Lab Report No: 03

Experiment Name: Signal generation and plotting in MATLAB2

Objective:

The objective of this lab is to generate and analyze also to explore different time-domain and frequency domain characteristics of signals using MATLAB.

Materials Required:

- MATLAB software R2024a
- Computer system with windows
- Basic knowledge of MATLAB scripting

MATLAB Code and Procedure:

Signal 1: $x(t) = Ae^{-\alpha t}\sin(2\pi f t)$

```
A = 1;
f = 5;
alpha = 2;
t = 0:0.01:2;
x = A*exp(-alpha*t).*sin(2*pi*f*t);
plot(t, x);
title('Damped Sine Wave KULDIP(IT22018)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
```

Signal 2: $x(t) = A e^{-\alpha t}$

```
A = 5;
alpha = 2;
duration = 2;
samplingRate = 10;
t = 0:1/samplingRate:duration;
x = A* exp(-alpha *t);
plot (t,x);
xlabel ('Time (s)');
ylabel ('Amplitude');
title ('Exponential Decay: x(t) = A·exp(-a·t)
                                KVLDIP (IT22018)');
```

Signal 3: $x(t) = \sin(2\pi 50 t)$

```
Fs = 10;
t = 0:1/Fs:1-1/Fs;
x = sin(2*pi*50*t);
N = length(x);
X = fft(x);
f = (0:N-1)*(Fs/N);
amplitude = abs(X)/N;
display (amplitude);
figure;
plot (f, amplitude);
xlabel ('Frequency (Hz)');
ylabel ('Amplitude');
title ('Frequency vs Amplitude  KVLDIP (IT22018)');
```

Signal 4 : $x(t) = \sin(2\pi f_1 t) + 0.5 \sin(2\pi f_2 t)$

```
Fs = 10;
t = 0:1/Fs : 1-1/Fs;
x = sin(2*pi*50*t) + 0.5*sin(2*pi*150*t);
N = length(x);
X = fft(x);
f = (0 : N-1) * (Fs/N);
amplitude = abs(X)/N;
figure;
plot(f, amplitude);
xlabel('Frequency (Hz)');
ylabel('Amplitude');
title('Frequency Domain Representation KULDIP(IT22018)');
xlim([0 Fs/2]);
grid on;
```

Conclusion:

The lab showed how signals change over time and how incorrect sampling causes aliasing it highlighted the need for proper sampling and demonstrated key signal processing concepts using MATLAB.

CamScanner

Lab Report NO.: 04

Experiment name: Signal generation and plotting in MATLAB

Objective:

To generate and analyze various types of periodic waveforms (Pulse, sawtooth, damped sin, square and triangular) using MATLAB scripting and signal processing techniques.

Materials Required:

MATLAB software R2024a, Computer system with windows and basic knowledge of MATLAB scripting.

4.1 Pulse Wave

```
f = 5;
duty = 50;
duration = 1;
fs = 1000;
t = 0: 1/fs : duration;
x = square(2*Pi*f*t, duty);
x = (x + 1)/2;
plot (t, x, 'Line Width', 1.5);
xlabel('Time (s)');
ylabel ('Amplitude');
title ('Pulse wave (50% Duty Cycle) kuldip (IT22018)');
```

4.2 Manually Generated Sawtooth Wave

```
f = 2;
a = 3;
d = 4;
samplingRate = 1000;

t = 0:1/samplingRate:d;

x = 2*a*(t*f - floor(t*f + 0.5));

plot(t,x,'Linewidth', 1.5);
xlabel('Time(s)');
ylabel('Amplitude');
title('Manually Generated Sawtooth Signal KULDIP
                                    (IT22018)');
```

4.3 Sawtooth Using Built-in Function

```
f = 2;
a = 3;
d = 4;
samplingRate = 1000;
t = 0:1/samplingRate:d ~~duration~~;
x = a * sawtooth(2*Pi*f*t);
plot(t,x,'LineWidth',1.5);
xlabel('Time (s)');
ylabel('Amplitude');
title('Sawtooth Signal : x(t) = A. sawtooth(2πft)
        KULDIP (IT22018)');
```

4.4 Damped Sine Wave

```
A = 1;
f = 5;
alpha = 2;
t = 0:0.001:2;
x = A* exp(-alpha*t).*sin(2*pi*f*t);
plot(t,x);
title('Damped Sine wave KVLDIP(IT22018)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
```

CamScanner

4.5 Square wave using sign() Function

```
f = 1;
a = 2;
d = 3;
samplingRate = 100;

t = 0:1/samplingRate:d;

x = a * sign(sin(2*pi*f*t));
plot(t, x);
xlabel('Time(s)');
ylabel('Amplitude');
title('Sine Square Signal KULDIP(IT22038)');
```

CamScanner

4.6 Manually Generated Triangle Wave

```
f = 2;
a = 3;
d = 5;
samplingRate = 100;
t = 0:1/samplingRate:d;
x= 4 * a * abs(t * f - floor(t * frequency + 0.5)) - a;
plot(t, x, 'LineWidth, 1.5);
xlabel('Time (s)');
ylabel('Amplitude');
title('Manually Triangular wave KULDIP (IT22018)');
```

Conclusion:

The MATLAB simulations successfully generated a variety of waveform types using both built-in and manual coding techniques. Each waveform holds importance in communication, control, and signal processing fields. These experiments demonstrate foundational concepts of periodic signals, harmonic content, and waveform synthesis through code.

CamScanner

Lab Report NO: 05

Experiment Name: Signal generation and plotting in MATLAB4

Objective:

The objective of this lab is to simulate and visualize various line coding techniques used in data communication systems. These include Polar NRZ, Polar RZ, Unipolar NRZ, Bipolar AMI, 2B1Q, and Manchester encoding. The main goal is to understand how binary data is represented and transmitted over physical channels using different encoding schemes.

Material Required:

- MATLAB software R2024a
- Computer system with windows
- Basic knowledge of MATLAB scripting.

CamScanner

5.1 Polar NRZ Line Coding

```
clc;
clear;
data = [1 0 1 1 0 0 1 0];
bit_duration = 1;
fs = 100;
t = 0 : 1/fs : bit_duration - 1/fs;
N = length(data);
time = 0 : 1/fs : bit_duration*N - 1/fs;
Polar_nrz = [ ];
for i = 1 : N
    if data(i) == 1
        Polar_nrz = [Polar_nrz ones(1, length(t))];
    else
        Polar_nrz = [Polar_nrz -ones(1, length(t))];
    end
end
figure;
plot(time, Polar_nrz, 'LineWidth', 2);
axis([0 bit_duration*N -1.5 1.5]);
grid on;
xlabel('Time (s)');
ylabel('Amplitude (V)');
title('Polar NRZ Line Coding KULDIP IT22038');
xticks(0 : bit_duration : N);
yticks([-1 0 1]);
```

5.2 Polar RZ Line Coding

```
clc;
clear;
data = [1 0 1 1 0 0 1 0];
bit_duration = 1;
fs = 100;
samples_per_bit = fs * bit_duration;
half_samples = samples_per_bit/2;
time = 0:1/fs:(length(data) * bit_duration) - 1/fs;
polar_rz = [];
for i = 1 . length(data)
    if data(i) == 1
        bit_wave = [ones(1, half_samples, zero(1, half_samples)];
    else
        bit_wave = [-ones(1, half_samples, zero(1, half_samples)];
    end
  polar_rz = [polar_rz bit_wave];
end
figure;
plot(time, polar_rz, 'LineWidth', 2);
axis([0 length(data) * bit_duration -1.5 1.5]);
grid on; xlabel('Time (s)'); ylabel('Amplitude');
title('Polar RZ Line Coding KULDIP (IT22018)');
xticks(0:bit_duration:length(data));
yticks([-1 0 1]);
```

5.3 Unipolar NRZ Line Coding

```
clc;
clear;
data=[1 0 1 1 0 0 1 0];
bit_duration = 1;
fs = 100;
t = 0:1/fs:bit_duration-1/fs;
N = length(data);
time = 0:1/fs:bit_duration*N-1/fs;
Unipolar_nrz = [ ];
for i = 1:N
    if data(i) == 1
        unipolar_nrz=[unipolar_nrz ones(1, length(t))];
    else
        unipolar_nrz=[unipolar_nrz zeros(1, length(t))];
    end
end
figure;
plot(time, unipolar_nrz, 'LineWidth', 2);
axis([0 bit_duration*N 0 1]);
grid on;
xlabel('Time (s)');
ylabel('Amplitude');
title('Unipolar NRZ Line Coding KULDIP (IT22018)');
xticks(0:bit_duration:N);
yticks([0 1]);
```

5.1 Bipolar AMI

```
clc;
clear;
data = [1 0 1 1 0 1 0 0 1];
bit_duration = 1;
fs = 100;
samples_per_bit = fs * bit_duration;
time = 0 : 1/fs : (length(data) * bit_duration) - 1/fs;
ami = [ ];
last_polarity = -1;
for i = 1 : length(data)
    if data(i) == 1
        last_polarity = -last_polarity;
        bit_wave = last_polarity * ones(1, samples_per_bit);
    else
        bit_wave = zeros(1, samples_per_bit);
    end
    ami = [ami bit_wave];
end
figure; plot(time, ami, 'LineWidth', 2);
axis([0 length(data) * bit_duration -1.5 1.5]);
grid on; xlabel('Time (s)'); ylabel('Amplitude (V)');
title('Bipolar AMI Line coding KULDIP (IT22018)');
xticks(0 : bit_duration : length(data));
yticks([-1 0 1]);
```

5.5 2B1Q

```
clc;
clear;
data = [1 0 0 0 1 1 0 1];
if mod(length(data),2)~=0
   error('Input data length must be even for 2B1Q.');
end

fs = 100;
symbol_duration = 1;
samples_per_symbol = fs * symbol_duration;
num_symbols = length(data)/2;
time = 0:1/fs:(num_symbols * symbol_duration) - 1/fs;
map = containers.Map({'00','01','10','11'}, [-3,-1,1,3]);
twoB1Q = [ ];
for i = 1:2:length(data)
   Pair = sprintf('%d%d', data(i), data(i+1));
   level = map(pair);
   twoB1Q = [twoB1Q level * ones(1, samples_per_symbol)];
end

figure;
plot(time, twoB1Q, 'LineWidth', 2);
axis([0 num_symbols * symbol_duration -3.5 3.5]);
grid on;
xlabel('Time (s)');
ylabel('Amplitude (V)');
```

```
title('2B1Q Line Coding KULDIP (IT22018)');
xticks(0:symbol_duration:num_symbols);
yticks([-3 -1 0 1 3]);
```

5.6 Manchester (Polar Biphase) Line Coding

```
clc;
clear;
data = [1 0 1 1 0 0 1 0];
bit_duration = 1;
fs = 100;
samples_per_bit = fs * bit_duration;
half_samples = samples_per_bit/2;
time = 0:1/fs:(length(data) * bit_duration) - 1/fs;
manchester = [ ];
for i = 1 : length(data)
    if data(i) == 1
        bit_wave = [ones(1, half_samples), -ones(1, half_samples)];
    else
        bit_wave = [-ones(1, half_samples), ones(1, half_samples)];
    end
manchester = [manchester bit_wave];
```

```
end
figure;
plot(time, manchester, 'LineWidth', 2);
axis([0 length(data)*bit_duration -3.5 3.5]);
grid on;
xlabel('Time (s)'); ylabel('Amplitude (V)');
title('Manchester Line coding NUL DIP (IT22039)');
xticks(0:bit_duration:length(data));
yticks([-3 0 3]);
```

Conclusion:

This lab demonstrated how different line coding techniques represent binary data in signal form. Each method has its own trade-offs in complexity, synchronization, and bandwidth. Visualizing these in MATLAB helped understand their practical applications and behavior in data communication system.

CamScanner

Lab Report NO.: 06

Experiment Name: Signal Generation and Plotting in MATLAB 5

Objective:

To simulate and analyze Frequency Modulation (FM), Pulse Code Modulation (PCM), Amplitude Modulation (AM), and Delta Modulation (DM) using MATLAB.

Materials Required:

- MATLAB software R2024a
- Computer system with windows
- Basic knowledge of MATLAB scripting

6.1 Frequency Modulation (FM)

```
Fs = 10000;
t = 0:1/Fs:0.01;
fc = 1000;
fm = 100;
Am = 1;
Ac = 1;
kf = 2*pi*75;
m = Am*cos(2*pi*fm*t);
int_m = cumsum(m)/Fs;
s_fm = Ac*cos(2*pi*fc*t + kf*int_m);
figure;
subplot(3,1,1);
plot(t,m);
title('Message Signal KULDIP IT22038');
```

```
xlabel ('Time (s)'); ylabel ('Amplitude');
subplot (3, 1, 2);
plot (t, cos (2*pi * fc * t));
title ('Carrier Signal KULDIP IT22018');
xlabel ('Time (s)'); ylabel('Amplitude');
subplot (3, 1, 3);
plot (t, s_fm);
title('Fm Signal KULDIP IT22018');
xlabel('Time (s)');
ylabel('Amplitude');
```

6.2 Pulse Code Modulation (PCM)

```
clc;
clear;
f = 5; fs = 100;
t = 0:1/fs :1;
x = sin(2*pi*f*t);
n_bits = 2;
L = 2^n_bits;
x_min = min(x); x_max = max(x);
q_step = (x_max - x_min)/(L-1);
xq_index = round((x - x_min)/q_step);
xq = xq_index * q_step + x_min;
bin_pcm = dec2bin(xq_index, n_bits);
figure;
plot(t, x, 'b', 'LineWidth', 1.5); hold on;
stairs(t, xq, 'r', 'LineWidth', 1.5);
xlabel('Time (s)'); ylabel('Amplitude');
title('PCM Encoding of a sine wave KULDIP IT22018');
legend('Original Signal', 'Quantized Signal');
grid on;
```

6.3 Amplitude Modulation (AM)

```
fs = 10000;
t = 0:1/fs:0.01;
fc = 1000; fm = 100;
Am = 1; Ac = 1; mu = 0.5;
m = Am * cos(2 * pi * fm * t);
c = Ac * cos(2 * pi * fc * t);
s = (1 + mu * cos(2 * pi * fm * t)). * c;
figure;
subplot (3, 1, 1);
plot (t, m);
title ('Message Signal KULDIP IT22038');
x label ('Time (s)');
y lased ('Amplitude');

subplot (3, 1, 2);
plot ( t, c);
title ('Carrier Signal KULDIP IT22038');
x label ('Time (s)'); y label ('Amplitude');

subplot (3, 1, 3);
plot (t, s);
title ('AM Modulated Signal KULDIP IT22038');
x lased ('Time (s)'); y lased ('Amplitude');
```

6.4 Delta Modulation (DM)

```
Clc;
clear;
fs =100;
t = 0:1/fs:1;
x = sin(2*pi*5*t);
delta = 0.2;

x_dm = zeros(size(x));
bitstream = zeros(size(x));

for i = 2:length(x)
    if x(i)> x_dm(i-1)
        bitstream(i)=1;
        x_dm(i) = x_dm(i-1) + delta;
    else
        bitstream(i)=0;
        x_dm(i) = x_dm(i-1) - delta;
    end
end

figure;
plot(t,x,'b','LineWidth',1.5); hold on;
stairs(t,x_dm,'r','LineWidth',1.5);
xlabel('Time(s)'); ylabel('Amplitude');
title('Delta Modulation of a sine wave KULDIP
        IT22018');
```

```
legend ('Original Signal', 'Delta Modulated
        Reconstruction');
grid on;
```

Conclusion:

This lab demonstrate the implementation of FM, AM, PCM, and DM using MATLAB. FM and AM showed how analog signals can modulate carrier waves, while PCM and DM converted analog signals into digital formats. Each techniques has its unique advantages and applications in Data Communication systems.

CamScanner