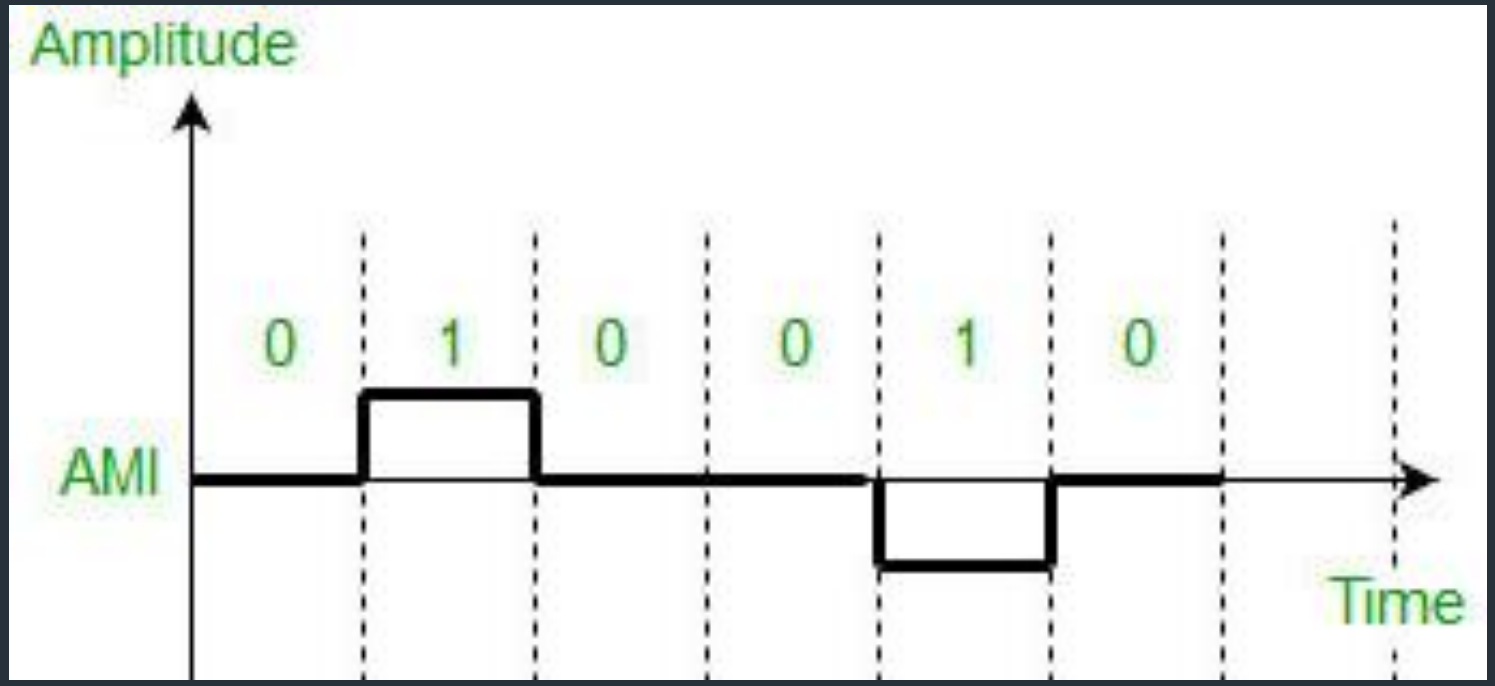
Polar Line Coding

- Alternate Mark Inversion (AMI)
- Pseudoternary

❑ Alternate Mark Inversion (AMI)

A neutral zero voltage represents binary 0. Binary 1's are represented by alternating positive and negative voltages.
Let's see how these are represented:



Step 1: Define the Binary Sequence

```
binary_sequence = [1 0 1 1 0 0 1 0]; % Example sequence
```

- ✓ This is the input sequence of bits you want to encode in the polar NRZ format.
- ✓ Each 1 will be represented by a high signal (e.g., 1), and each 0 by a low signal (e.g., -1).
- ✓ Replace this array with your desired binary data.

Step 2: Set Parameters

```
bit_duration = 1; % Duration of each bit in seconds
fs = 1000; % Sampling frequency in Hz
t = 0:1/fs:bit_duration; % Time vector for one bit
```

- ✓ bit_duration: Specifies how long each bit lasts in seconds.
- ✓ fs: Sampling frequency determines how many samples are taken per second. A higher fs provides smoother curves.
- ✓ t: Creates a time vector for one bit, with intervals of 1/fs.

Step 3: Initialize the Signal

```
bipolar_signal = [ ];
```

- ✓ bipolar_signal: This will store the final signal. Initially, it's empty and will be filled bit by bit.

Step 4: Loop Through Each Bit to Generate the Signal

```
last_polarity = -1; % Track the polarity of the last '1'
for i = 1:N
        if data(i) == 1
           % Alternate polarity for 1s
           last_polarity = -last_polarity;
           bipolar_signal = [bipolar_signal last_polarity * ones(1, length(t))];
        else
           % Zero voltage for 0s
           bipolar_signal = [bipolar_signal zeros(1, length(t))];
        end
End
```

- ✓ bipolar_signal: An array to store the encoded waveform.
- ✓ last_polarity: Keeps track of the polarity of the last 1 to ensure alternation.
- ✓ for loop: Iterates through each bit in the binary data:
  - If the bit is 1, the signal alternates between +1 and -1.
  - If the bit is 0, the signal is 0 for the entire bit duration.

Step 5: Create the Time Axis

```
total_time = length(binary_sequence) * bit_duration; % Total signal duration
time_axis = linspace(0, total_time, length(polar_nrz_signal));
```

- ✓ total_time: Calculate the total duration of the signal by multiplying the number of bits by bit_duration.
- ✓ time_axis: Create a time array that spans the entire signal, ensuring proper alignment for plotting.

Step 6: Plot the Signal

```
figure;
plot(total_time, bipolar_signal, 'LineWidth', 1.5);
grid on;
xlabel('Time (s)');
ylabel('Amplitude');
title('Bipolar Line Coding (AMI)');
ylim([-1.5 1.5]);
xlim([0 N * bit_duration]);
```

- ✓ plot: Displays the bipolar waveform.
- ✓ Labels and title: Describe the plot for clarity.
- ✓ ylim and xlim: Adjust the plot limits for better visualization.

Step 7: Annotate the Binary Sequence

```
binary_labels = arrayfun(@num2str, binary_sequence, 'UniformOutput', false);
for i = 1:length(binary_sequence)
    text((i-0.5)*bit_duration, 1.2, binary_labels{i}, 'FontSize', 12, 'HorizontalAlignment', 'center');
end
```

- ✓ arrayfun: Converts each bit to a string so you can use it as a label.
- ✓ text: Places the bit value above each corresponding bit in the plot.

Complete Code:

```
clc;clear all;close all;
binary_sequence = [1 0 1 1 0 0 1 0];

bit_duration = 1; % Duration of each bit in seconds
sampling_rate = 1000; % Samples per second
t_bit = linspace(0, bit_duration, sampling_rate);

% Bipolar Encoding
bipolar_signal = [];
last_polarity = -1; % Track the polarity of the last '1'

for i = 1:N
    if data(i) == 1
        % Alternate polarity for 1s
        last_polarity = -last_polarity;
        bipolar_signal = [bipolar_signal last_polarity * ones(1, length(t))];
    else
        % Zero voltage for 0s
        bipolar_signal = [bipolar_signal zeros(1, length(t))];
    end
end
total_time = length(binary_sequence) * bit_duration;
time_axis = linspace(0, total_time, length(polar_nrz_signal));

figure;
plot(time_axis, bipolar_signal, 'LineWidth', 2);
ylim([-1.5, 1.5]); % Adjust y-axis limits for better visualization
xlabel('Time (s)');
ylabel('Amplitude');
title(‘Bipolar Signal');
grid on;

binary_labels = arrayfun(@num2str, binary_sequence, 'UniformOutput', false);
for i = 1:length(binary_sequence)
    text((i-0.5)*bit_duration, 1.2, binary_labels{i}, 'FontSize', 12, 'HorizontalAlignment', 'center');
```

Thank You!