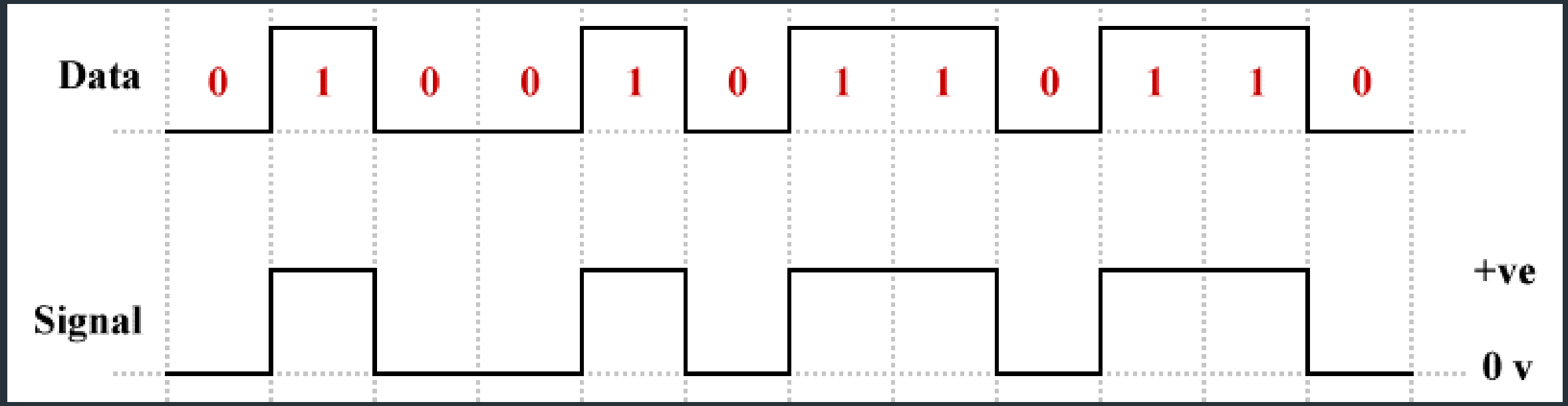
Unipolar Line Coding

❑ Unipolar NRZ



Steps

- Define the binary sequence and parameters (bit duration and sampling rate).
- Create a time array for one bit and initialize an empty signal array.
- Use a loop to append a constant signal for each bit (high for 1, low for 0).
- Create a time axis for the entire signal.
- Plot the signal and add labels to make it easy to interpret.

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 Unipolar NRZ format.
- ✓ Each 1 will be represented by a high signal (e.g., 1), and each 0 by a low signal (e.g., 0).
- ✓ Replace this array with your desired binary data.

Step 2: Set Parameters

```
bit_duration = 1;
sampling_rate = 1000;
t_bit = linspace(0, bit_duration, sampling_rate);
```

- ✓ bit_duration: The time duration for each bit (e.g., 1 second).
- ✓ sampling_rate: The number of points to represent each second. A higher sampling rate gives smoother plots.
- ✓ t_bit: Creates a time array for one bit. This helps in defining the signal's shape for a single bit.

Step 3: Initialize the Signal

```
nrz_signal = [];
```

- ✓ nrz_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

```
for bit = binary_sequence
   if bit == 1
      % For bit '1', the signal is at a high level (e.g., 1)
      nrz_signal = [nrz_signal, ones(1, length(t_bit))];
   else
      % For bit '0', the signal is at a low level (e.g., 0)
      nrz_signal = [nrz_signal, zeros(1, length(t_bit))];
   end
End
```

- ✓ Loop through each bit in binary_sequence.
- ✓ If the bit is 1, append a high-level signal (array of ones) to nrz_signal.
- ✓ If the bit is 0, append a low-level signal (array of zeros) to nrz_signal.
- ✓ The use of ones and zeros ensures the signal has a constant level for the duration of each bit.

Step 5: Create the Time Axis

```
total_time = length(binary_sequence) * bit_duration;
time_axis = linspace(0, total_time, length(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(time_axis, nrz_signal, 'LineWidth', 2);
ylim([-0.5, 1.5]);
xlabel('Time (s)');
ylabel('Amplitude');
title('Unipolar Non-Return-to-Zero (NRZ) Signal');
grid on;
```

- ✓ plot: Plots the NRZ signal against the time axis.
- ✓ ylim: Limits the y-axis to make the plot easier to interpret.
- ✓ Labels (xlabel, ylabel, title) and grid are added for clarity.

Step 7: Annotate the Binary Sequence

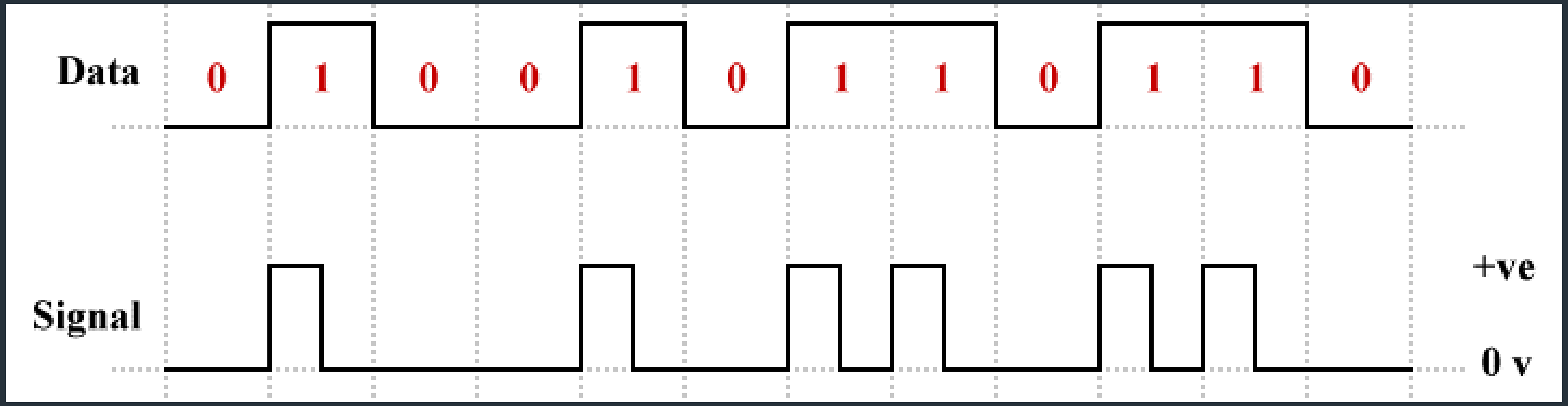
```
binary_labels = arrayfun(@num2str, binary_sequence, 'UniformOutput', false);
for i = 1:length(binary_sequence)
    text((i-0.5)*bit_duration, 1.2, binary_labels{i}, 'FontSize', 12, 'HorizontalAlignment', 'center');
end
```

- ✓ 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;
sampling_rate = 1000;
t_bit = linspace(0, bit_duration, sampling_rate);
nrz_signal = [];

for bit = binary_sequence
    if bit == 1
        nrz_signal = [nrz_signal, ones(1, length(t_bit))];
    else
        nrz_signal = [nrz_signal, zeros(1, length(t_bit))];
    end
end

total_time = length(binary_sequence) * bit_duration;
time_axis = linspace(0, total_time, length(nrz_signal));

figure;
plot(time_axis, nrz_signal, 'LineWidth', 2);
ylim([-0.5, 1.5]);
xlabel('Time (s)');
ylabel('Amplitude');
title('Unipolar Non-Return-to-Zero (NRZ) 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');
end
```

Unipolar RZ



1. Define the Binary Sequence

```
binary_sequence = [1 0 1 1 0 0 1 0];
```

- ✓ Input the binary data you want to encode in Unipolar RZ format.
- ✓ Modify this sequence to test with different binary values.

2. Set Parameters

```
bit_duration = 1;
sampling_rate = 1000;
t_bit = linspace(0, bit_duration, sampling_rate);
```

- ✓ bit_duration: Time allocated to each bit in the signal.
- ✓ sampling_rate: Determines the resolution of the signal.
- ✓ t_bit: Creates a time vector for one bit.

3. Initialize the Signal

```
rz_signal = [];
```

- ✓ An empty array to store the entire Unipolar RZ signal.

4. Generate the RZ Signal

```
for bit = binary_sequence
    if bit == 1
        rz_high = [ones(1, length(t_bit)/2), zeros(1, length(t_bit)/2)];
    else
        rz_high = zeros(1, length(t_bit));
    end
    rz_signal = [rz_signal, rz_high];
end
```

- ✓ Loop through each bit in the binary sequence:
- ✓ For 1: High level for the first half, then return to zero.
- ✓ For 0: Always low level.
- ✓ Append the generated signal for each bit to the final signal array.

5. Create the Time Axis

```
total_time = length(binary_sequence) * bit_duration;
time_axis = linspace(0, total_time, length(rz_signal));
```

- ✓ Compute the total duration of the signal.
- ✓ Create a time array for plotting the entire signal.

6. Plot the RZ Signal

```
figure;
plot(time_axis, rz_signal, 'LineWidth', 2);
ylim([-0.5, 1.5]); % Adjust y-axis limits for better visibility
xlabel('Time (s)');
ylabel('Amplitude');
title('Unipolar Return-to-Zero (RZ) Signal');
grid on;
```

- ✓ Plot the signal against the time axis.
- ✓ Use ylim to control the range of the y-axis for clear visualization.

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
```

- ✓ Add text above each bit interval to show the corresponding binary value.

Complte Code:

```
clc;clear all;close all;
binary_sequence = [1 0 1 1 0 0 1 0];
bit_duration = 1;
sampling_rate = 1000;
t_bit = linspace(0, bit_duration, sampling_rate);
rz_signal = [];

for bit = binary_sequence
    if bit == 1
        rz_high = [ones(1, length(t_bit)/2), zeros(1, length(t_bit)/2)];
    else
        rz_high = zeros(1, length(t_bit));
    end
    rz_signal = [rz_signal, rz_high];
end

total_time = length(binary_sequence) * bit_duration;
time_axis = linspace(0, total_time, length(rz_signal));

figure;
plot(time_axis, rz_signal, 'LineWidth', 2);
ylim([-0.5, 1.5]);
xlabel('Time (s)');
ylabel('Amplitude');
title('Unipolar Return-to-Zero (RZ) 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');
end
```

Thank You!