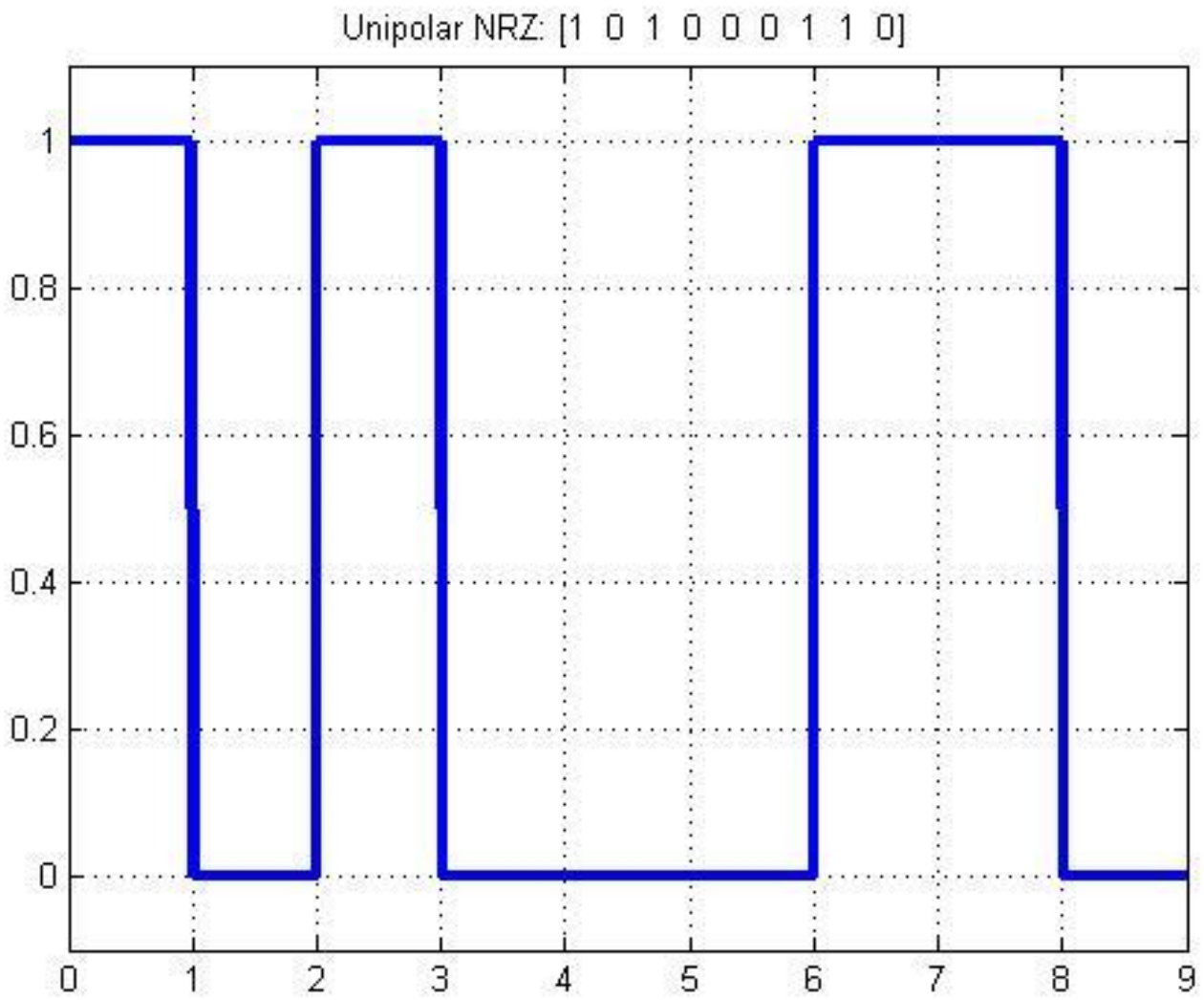
Calling the functions (Cont…)

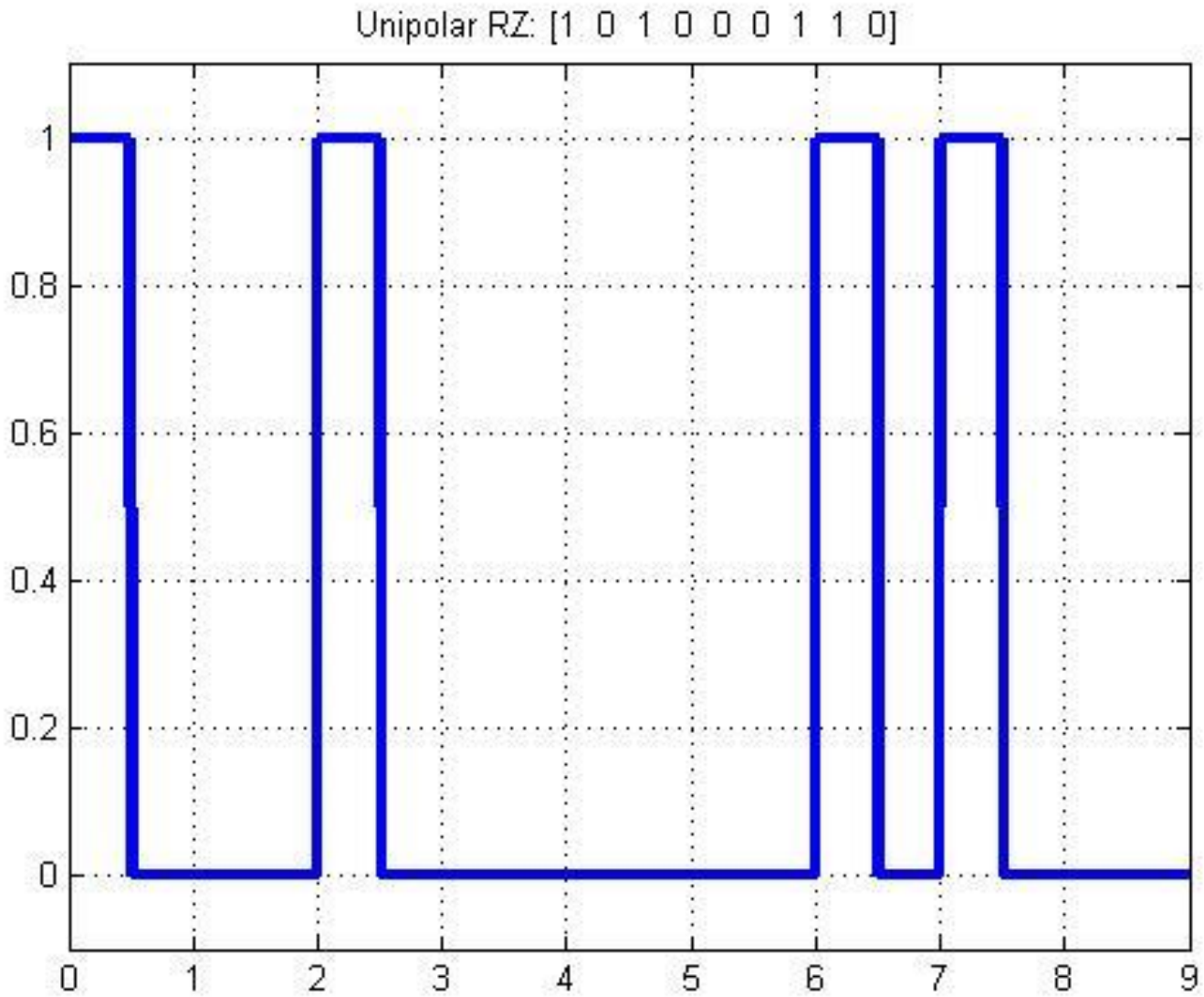
```
figure;
[t,s] = prz(bits,bitrate);
plot(t,s,'LineWidth',3);
axis([0 t(end) -1.1 1.1])
grid on;
title(['Polar RZ: [' num2str(bits) ']']);

figure;
[t,s] = manchester(bits,bitrate);
plot(t,s,'LineWidth',3);
axis([0 t(end) -1.1 1.1])
grid on;
title(['Manchester: [' num2str(bits) ']']);
```


Manchester: [1 0 1 0 0 0 1 1 0]
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7
8
9


Polar RZ: [1 0 1 0 0 0 1 1 0]
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
1
2
3
4
5
6
7
8
9


Unipolar NRZ: [1 0 1 0 0 0 1 1 0]
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7
8
9


Unipolar RZ: [1 0 1 0 0 0 1 1 0]
1
0.8
0.6
0.4
0.2
0
0
1
2
3
4
5
6
7
8
9

Lab Task

- Write a MATLAB Function for Differential Manchester line coding scheme.