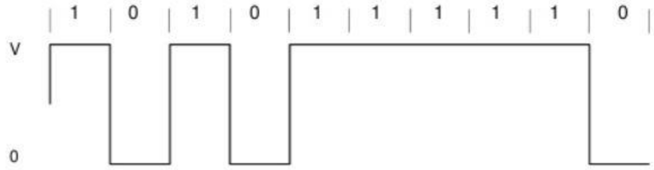
Department of Information and Communication Engineering

Course Code: ICE-3206

Course Title: Digital Communication Sessional

1. Unipolar NRZ Signaling:

In this line code, symbol 1 is represented by transmitting a pulse of amplitude A/voltage V for the duration of the symbol, and symbol 0 is represented by switching off the pulse, as illustrated in the following figure:



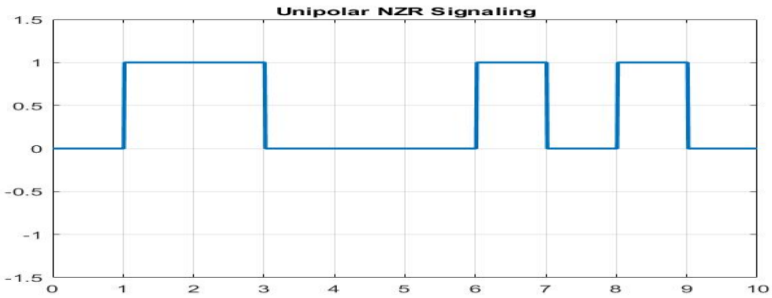
Matlab Code:

```
%Unipolar Non Return to Zero Line Coding
clc;
clear all;
close all;
N=10;                          %Number of bits
n=randi([0,1],1,N)             %Random bit generation
%Mapping Function
for m=1:N
   if n(m)==1
          nn(m)=1;
   else
          nn(m)=0;
   end
end
nn
%Signal Shaping
i=1;
t=0:0.01:length(n);            %100 Times duration set up for a single binary bit
for j=1:length(t)        %Indexing set-up for time duration
   if t(j)<=i                  %Binary input data index Check-up Condition
          y(j)=nn(i);          %Assign value from the mapping function
   else
          y(j)=nn(i);
          i=i+1;               %Binary input data index increment
   end
end
plot(t,y, 'linewidth',2);
axis([0,N,-1.5,1.5]);          %Axis set-up
grid on;
title("Unipolar NZR Signaling");
```

Output:

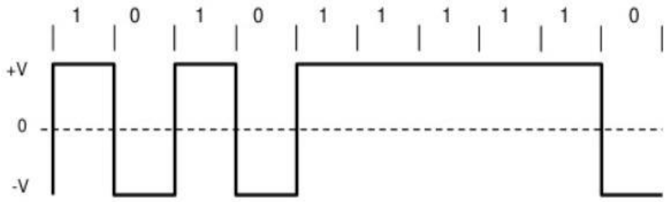
Binary Data: n =

0 1 1 0 0 0 1 0 1 0



2. Polar NRZ Signaling:

In this line code, symbols 1 and 0 are represented by transmitting pulses of amplitudes $+A$ and $-A$, respectively, as illustrated in the following figure:



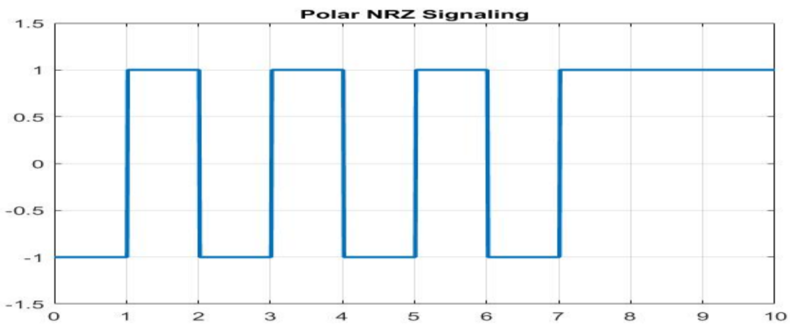
Matlab Code:

```
%Polar Non Return to Zero Line Coding
clc;
clear all;
close all;
N=10;                          %Number of bits
n=randi([0,1],1,N)             %Random bit generation
%Mapping Function
for m=1:N
   if n(m)==1
         nn(m)=1;
   else
         nn(m)=-1;
   end
end
nn
%Signal Shaping
i=1;
t=0:0.01:length(n);            %100 Times duration set up for a single binary bit
for j=1:length(t)              %Indexing set-up for time duration
   if t(j)<=i                  %Binary input data index Check-up Condition
         y(j)=nn(i);           %Assign value from the mapping function
   else
         y(j)=nn(i);
         i=i+1;                %Binary input data index increment
   end
end
plot(t,y, 'linewidth',2);
axis([0,N,-1.5,1.5]);          %Axis set-up
grid on;
title("Polar NZR Signaling");
```

Output:

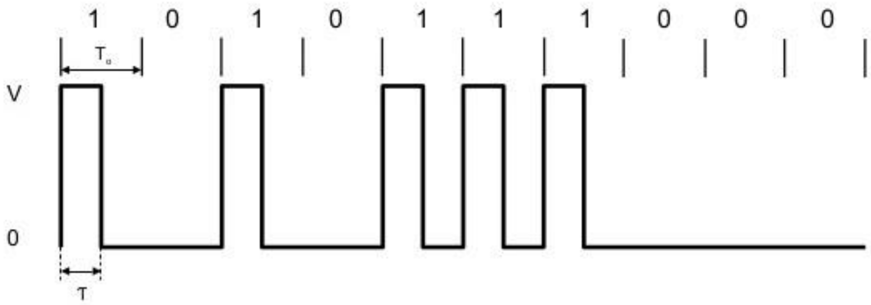
Binary Data: n =

0 1 0 1 0 1 0 1 1 1



3. Unipolar RZ Signaling:

In this line code, symbol 1 is represented by a rectangular pulse of amplitude A/ voltage v for half-symbol width, and symbol 0 is represented by transmitting no pulse, as illustrated in figure:



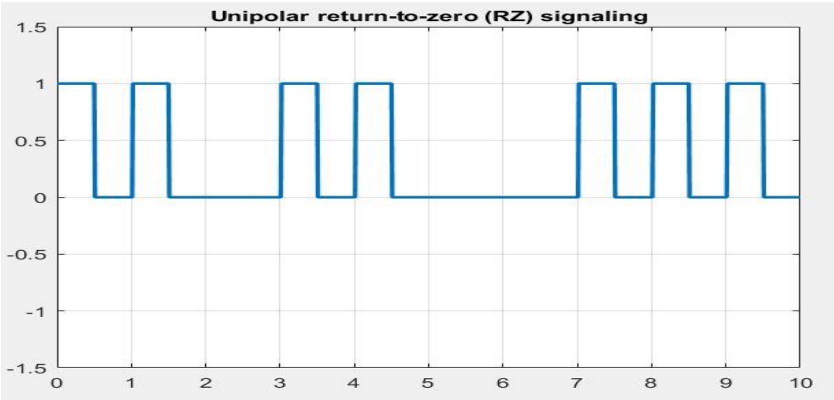
Matlab Code:

```
%RZ Unipolar line coding
clc;
clear all;
close all;
N=10;                           %Number of bits
n=randi([0,1],1,N)              %Random bit generation
%RZ Pulse Shaping
i=1;
a=0;                            %Initial value for the first half cycle
b=0.5;                          %Initial value for the second half cycle
t=0:0.01:length(n);
for j=1:length(t)
    if t(j)>=a && t(j)<=b       %Condition for the first half cycle
        y(j)=n(i);              %Assign first 50 values for
    elseif t(j)>b && t(j)<=i    %Condition for the Second half cycle
        y(j)=0;                 %Set all values 0 for the second half cycle
    else
            i=i+1;              %Binary input data index increment
            a=a+1;              %Initial value for the first half cycle increment
            b=b+1;              %Initial value for the second half cycle increment
    end
end
plot(t,y,'lineWidth', 2);       %Linewidth 2 for clear visualization
axis([0,N,-1.5,1.5]);           %Axis set-up
grid on;
title('Unipolar return-to-zero (RZ) signaling');
```

Output:

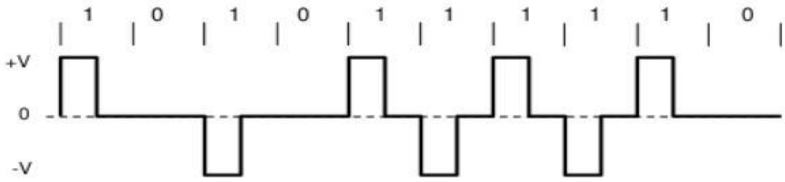
Binary Data: n =

1 1 0 1 1 0 0 1 1 1



4. **Bipolar RZ Signaling:**

This line code uses three amplitude levels as indicated in the following figure. Specifically, positive and negative pulses of equal amplitude (i.e., $+A$ and $-A$) are used alternatively for symbol 1, with each pulse having a half-symbol width; no pulse is always used for symbol 0.



Matlab Code:

```
%Bipolar Return to Zero Signaling
clc;
clear all;
close all;
N=10;                  %Number of bits
n=randi([0,1],1,N)     %Random bit generation
%Binary to Bipolar Conversion
f=1;
for m=1:N
   if n(m)==1
      if f==1
         nn(m)=1;
         f=-1;
      else
         nn(m)=-1;
         f=1;
      end
   else
      nn(m)=0;
   end
end
nn
%Bipolar RZ Pulse Shaping
i=1;
a=0;                   %Initial value for the first half cycle
b=0.5;                 %Initial value for the second half cycle
```

```
t=0:0.01:length(n);
for j=1:length(t)
if t(j)>=a && t(j)<=b       %Condition for the first half cycle
                        y(j)=nn(i);             %Assign first 50 values for
elseif t(j)>b && t(j)<=i    %Condition for the Second half cycle
                        y(j)=0;                %Set all values 0 for the second half cycle
else
            i=i+1;              %Binary input data index increment
                        a=a+1;              %Initial value for the first half cycle increment
            b=b+1;              %Initial value for the second half cycle increment
end
 end
 plot(t,y,'lineWidth', 2);       %Linewidth 2 for clear visualization
 axis([0,N,-1.5,1.5]);          %Axis set-up
 grid on;
 title('Bipolar return-to-zero (BRZ) signaling');
```

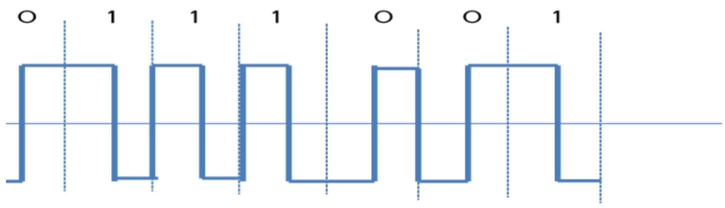
Output:

Binary Data: n =

0 0 1 1 0 0 0 1 1 1

5. Split-Phase (Manchester Code):

In this method of signaling, illustrated in the figure, symbol 1 is represented by a positive pulse of amplitude *A* followed by a negative pulse of amplitude – *A*, with both pulses being half-symbol wide. For symbol 0, the polarities of these two pulses are reserved.



Matlab Code:

```
%Split Phase-Manchester Coding
clc;
clear all;
close all;
N=10;                    %Number of bits
n=randi([0,1],1,N)       %Random bit generation
%Binary to Manchester Conversion
nnn=[];
for m=1:N
   if n(m)==1
       nn=[1 -1];
```

```
    else
        nn=[-1 1];
    end
    nnn=[nnn nn];
end
nnn
%Manchester Coding Pulse Shaping
i=1;
l=0.5;
t=0:0.01:length(n);
for j=1:length(t)
    if t(j)<=l                  %Condition for the first half cycle
        y(j)=nnn(i);              %Assign first 50 values for
    else
        y(j)=nnn(i);
        i=i+1;
        l=l+0.5;
    end
end
plot(t,y,'lineWidth', 2);         %Linewidth 2 for clear visualization
axis([0,N,-1.5,1.5]);             %Axis set-up
grid on;
title('Manchester Coding');
```

Output:

Binary Data: n =

0 0 1 0 0 0 0 0 0 1