Netaji Subhas University of Technology

MOBILE COMPUTING

PRACTICAL FILE

ROLL NO.: 2022UIT3046 to 2022UIT3060
BRANCH: Information Technology(Section-1)

NAMES:

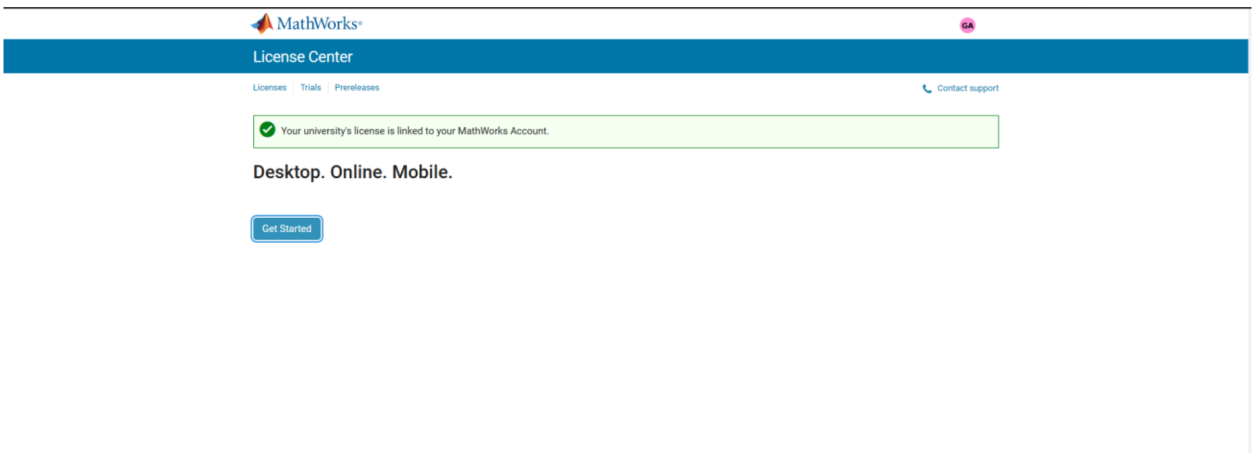
2022UIT3046 - Jessica
2022UIT3047 - Varun
2022UIT3048 - Vaishant Sharma
2022UIT3049 - Omprakash
2022UIT3050 - Gautam Arora
2022UIT3051 - Aayush Dubey
2022UIT3052 - Jeetesh Meena
2022UIT3053 - Aman Chaudhary
2022UIT3054 - Aditya Kumar S.K Lal
2022UIT3055 - Ashfaq
2022UIT3056 - Sumit
2022UIT3057 - Mridul
2022UIT3058 - Pankaj
2022UIT3059 - Rohan Paul
2022UIT3060 - Nikhil Kumar Shah

INDEX:

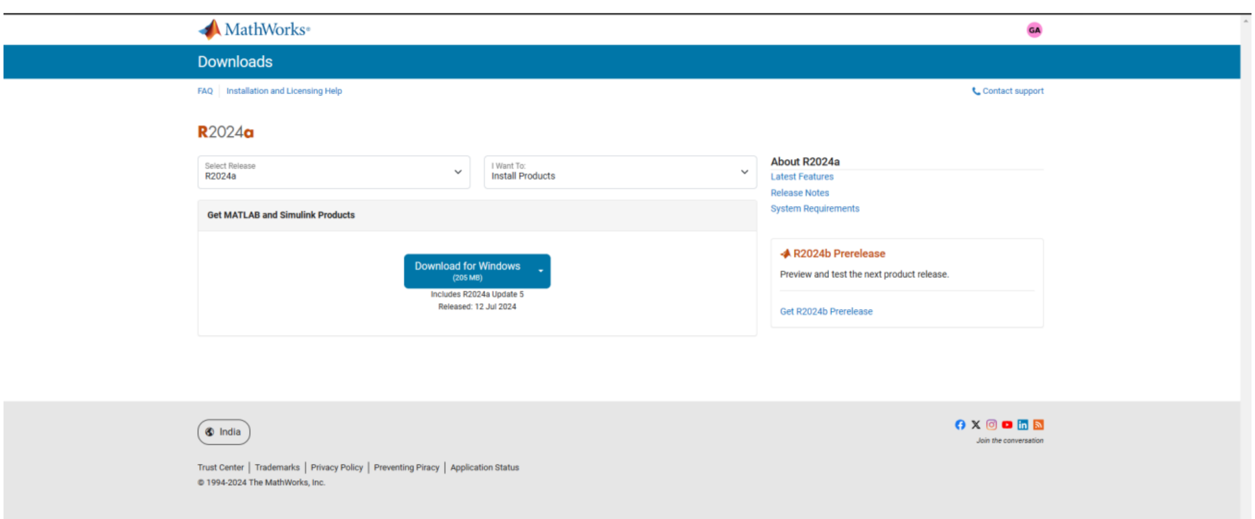
S.No	PRACTICAL	DATE	SIGN
1.	Write the steps to install MATLAB with flowchart.		
	Write the program in MATLAB to implement network of 10 nodes.		
	Implement a point-to-point network consisting of 10 nodes using MATLAB with duplex links between them. Initiate a communication between these nodes. Set the queue size, vary the bandwidth and find the number of packets dropped. Finally plot the graph showing the performance of this network in terms of number of packets dropped with varying bandwidth.		
	Implement FDMA, TDMA & CDMA using MATLAB and show the results using graph for 10 users using clustering techniques.		
	Implement GSM using MATLAB		
.	Implement GPRS using MAC layer in MATLAB		
.	Implement LTE using MATLAB		
.	Implement snooping and analysing the traffic using Wireshark		

1. Write the steps to install MATLAB with flowchart:

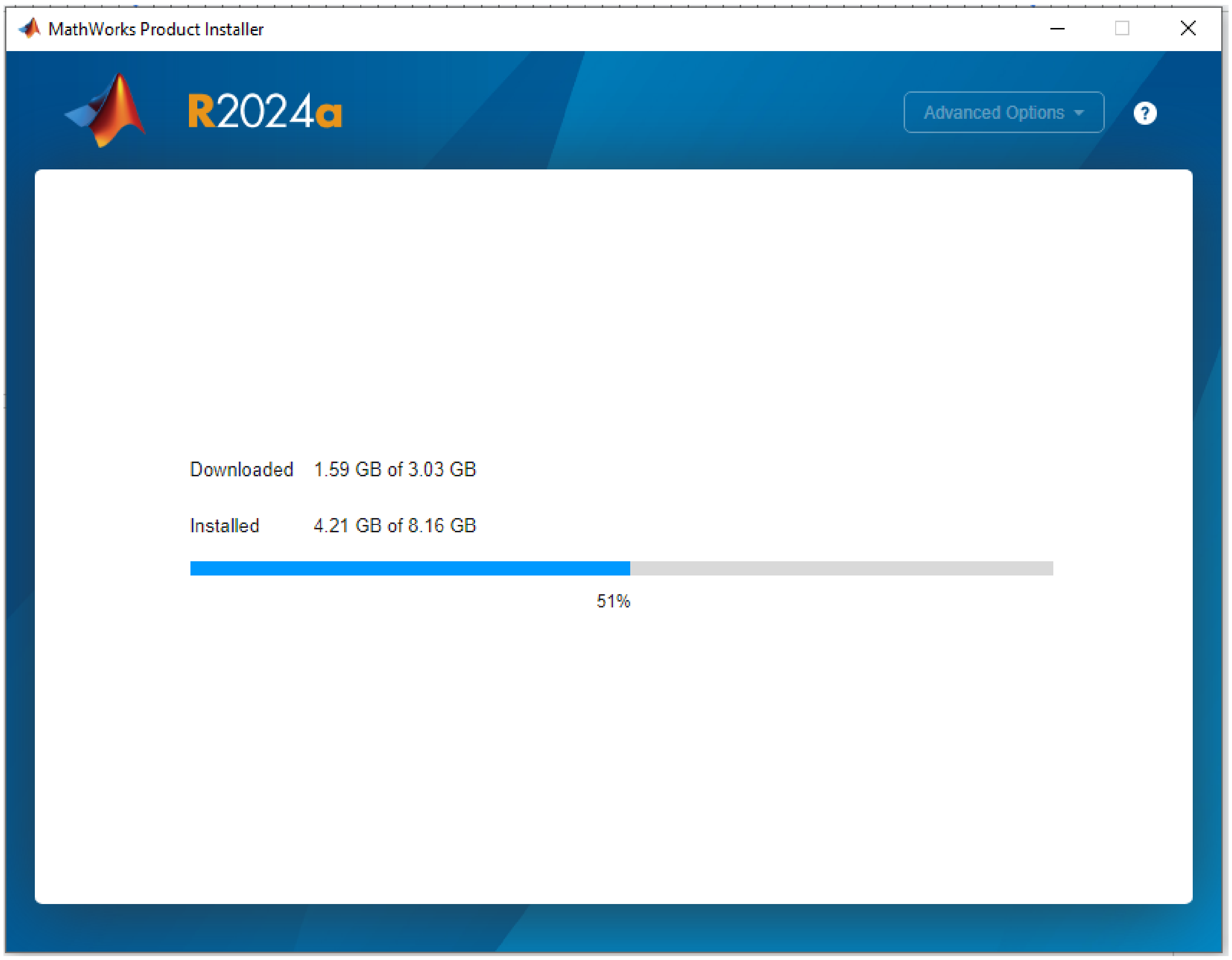
Step 1: To download the MATLAB installer, you must have a MathWorks Account



Step 2: From MathWorks Downloads, select a MATLAB release and download the installer.



Step 3: Double-click the downloaded installer and follow the prompts to complete the installation.



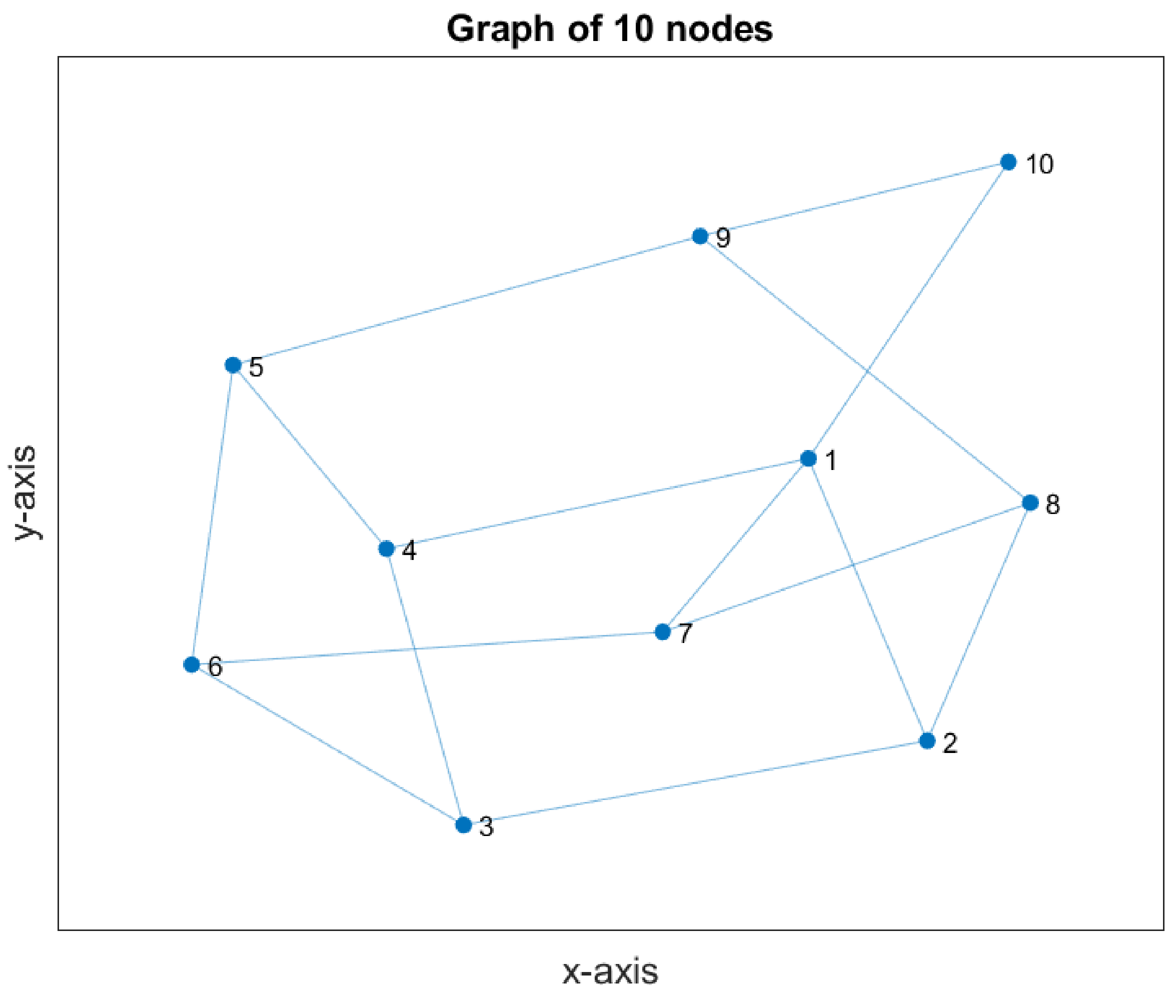
2.Write the program in MATLAB to implement a network of 10 nodes:

CODE:

```
%implement a network of 10 nodes (static -> just show
the connections)

clc;
clear;
%no of nodes
n=10;
G=graph();
G=addnode(G,n);
%no of edges
m=input("Enter number of edges: ");
for i=1:m
   u=input("Enter the first node of the edge: ");
   v=input("Enter the second node of the edge: ");
   G=addedge(G,u,v);
end
plot(G);
title("Graph of 10 nodes");
xlabel("x-axis");
ylabel("y-axis");
```

OUTPUT:



3. Implement a point-to-point network of 10 nodes using MATLAB with duplex links between them. Initiate communication between these nodes. Set the queue size, vary the bandwidth, and find the number of packets dropped. Finally, plot the graph showing the performance of this network in terms of the number of packets dropped with varying bandwidths.

CODE:

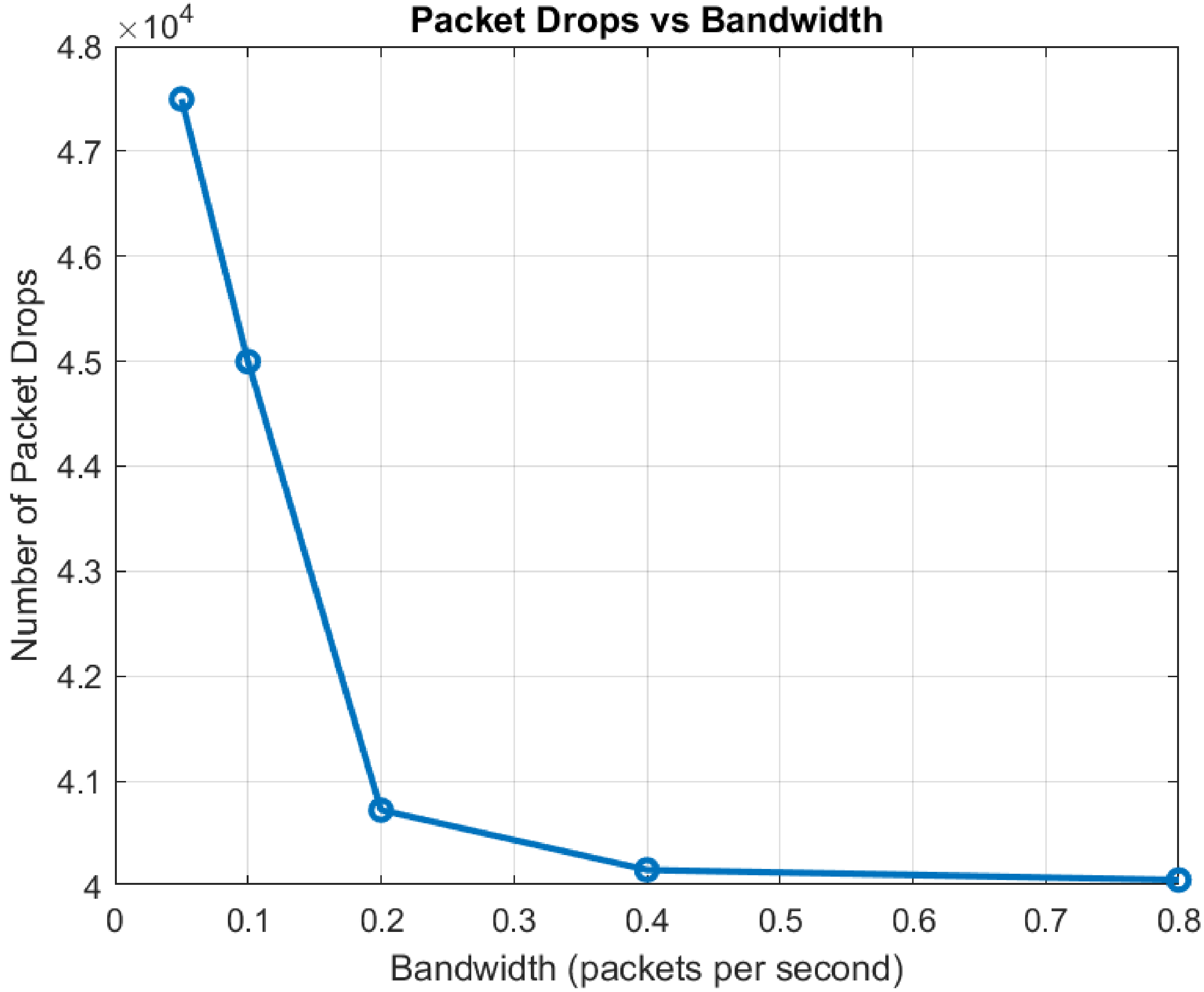
```
clc;
clear;
%parameters
n=10;
queueSize=5; %smaller the queue size, more packets are dropped
bandwidths=[0.05,0.1,0.2,0.4,0.8]; %in terms of packets sent per second
numPackets=50000; %traffic
packetDrops=zeros(size(bandwidths)); %tracks no. of packets dropped for each bandwidth we have
%creating a fully connected duplex graph
G=graph();
G=addnode(G,n);
for i=1:n
   for j=i+1:n
       G=addedge(G,i,j);
       G=addedge(G,j,i);
   end
end
```

```
%simulate packet transmission with different bandwidths
for b=1:length(bandwidths)
   bw=bandwidths(b);
   curr=0; %Current packets dropped
   queue=0; %global queue variable which will store
number of packets to be processed

   for p=1:numPackets
       % Randomly select two nodes for packet
transmission
       u=randi(n);
       v=randi(n);
       while u==v %they have to be different nodes
           v=randi(n);
       end

       if queue<queueSize && rand<0.2 %queue is not
full and only 20% chance of queueing due to delay
           queue=queue+1;%queue the packet
       else
           curr=curr+1;%packet is not queued-> dropped
       end
       if bw>queue
           queue=0; %all packets are transmitted
       else
           queue=queue-bw; %only bw packets are
transmitted
       end
   end
   packetDrops(b)=curr;
end
% Plot packet drops vs bandwidth
figure;
plot(bandwidths, packetDrops, '-o', 'LineWidth', 2);
title('Packet Drops vs Bandwidth');
xlabel('Bandwidth (packets per second)');
ylabel('Number of Packet Drops');
grid on;
```

OUTPUT:



Packet Drops vs Bandwidth
×10^4
4.8
4.7
4.6
4.5
4.4
4.3
4.2
4.1
4
Number of Packet Drops
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
Bandwidth (packets per second)

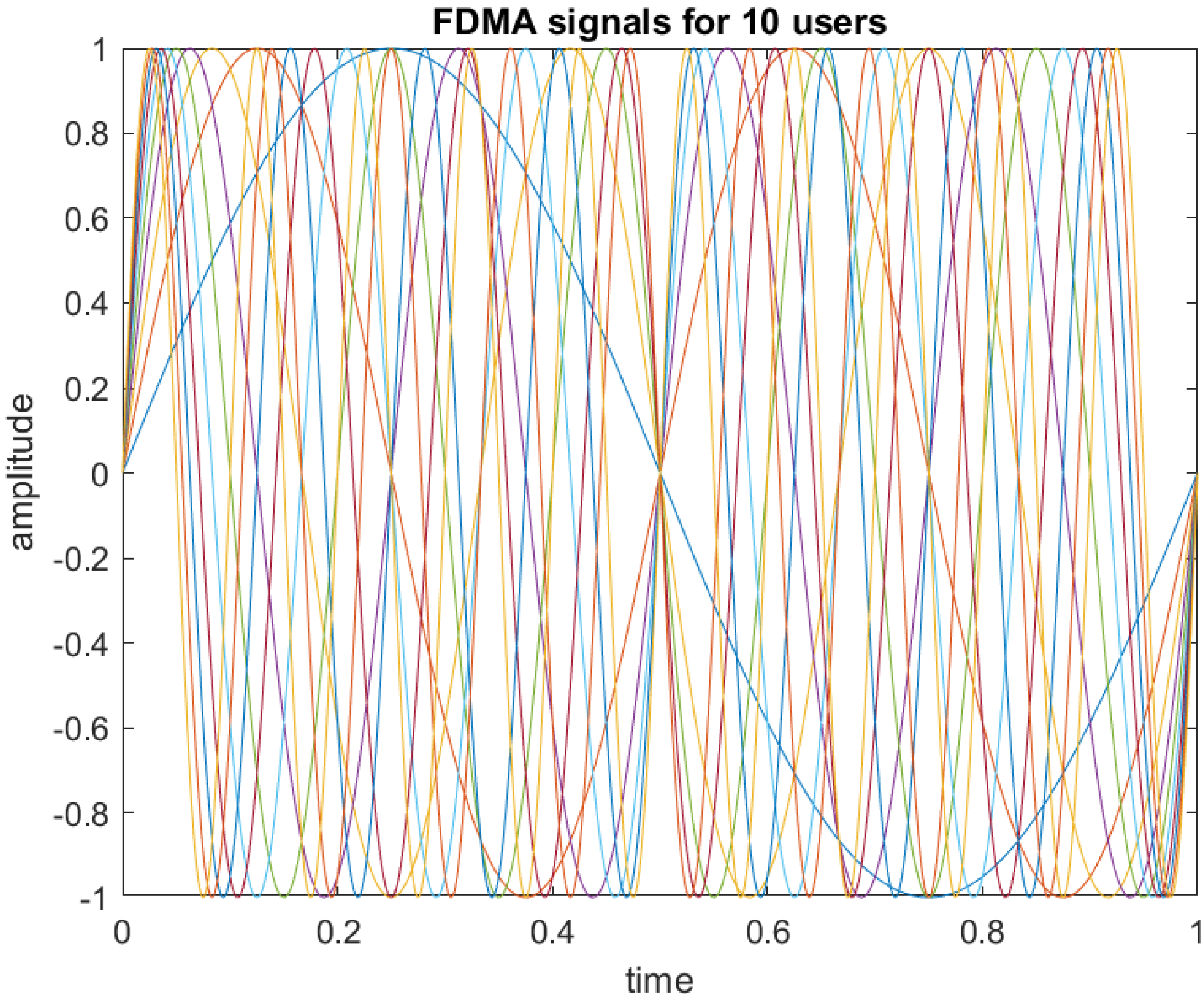
4.Implement FDMA, TDMA & CDMA using MATLAB and show the results using graph for 10 users using clustering techniques.

FDMA->

CODE:

```
%FDMA
clc;
clear;
num_users=10;
frequency_bands = linspace(1,10,num_users); %evenly
spaced values from 1 to 10 and total of num_users of
such
%generate signals
t=0:0.001:1; %time vector
signals_fdma=zeros(num_users,length(t));
for i=1:num_users
   signals_fdma(i,:) = sin(2*pi*frequency_bands(i)*t);
end
%plot signals
figure;
for i=1:num_users
   plot(t,signals_fdma(i,:));
   hold on;
end
title('FDMA signals for 10 users');
xlabel('time');
ylabel('amplitude');
```

OUTPUT:



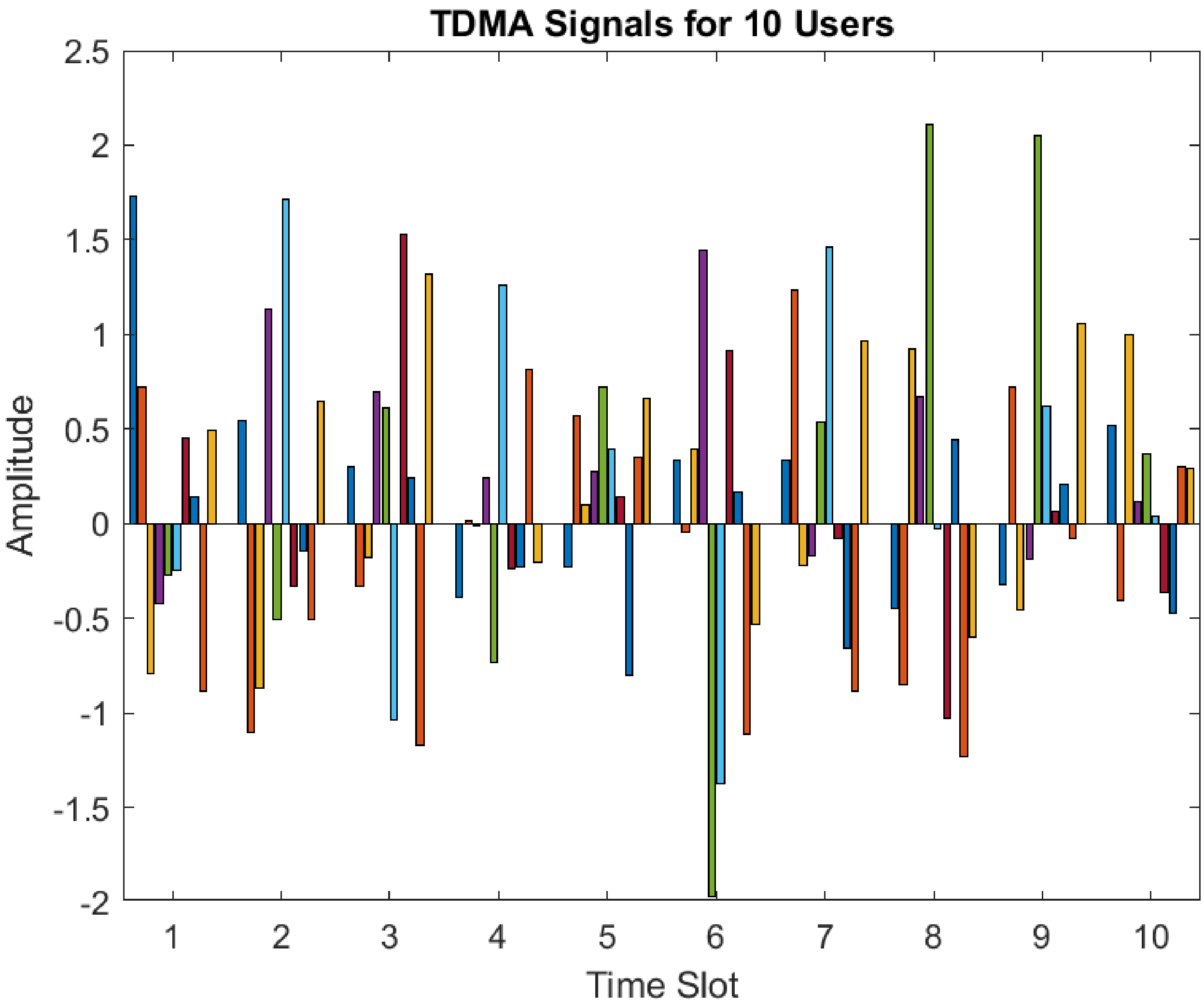
FDMA signals for 10 users
amplitude
time
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
0.2
0.4
0.6
0.8
1

TDMA->

CODE:

```
%TDMA
clc;
clear;
%parameters
num_users = 10;
time_slots = 10;
signals_tdma = zeros(num_users, time_slots);
% Generate signals for each user in different time
slots
for i = 1:num_users
   signals_tdma(i, :) = randn(1, time_slots);
end
% Plot signals for TDMA
figure;
bar(signals_tdma);
title('TDMA Signals for 10 Users');
xlabel('Time Slot');
ylabel('Amplitude');
```

OUTPUT:



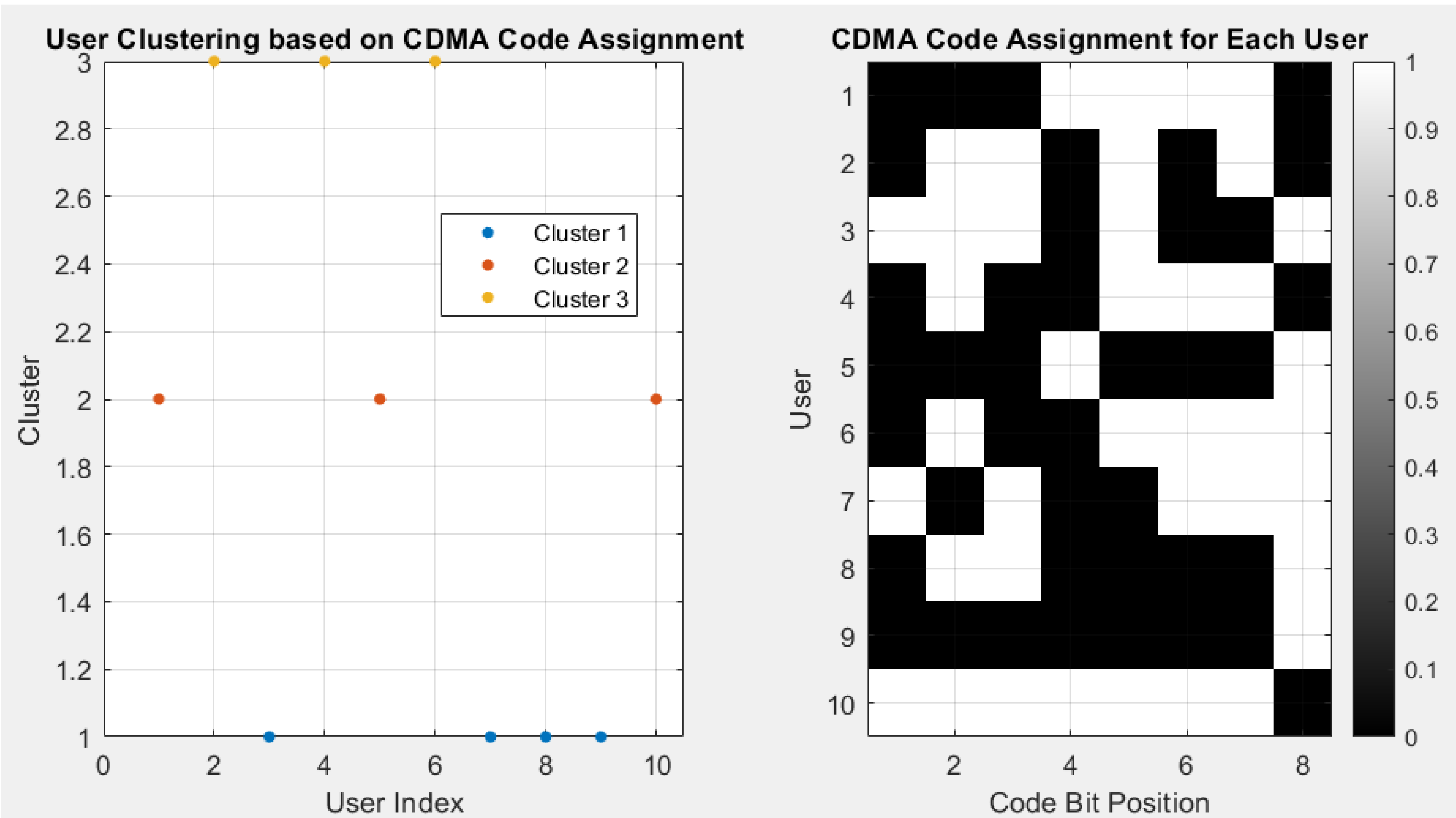
TDMA Signals for 10 Users
Amplitude
Time Slot
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
1
2
3
4
5
6
7
8
9
10

CDMA->

CODE:

```
%CDMA
clc;
clear;
% Parameters
numUsers = 10;
numClusters = 3; % Number of clusters
codeLength = 8; % Length of each code (arbitrary units)
% Generate random binary codes for each user (0s and
1s)
codes = randi([0, 1], numUsers, codeLength);
% Apply K-Means Clustering to group users based on
their codes
[idx, clusterCenters] = kmeans(codes, numClusters);
% Plot the clustering results based on code assignment
figure;
subplot(1, 2, 1);
gscatter(1:numUsers, idx, idx);
title('User Clustering based on CDMA Code Assignment');
xlabel('User Index');
ylabel('Cluster');
legend('Cluster 1', 'Cluster 2', 'Cluster 3');
grid on;
% Plot the CDMA codes for each user
subplot(1, 2, 2);
imagesc(codes);
colormap(gray);
title('CDMA Code Assignment for Each User');
xlabel('Code Bit Position');
ylabel('User');
colorbar;
grid on;
```

OUTPUT:



User Clustering based on CDMA Code Assignment
Cluster
User Index
Cluster 1
Cluster 2
Cluster 3
CDMA Code Assignment for Each User
User
Code Bit Position

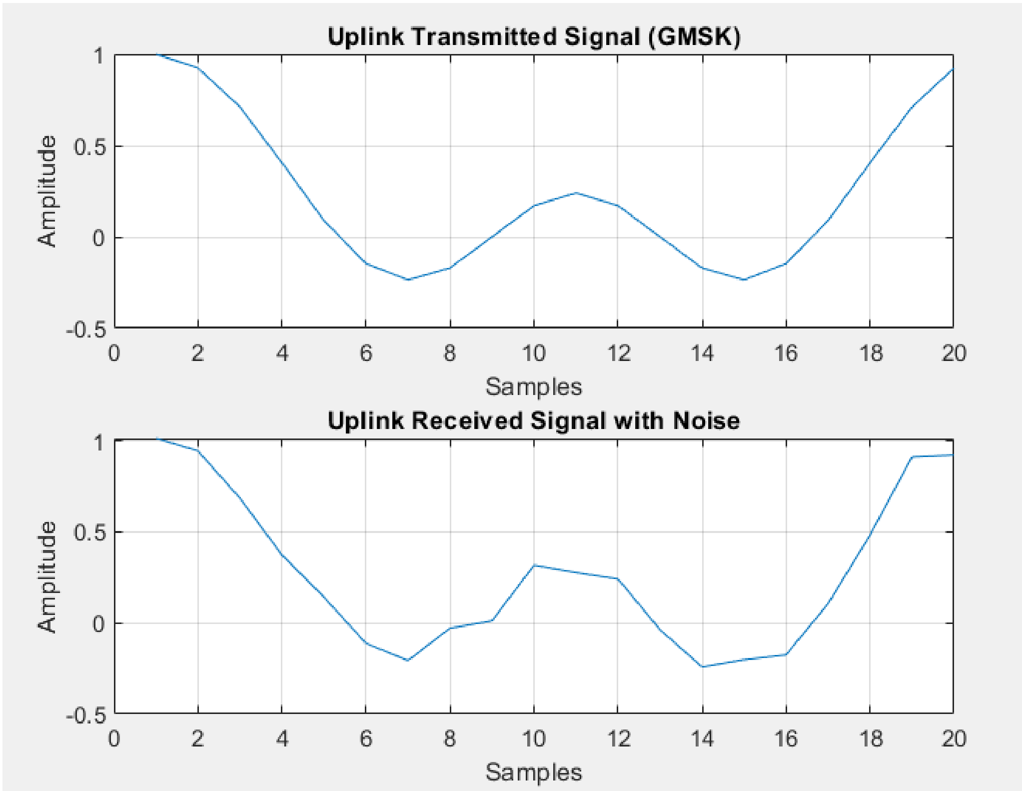
5. Implement GSM using MATLAB:

CODE:

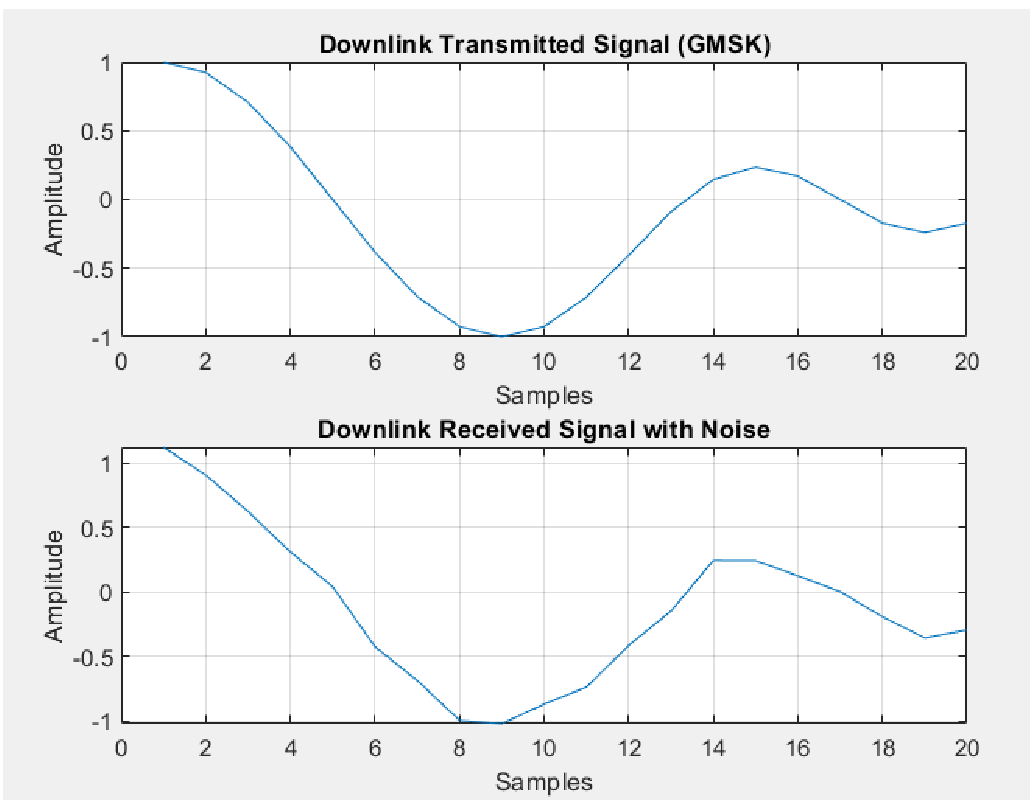
```
clc;
clear;
% Parameters
Nbits = 5;         % Number of bits to transmit
% Generate random binary data for uplink
dataUplink = randi([0 1], 1, Nbits); % Random binary
data
% Create a GMSK modulator
gmskModulator = comm.GMSKModulator('BitInput', true,
'SamplesPerSymbol', 4);
% Modulate the uplink signal
modulatedUplink = gmskModulator(dataUplink');
% Transmit uplink signal (simple channel model)
txUplink = awgn(modulatedUplink, 20); % Add white
Gaussian noise
% Create a GMSK demodulator
gmskDemodulator = comm.GMSKDemodulator('BitOutput',
true, 'SamplesPerSymbol', 4);
% Demodulate uplink signal
receivedUplink = gmskDemodulator(txUplink);
% Plot uplink signals
figure;
subplot(2,1,1);
plot(real(modulatedUplink));
title('Uplink Transmitted Signal (GMSK)');
xlabel('Samples');
ylabel('Amplitude');
grid on;
subplot(2,1,2);
plot(real(txUplink));
title('Uplink Received Signal with Noise');
xlabel('Samples');
ylabel('Amplitude');
grid on;
%% Downlink Transmission
% Generate random binary data for downlink
dataDownlink = randi([0 1], 1, Nbits); % Random binary
data
```

```matlab
% Modulate the downlink signal
modulatedDownlink = gmskModulator(dataDownlink');
% Transmit downlink signal (simple channel model)
txDownlink = awgn(modulatedDownlink, 20); % Add white
Gaussian noise
% Demodulate downlink signal
receivedDownlink = gmskDemodulator(txDownlink);
% Plot downlink signals
figure;
subplot(2,1,1);
plot(real(modulatedDownlink));
title('Downlink Transmitted Signal (GMSK)');
xlabel('Samples');
ylabel('Amplitude');
grid on;
subplot(2,1,2);
plot(real(txDownlink));
title('Downlink Received Signal with Noise');
xlabel('Samples');
ylabel('Amplitude');
grid on;
```

OUTPUT:



Uplink Transmitted Signal (GMSK)
Amplitude
Samples
Uplink Received Signal with Noise
Amplitude
Samples



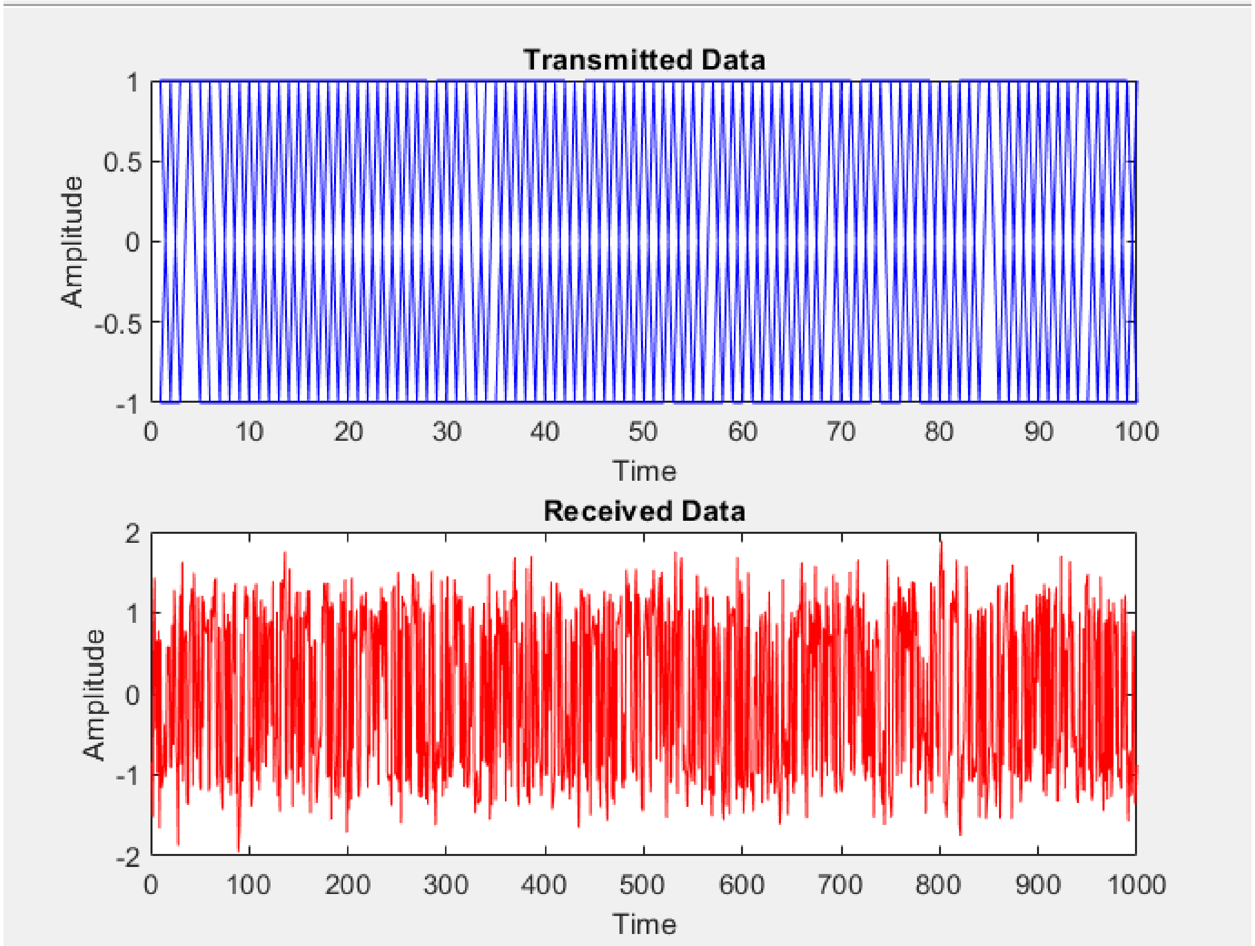
Downlink Transmitted Signal (GMSK)
Amplitude
Samples
Downlink Received Signal with Noise
Amplitude
Samples

6. Implement GPRS using MAC layer in MATLAB:

CODE:

```
clc;
clear;
% Constants for the MAC layer
frame_size = 100; % Frame size in bits
num_frames = 10; % Number of frames
% Generate random data
data = randi([0, 1], 1, frame_size * num_frames);
% MAC Layer: Frame segmentation and interleaving
frames = reshape(data, frame_size, num_frames); %
Divide data into frames
interleaved_frames = frames(:, randperm(num_frames)); %
Interleave frames
% Transmitter
% Use BPSK modulation(any modulation can be used)
modulated_data = 2 * interleaved_frames - 1; % BPSK
modulation
% Display the transmitted data
figure;
subplot(2, 1, 1);
plot(modulated_data, 'b');
title('Transmitted Data');
xlabel('Time');
ylabel('Amplitude');
% Simulate received signal
received_signal = awgn(modulated_data, 10); % Add AWGN
% MAC Layer: De-interleaving and reassembly
deinterleaved_frames = reshape(received_signal,
frame_size, num_frames);
reconstructed_data = deinterleaved_frames(:,
randperm(num_frames));
received_data = reconstructed_data(:);
% Display the received data
subplot(2, 1, 2);
plot(received_data, 'r');
title('Received Data');
xlabel('Time');
ylabel('Amplitude');
```

OUTPUT:



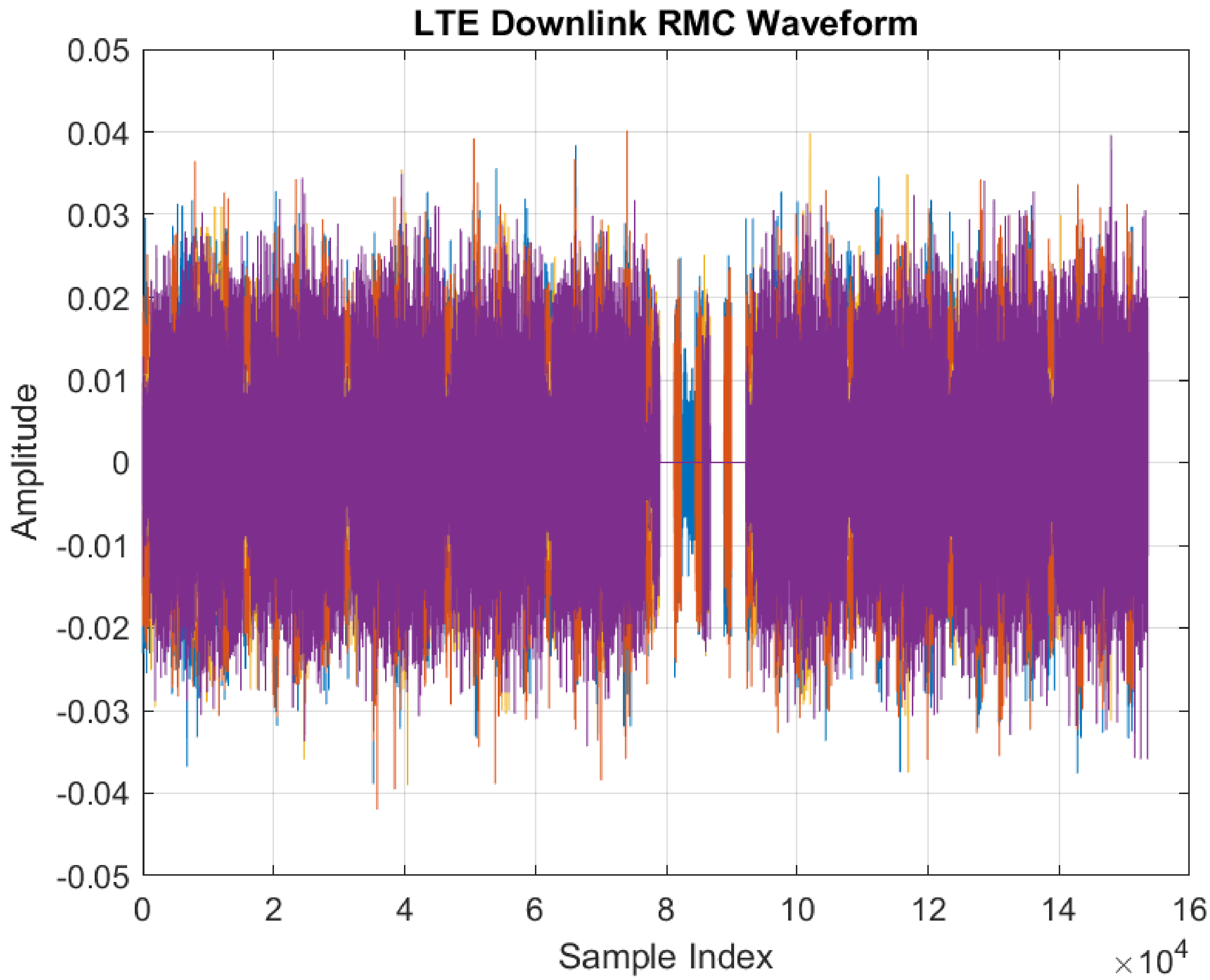
Transmitted Data
Amplitude
1
0.5
0
-0.5
-1
0
10
20
30
40
50
60
70
80
90
100
Time
Received Data
Amplitude
2
1
0
-1
-2
0
100
200
300
400
500
600
700
800
900
1000
Time

7. Implement LTE using MATLAB:

CODE:

```
clc;
clear;
% Set up the LTE downlink reference measurement channel
(RMC) configuration
rmc = lteRMCDL('R.13');
% Generate random data for transmission based on the
transport block sizes
Data = randi([0 1], 1, sum(rmc.PDSCH.TrBlkSizes));
% Generate LTE-compliant waveform, resource grid, and
output configuration
[waveform, txgrid, RMCcfgOut] = lteRMCDLTool(rmc,
Data);
% Plot the generated waveform in the time domain
figure;
plot(real(waveform));
title('LTE Downlink RMC Waveform');
xlabel('Sample Index');
ylabel('Amplitude');
grid on;
```

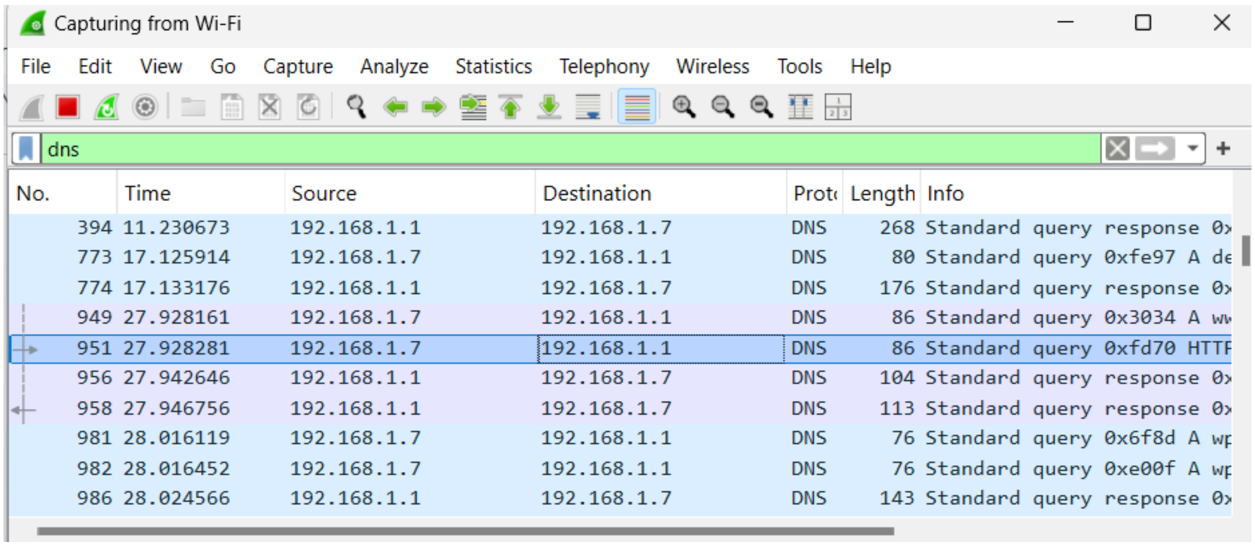
OUTPUT:



LTE Downlink RMC Waveform
Amplitude
0.05
0.04
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.05
0
2
4
6
8
10
12
14
16
Sample Index
$\times 10^4$

8.Implement snooping and analysing the traffic using Wireshark:

Capturing traffic from Wi-fi and applying DNS display filter-



No.	Time	Source	Destination	Prot	Length	Info
394	11.230673	192.168.1.1	192.168.1.7	DNS	268	Standard query response 0x
773	17.125914	192.168.1.7	192.168.1.1	DNS	80	Standard query 0xfe97 A de
774	17.133176	192.168.1.1	192.168.1.7	DNS	176	Standard query response 0x
949	27.928161	192.168.1.7	192.168.1.1	DNS	86	Standard query 0x3034 A ww
951	27.928281	192.168.1.7	192.168.1.1	DNS	86	Standard query 0xfd70 HTTP
956	27.942646	192.168.1.1	192.168.1.7	DNS	104	Standard query response 0x
958	27.946756	192.168.1.1	192.168.1.7	DNS	113	Standard query response 0x
981	28.016119	192.168.1.7	192.168.1.1	DNS	76	Standard query 0x6f8d A wp
982	28.016452	192.168.1.7	192.168.1.1	DNS	76	Standard query 0xe00f A wp
986	28.024566	192.168.1.1	192.168.1.7	DNS	143	Standard query response 0x

Analysing the queries-

```
> [2 Reassembled TCP Segments (34 bytes): #950(2), #951(32)]
v Domain Name System (query)
    Length: 32
    Transaction ID: 0xfd70
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
    v www.google.com: type HTTPS, class IN
          Name: www.google.com
          [Name Length: 14]
          [Label Count: 3]
          Type: HTTPS (65) (HTTPS Specific Service Endpoints)
          Class: IN (0x0001)
    [Response In: 958]
```

```
Domain Name System (query)
    Transaction ID: 0xce53
  > Flags: 0x0100 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  v Queries
      v leetcode.com: type HTTPS, class IN
            Name: leetcode.com
            [Name Length: 12]
            [Label Count: 2]
            Type: HTTPS (65) (HTTPS Specific Service Endpoints)
            Class: IN (0x0001)
    [Response In: 16768]
```

By capturing the packets, one can analyse traffic on a shared network, snooping on what kind of services are being accessed by the DNS requests that are made by which device and all this can be done in real time.