**American International University- Bangladesh**

**Department of Electrical and Electronic Engineering**

COE 3201: Data Communication Laboratory

**Title: Frequency Division Multiplexing using MATLAB**

**Abstract:**

This experiment is designed to-

1. To understand the use of MATLAB for solving communication engineering problems.

2.To develop understanding of FDM concept and how to implement it in Matlab.

**Introduction:**

**Frequency-division multiplexing (FDM)** is an analog technique that can be applied when the bandwidth of a link (in hertz) is greater than the combined bandwidths of the signals to be transmitted. In FDM, signals generated by each sending device modulate different carrier frequencies. These modulated signals are then combined into a single composite signal that can be transported by the link. Carrier frequencies are separated by sufficient bandwidth to accommodate the modulated signal. These bandwidth ranges are the channels through which the various signals travel. Channels can be separated by strips of unused bandwidth—guard bands—to prevent signals from overlapping. In addition, carrier frequencies must not interfere with the original data frequencies.

Figure 1 gives a conceptual view of FDM. In this illustration, the transmission path is divided into three parts, each representing a channel that carries one transmission.

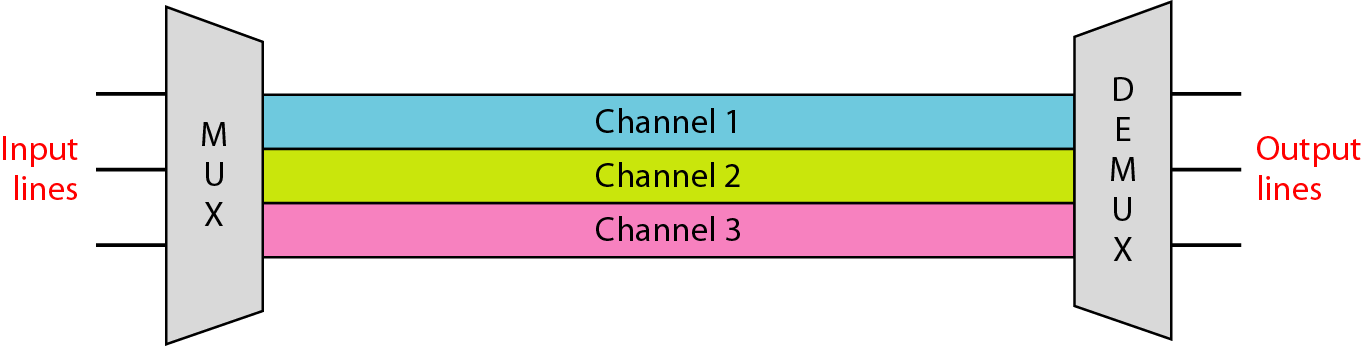


Figure 1: Frequency Division Multiplexing (FDM)

We consider FDM to be an analog multiplexing technique; however, this does not mean that FDM cannot be used to combine sources sending digital signals. A digital signal can be converted to an analog signal (Using ASK, FSK, PSK, QAM) before FDM is used to multiplex them.

**Multiplexing Process:** Figure 2 is a conceptual illustration of the multiplexing process. Each source generates a signal of a similar frequency range. Inside the multiplexer, these similar signals modulate different carrier frequencies ( f1, f2, and f3). The resulting modulated signals are then combined into a single composite signal that is sent out over a media link that has enough bandwidth to accommodate it.

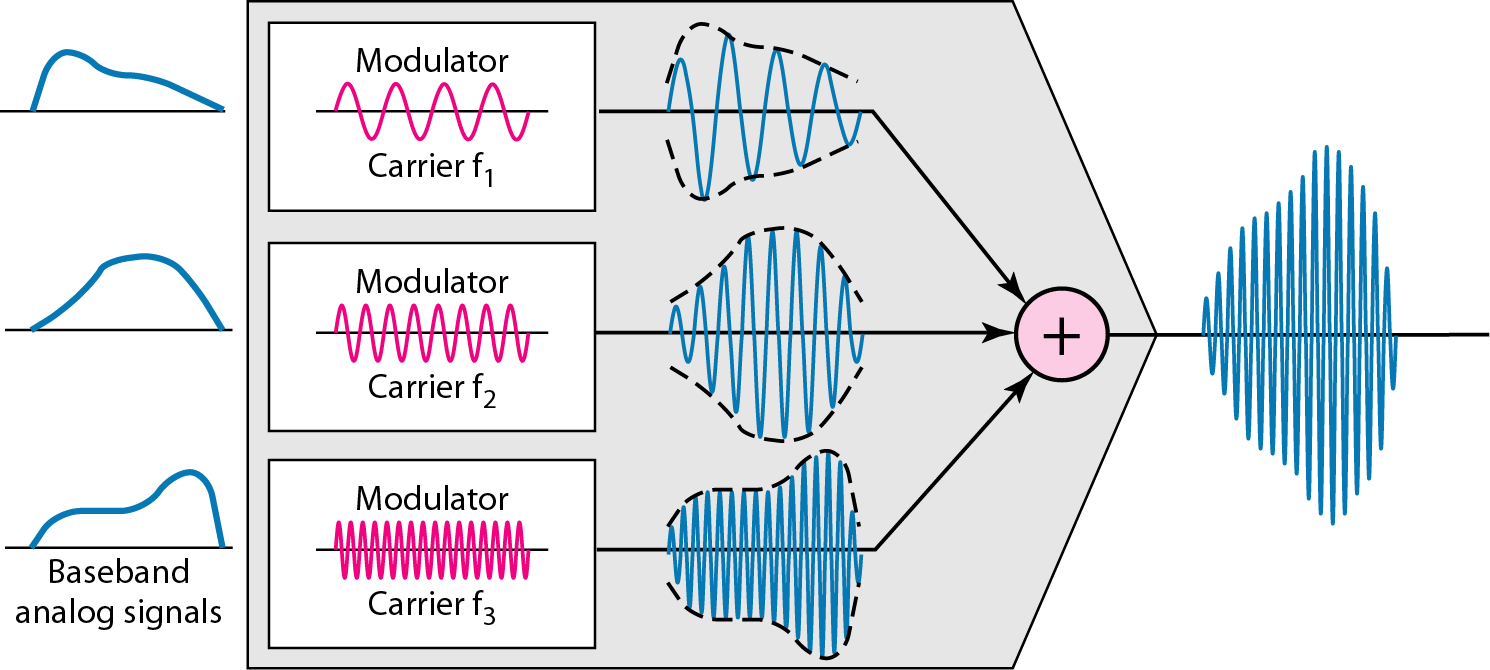


Figure 2: Multiplexing Process in FDM

**Demultiplexing Process:** The demultiplexer uses a series of filters to decompose the multiplexed signal into its constituent component signals. The individual signals are then passed to a demodulator that separates them from their carriers and passes them to the output lines. Figure 3 is a conceptual illustration of demultiplexing process.

**Implementation:** FDM can be implemented very easily. In many cases, such as radio and television broadcasting, there is no need for a physical multiplexer or demultiplexer. As long a the stations agree to send their broadcasts to the air using different carrier frequencies, multiplexing is achieved. **To make sure we are transmitting signals in different frequencies we need to use AM, FM, PM with suitable carrier frequencies.** In other cases, such as the cellular telephone system, a base station needs to assign a carrier frequency to the telephone user. There is not enough bandwidth in a cell to permanently assign a bandwidth range to every telephone user. When a user hangs up, her or his bandwidth is assigned to another caller.

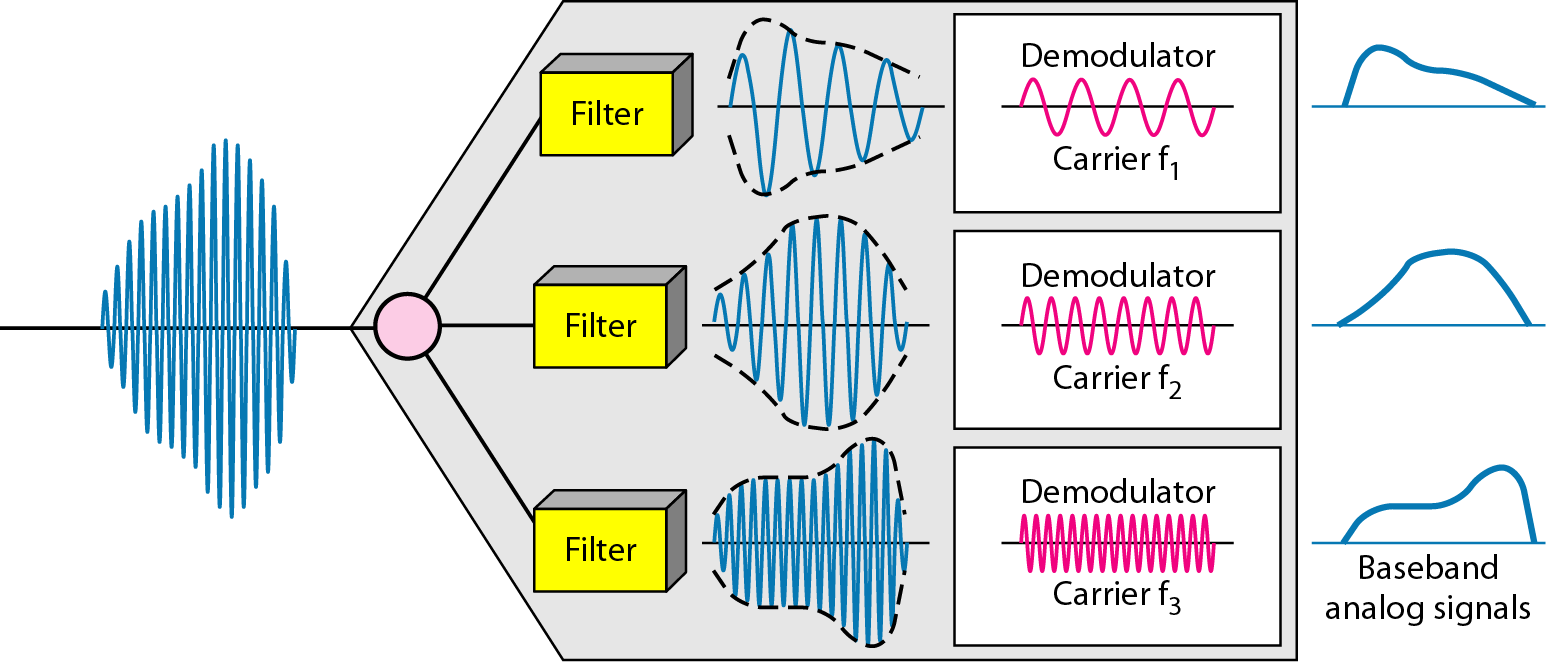


Figure 3: De-multiplexing Process in FDM

An example code of doing FDM in MATLAB is given below:

|  |
| --- |
| %Lab 10  %Matlab Program for Frequency Division Multiplexing  clc  clear all  close all  %% Message Signal Generation  fs = 4001; %Sampling Frequency  t = 0:1/fs:1-1/fs; %Generating Time axis  Am1 = 2; %Amplitude of First Message Signal  fm1 = 4; %Frequency of First Message Signal  m1 = Am1\*cos(2\*pi\*fm1\*t); % First Message Signal  Am2 = 3; %Amplitude of Second Message Signal  fm2 = 5; %Frequency of Second Message Signal  m2 = Am2\*cos(2\*pi\*fm2\*t); % Second Message Signal  Am3 = 4; %Amplitude of Third Message Signal  fm3 = 6; %Frequency of Third Message Signal  m3 = Am3\*cos(2\*pi\*fm3\*t); % Third Message Signal  %%  %% Carrier Signal Generation  Cm1 = 1; %Amplitude of First Carrier Signal  fc1 = 100; %Frequency of First Carrier Signal  c1 = Cm1\*cos(2\*pi\*fc1\*t); % First Carrier Signal  Cm2 = 1; %Amplitude of Second Carrier Signal  fc2 = 170; %Frequency of Second Carrier Signal  c2 = Cm2\*cos(2\*pi\*fc2\*t); % Second Carrier Signal  Cm3 = 1; %Amplitude of Third Carrier Signal  fc3 = 250; %Frequency of Third Carrier Signal  c3 = Cm3\*cos(2\*pi\*fc3\*t); % Third Carrier Signal  %%  %% Composite Signal Generation  x = (m1).\*c1+(m2).\*c2+(m3).\*c3;  %%%% Plotting the Signals in Time-Domain and Frequency-Domain  figure  subplot(3,1,1)  plot(t,m1)  xlabel('time')  ylabel('amplitude')  title('Message Signal 1 in Time Domain')  ylim([-Am1 Am1])  subplot(3,1,2)  plot(t,m2)  xlabel('time')  ylabel('amplitude')  title('Message Signal 2 in Time Domain')  ylim([-Am2 Am2])  subplot(3,1,3)  plot(t,m3)  xlabel('time')  ylabel('amplitude')  title('Message Signal 3 in Time Domain')  ylim([-Am3 Am3])  M1 = abs(fftshift(fft(m1)))/(fs/2); %Fourier Transformation of m1  M2 = abs(fftshift(fft(m2)))/(fs/2); %Fourier Transformation of m2  M3 = abs(fftshift(fft(m3)))/(fs/2); %Fourier Transformation of m3  X = abs(fftshift(fft(x)))/(fs/2); %Fourier Transformation of x  f = fs/2\*linspace(-1,1,fs);  figure  subplot(3,1,1)  stem(f,M1)  xlabel('frequency')  ylabel('amplitude')  title('Message Signal 1 in Frequency Domain')  axis([-10 10 0 2.5])  subplot(3,1,2)  stem(f,M2)  xlabel('frequency')  ylabel('amplitude')  title('Message Signal 2 in Frequency Domain')  axis([-10 10 0 3.5])  subplot(3,1,3)  stem(f,M3)  xlabel('frequency')  ylabel('amplitude')  title('Message Signal 3 in Frequency Domain')  axis([-10 10 0 4.5])  figure  subplot(2,1,1)  plot(t,x)  xlabel('time')  ylabel('amplitude')  title('Composite Signal in Time Domain')  subplot(2,1,2)  stem(f,X)  xlabel('frequency')  ylabel('amplitude')  title('Composite Signal in Frequency Domain')  axis([-270 270 0 2.5])  %%  %% Passing the Composite Signal Through Bandpass Filter  [num1, den1] = butter(5, [(fc1-fm1-6)/(fs/2),(fc1+fm1+6)/(fs/2)]);  %Butterworth Filter Window Determining for Bandpass Filter  bpf1 = filter(num1,den1,x); %Filtering is done here  [num2, den2] = butter(5, [(fc2-fm2-6)/(fs/2),(fc2+fm2+6)/(fs/2)]);  %Butterworth Filter Window Determining for Bandpass Filter  bpf2 = filter(num2,den2,x); %Filtering is done here  [num3, den3] = butter(5, [(fc3-fm3-6)/(fs/2),(fc3+fm3+6)/(fs/2)]);  %Butterworth Filter Window Determining for Bandpass Filter  bpf3 = filter(num3,den3,x); %Filtering is done here  %%  %% Mixing  z1 = 2\*bpf1.\*c1;  z2 = 2\*bpf2.\*c2;  z3 = 2\*bpf3.\*c3;  %%  %% Passing the Mixed Signals Through Lowpass Filter  [num4, den4] = butter(5, (fm1+3)/(fs/2)); %Low pass filter is made here  rec1 = filter(num4,den4,z1); %Filtering is done here  [num5, den5] = butter(5, (fm2+3)/(fs/2)); %Low pass filter is made here  rec2 = filter(num5,den5,z2); %Filtering is done here  [num6, den6] = butter(5, (fm3+3)/(fs/2)); %Low pass filter is made here  rec3 = filter(num6,den6,z3); %Filtering is done here  %%  %% Plotting the Received Signals in Time-Domain and Frequency Domain  figure  subplot(3,1,1)  plot(t,rec1)  xlabel('time')  ylabel('amplitude')  title('received signal 1 in time domain')  ylim([-Am1 Am1])    subplot(3,1,2)  plot(t,rec2)  xlabel('time')  ylabel('amplitude')  title('received signal 2 in time domain')  ylim([-Am2 Am2])    subplot(3,1,3)  plot(t,rec3)  xlabel('time')  ylabel('amplitude')  title('received signal 3 in time domain')  ylim([-Am3 Am3])    R1 = abs(fftshift(fft(rec1)))/(fs/2); %Fourier Transformation is done here  R2 = abs(fftshift(fft(rec2)))/(fs/2); %Fourier Transformation is done here  R3 = abs(fftshift(fft(rec3)))/(fs/2); %Fourier Transformation is done here    figure  subplot(3,1,1)  stem(f,R1)  xlabel('frequency')  ylabel('amplitude')  title('received signal 1 in frequency domain')  xlim([-10 10])    subplot(3,1,2)  stem(f,R2)  xlabel('frequency')  ylabel('amplitude')  title('received signal 2 in frequency domain')  xlim([-10 10])    subplot(3,1,3)  stem(f,R3)  xlabel('frequency')  ylabel('amplitude')  title('received signal 3 in frequency domain')  xlim([-10 10])  %% End |

**Performance Task for Lab Report: (your ID = AB-CDEFG-H)**

You have four message signals:

a) mt1 = am1\*cos(2\*pi\*fm1\*t);

b) mt2 = am2\*cos(2\*pi\*fm2\*t);

c) mt3 = am3\*cos(2\*pi\*fm3\*t);

d) mt4 = am4\*cos(2\*pi\*fm4\*t);

where,

am1 = (F+2);

am2 = (F+5);

am3 = (F+8);

am4 = (F+11);

and

fm1 = (G+1);

fm2 = (G+2);

fm3 = (G+3);

fm4 = (G+4);

We want to simultaneously transmit these four signals through a single data link that can support a frequency range of 50 Hz to 250 Hz.

\*\*\* Write a code that can modulate and multiplex the four given message signals in transmitting side (use appropriate carrier signals for amplitude modulation as required) and de-multiplex (use appropriate cut-off frequencies in your bandpass filters) and de-modulate (use appropriate cut-off frequencies in your lowpass filters) to recover the four message signals in receiving side