

WIRELESS COMMUNICATION

ETEC 463

Faculty Name: Mr. Saurabh Rastogi

Student Name: Syeda Reeha Quasar

Roll No: 14114802719

Batch: 7C7



Maharaja Agrasen Institute of Technology,
PSP Area, Sector 22, Rohini, Delhi – 110085

INDEX

Exp. No.	Experiment Name	Date of Performance	Date of Checking	Marks	Signature
1.	Introduction to Scilab.				
2.	Write a program in Scilab to Calculate Frequency Reuse Distance ,Co-Channel Interference reduction factor, Cellular System Capacity, S/I Ratio for a given variables.				
3.	Write a program in Scilab to Calculate maximum traffic intensity and maximum number of users accomodated in Erlang B and Erlang C system for given no of channels.				
4.	Write a Program in Scilab to calculate Bit Error rate performance of BPSK modulated signal over only AWGN channel and AWGN and Rayleigh channel both.				
5.	Program in Scilab to Generate Walsh Codes and then spread the user information using it.				
6.	Write a program in scilab to Generate PN Sequence for CDMA Systems.				
7.	Write a Program in NS3 to connect WIFI TO BUS (CSMA) Network.				
8.	Write a Program in NS3 to create WIFI Network in SIMPLE INFRASTRUCTURE MODE (of nodes)				
9.	Write a Program in NS3 to Create a wireless mobile ad-hoc network between three nodes.				

Experiment – 1

AIM –

Introduction to Scilab.

Overview:

Scilab is a programming language associated with a rich collection of numerical algorithms covering many aspects of scientific computing problems. From the software point of view, Scilab is an interpreted language. This generally allows to get faster development processes, because the user directly accesses to a high-level language, with a rich set of features provided by the library. The Scilab language is meant to be extended so that user-defined data types can be defined with possibly overloaded operations.

Scilab users can develop their own module so that they can solve their particular problems. The Scilab language allows to dynamically compile and link other languages such as Fortran and C: this way, external libraries can be used as if they were a part of Scilab built-in features.

Scilab also interfaces LabVIEW, a platform and development environment for a visual programming language from National Instruments. From the license point of view, Scilab is a free software in the sense that the user does not pay for it and Scilab is an open source software, provided under the Cecill license. The software is distributed with source code, so that the user has an access to Scilab most internal aspects. Most of the time, the user downloads and installs, a binary version of Scilab since the Scilab consortium provides Windows, Linux and Mac OS executable versions. An online help is provided in many local languages. From a scientific point of view, Scilab comes with many features. At the very beginning of Scilab, features were focused on linear algebra. But, rapidly, the number of features extended to cover many areas of scientific computing. The following is a short list of its capabilities:

- Linear algebra, sparse matrices,
- Polynomials and rational functions,
- Interpolation, approximation,
- Linear, quadratic and non linear optimization,
- Ordinary Differential Equation solver and Differential Algebraic Equations solver,
- Classic and robust control, Linear Matrix Inequality optimization,
- Differentiable and non-differentiable optimization,
- Signal processing,
- Statistics.

Scilab provides many graphics features, including a set of plotting functions, which allow to create 2D and 3D plots as well as user interfaces. The Xcos environment provides an hybrid dynamic systems modeler and simulator.

Installing Scilab under Mac OS:

Under Mac OS, the binary versions are available from Scilab website as a .dmg file. This binary works for Mac OS versions starting from version 10.5. It uses the Mac OS installer, which provides a classical installation process. Scilab is not available on Power PC systems. Scilab version 5.2 for Mac OS comes with a Tcl / Tk library which is disabled for technical reasons. As a consequence, there are some small limitations on the use of Scilab on this platform. For example, the Scilab / Tcl interface (TclSci), the graphic editor and the variable editor are not working. These features will be rewritten in Java in future versions of Scilab and these limitations will disappear. Still, using Scilab on Mac OS system is easy, and uses the shortcuts which are familiar to users of this platform. For example, the console and the editor use the Cmd key (Apple key) which is found on Mac keyboards. Moreover, there is no right-click on this platform. Instead, Scilab is sensitive to the Control-Click keyboard event. For now, Scilab comes on Mac OS with a linear algebra library which is optimized and guarantees portability. Under Mac OS, Scilab does not come with a binary version of ATLAS, so that linear algebra is a little slower for that platform.

The Console:

The console The first way is to use Scilab interactively, by typing commands in the console, analyzing Scilab result, continuing this process until the final result is computed. This document is designed so that the Scilab examples which are printed here can be copied into the console. The goal is that the reader can experiment by himself Scilab behavior. This is indeed a good way of understanding the behavior of the program and, most of the time, it allows a quick and smooth way of performing the desired computation. In the following example, the function `disp` is used in interactive mode to print out the string "Hello World !".

```
-->s=" Hello World ! " s = Hello World !  
--> disp ( s) Hello World !
```

In the previous session, we did not type the characters "-->" which is the prompt, and which is managed by Scilab. We only type the statement `s="Hello World!"` with our keyboard and then hit the key. Scilab answer is `s =` and `Hello World!`. Then we type `disp(s)` and Scilab answer is `Hello World!`.

Creating real variables:

Scilab is an interpreted language, which implies that there is no need to declare a variable before using it. Variables are created at the moment where they are first set. In the following example, we create and set the real variable `x` to 1 and perform a multiplication on this variable. In Scilab, the "=" operator means that we want to set the variable on the left hand side to the value associated to the right hand side (it is not the comparison operator, which syntax is associated to the "==" operator).

```
-->x=1 x = 1.  
-->x = x * 2 x = 2.
```

The value of the variable is displayed each time a statement is executed. That behavior can be suppressed if the line ends with the semicolon ";" character, as in the following example.

```
-->y=1;  
-->y=y*2;
```

Variable Name:

Variable names may be as long as the user wants, but only the first 24 characters are taken into account in Scilab. For consistency, we should consider only variable names which are not made of more than 24 characters. All ASCII letters from "a" to "z", from "A" to "Z" and from "0" to "9" are allowed, with the additional letters "%", "_", "#", "!", "\$", "?".

Comments and continuation lines:

Any line which begins with two slashes "//" is considered by Scilab as a comment and is ignored. There is no possibility to comment out a block of lines, such as with the "/* ... */" comments in the C language. When an executable statement is too long to be written on a single line, the second line and above are called continuation lines. In Scilab, any line which ends with two dots is considered to be the start of a new continuation line. In the following session, we give examples of Scilab comments and continuation lines.

Strings:

Strings can be stored in variables, provided that they are delimited by double quotes ". ". The concatenation operation is available from the "+" operator. In the following Scilab session, we define two strings and then concatenate them with the "+" operator.

```
-->x = " foo " x = foo 20
```

```
-->y=" bar "y = bar
```

```
-->x+y  
ans = foobar
```

There are many functions which allow to process strings, including regular expressions. We will not give further details about this topic in this document.

Matrices:

In the Scilab language, matrices play a central role. In this section, we introduce Scilab matrices and present how to create and query matrices. We also analyze how to access to elements of a matrix, either element by element, or by higher level operations.

Create a matrix of real values

There is a simple and efficient syntax to create a matrix with given values. The following is the list of symbols used to define a matrix:

- square brackets "[" and "]" mark the beginning and the end of the matrix,
- commas "," separate the values on different columns,
- semicolons ";" separate the values of different rows.

```
Scilab branch-6.1 Console ? ? X  
  
--> A = [1,2,3;3,4,5]  
A =  
  
1.  2.  3.  
3.  4.  5.  
  
--> B = [1,4,6;7,8,4]  
B =  
  
1.  4.  6.  
7.  8.  4.  
  
--> C = A+B  
C =  
  
2.  6.  9.  
10. 12. 9.  
  
--> D = [1,2,3;4,5,6;7,8,9]  
D =  
  
1.  2.  3.  
4.  5.  6.  
7.  8.  9.  
  
--> rank(D)  
ans =  
  
2.  
  
-->
```

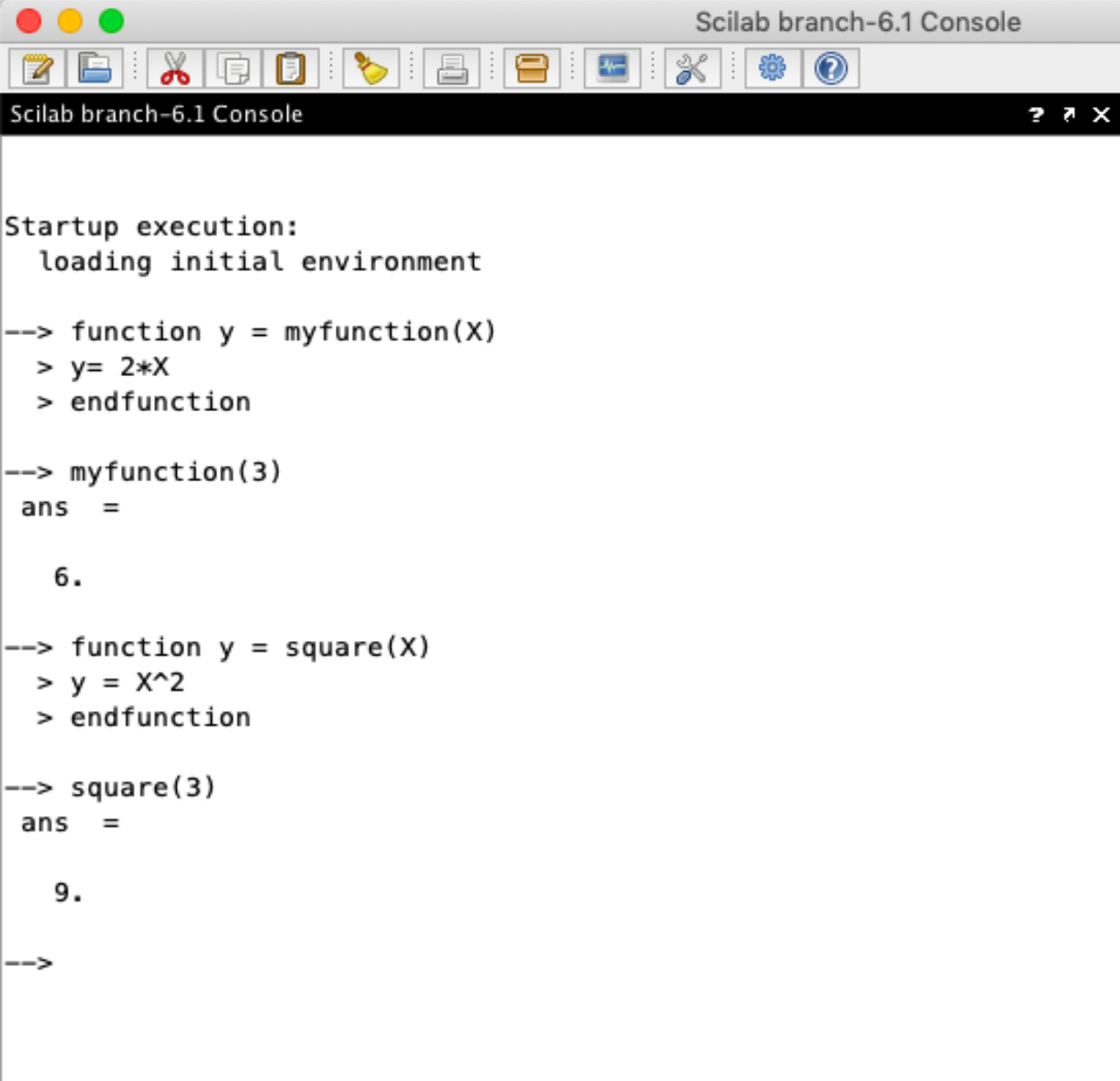
Functions

To define a new function, we use the `function` and `endfunction` Scilab keywords. In the following example, we define the function `myfunction`, which takes the input argument `x`, multiplies it by 2, and returns the value in the output argument `y`.

```
-->function y = myfunction ( x )
```

```
>y = 2 * x
```

```
> endfunction
```

The image shows a screenshot of the Scilab branch-6.1 Console window. The window has a title bar with three colored buttons (red, yellow, green) and the text "Scilab branch-6.1 Console". Below the title bar is a toolbar with various icons for file operations (like save, open, print) and editing (like copy, paste, undo, redo). The main area of the console is a text editor showing the following code and output:

```
Startup execution:
loading initial environment

--> function y = myfunction(X)
> y= 2*X
> endfunction

--> myfunction(3)
ans  =

    6.

--> function y = square(X)
> y = X^2
> endfunction

--> square(3)
ans  =

    9.

-->
```

Plotting:

Producing plots and graphics is a very common task for analysing data and creating reports. Scilab offers many ways to create and customize various types of plots and charts. In this section, we present how to create 2D plots and contour plots. Then we customize the title and the legend of our graphics. We finally export the plots so that we can use it in a report.

Scilab can produce many types of 2D and 3D plots. The following is a short list of

several common charts that Scilab can create:

- x-y plots: plot,
- contour plots: contour,
- 3D plots: surf,
- histograms: histplot,
- bar charts:

```
-->function f = myquadratic ( x )  
> f = x ^2  
> endfunction  
-->xdata = linspace ( 1 , 10 , 50 );  
-->ydata = myquadratic ( xdata );  
-->plot ( xdata , ydata )
```



Startup execution:

loading initial environment

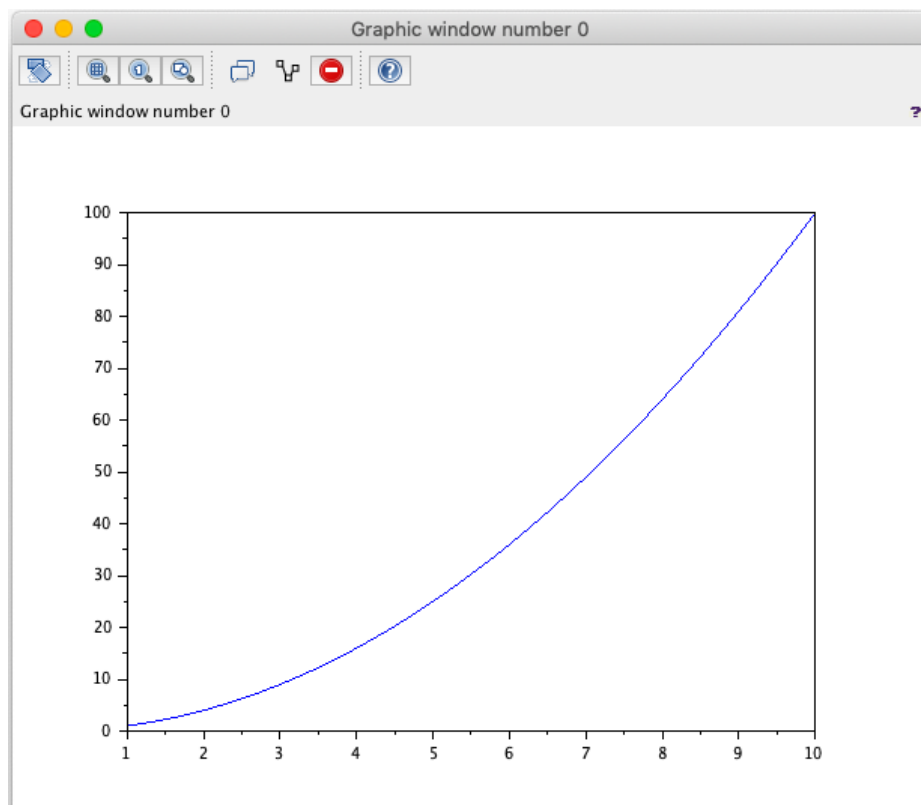
```
--> function f = myquadratic ( x )  
  > f = x ^2  
  > endfunction
```

```
--> xdata = linspace ( 1 , 10 , 50 );
```

```
--> ydata = myquadratic ( xdata );
```

```
--> plot ( xdata , ydata )
```

```
-->
```



Experiment – 2

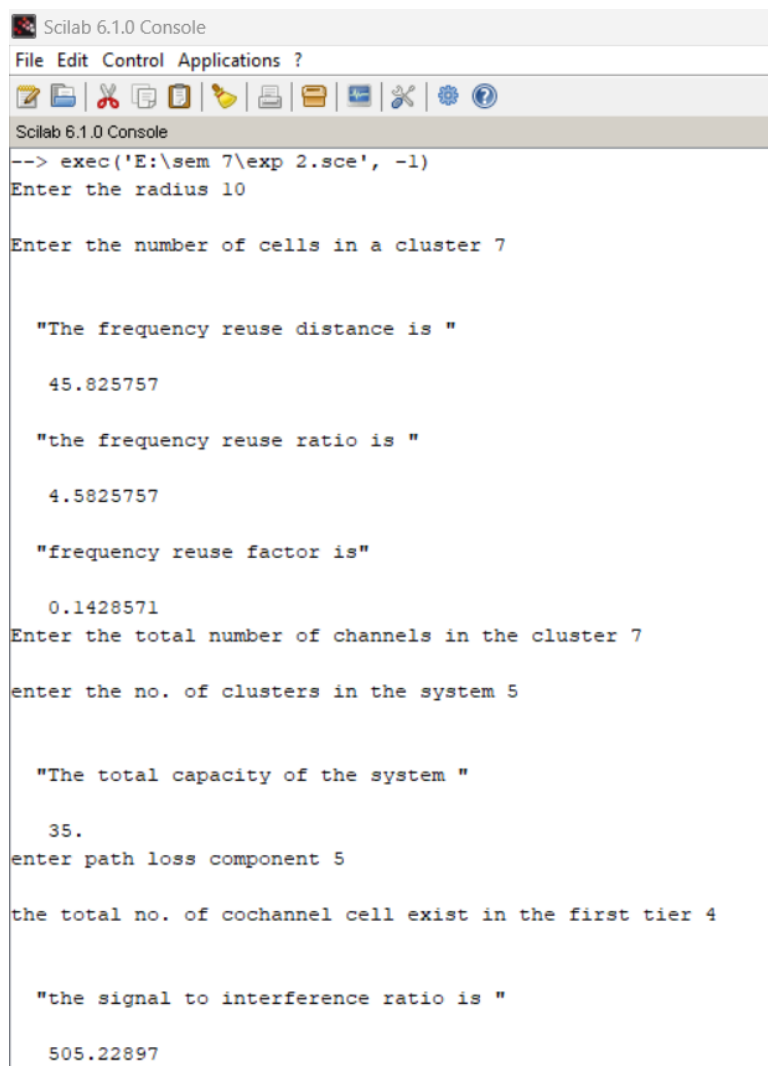
AIM –

Write a program in Scilab to Calculate Frequency Reuse Distance ,Co- Channel Interference reduction factor, Cellular System Capacity, S/I Ratio for a given variables.

Code:

```
r = input("Enter the radius ")
n = input("Enter the number of cells in a cluster ")
d = r*sqrt(3*n)
q = d/r
disp("The frequency reuse distance is ",d)
disp("the frequency reuse ratio is ",q)
z=1/n
disp('frequency reuse factor is',z)
p = input('Enter the total number of channels in the cluster ')
n = input('enter the no. of clusters in the system ')
c = p*n
disp('The total capacity of the system ',c)
x = input('enter path loss component ')
o = input('the total no. of cochannel cell exist in the first tier ')
w = (q^x)/o
disp('the signal to interference ratio is ', w)
```

Output:



```
Scilab 6.1.0 Console
File Edit Control Applications ?
--> exec('E:\sem 7\exp 2.sce', -1)
Enter the radius 10

Enter the number of cells in a cluster 7

"The frequency reuse distance is "

45.825757

"the frequency reuse ratio is "

4.5825757

"frequency reuse factor is"

0.1428571
Enter the total number of channels in the cluster 7

enter the no. of clusters in the system 5

"The total capacity of the system "

35.
enter path loss component 5

the total no. of cochannel cell exist in the first tier 4

"the signal to interference ratio is "

505.22897
```


Enter the blocking probability = 0.1

Enter the call rate = 3/60

Enter the call duration = 2

Enter no. of Channels = 50

"Blocking probability entered by user ="

0.1

"Maximum Traffic Intensity ="

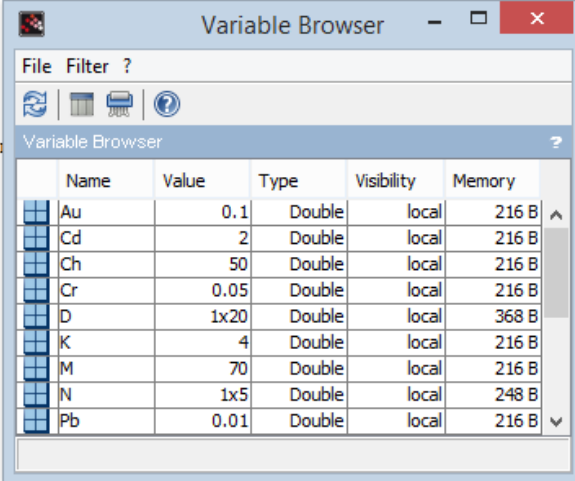
50.

"Number of Users ="

500.

exec: Wrong number of output arguments

Enter the call duration =



Variable Browser

	Name	Value	Type	Visibility	Memory
<input checked="" type="checkbox"/>	Au	0.1	Double	local	216 B
<input checked="" type="checkbox"/>	Cd	2	Double	local	216 B
<input checked="" type="checkbox"/>	Ch	50	Double	local	216 B
<input checked="" type="checkbox"/>	Cr	0.05	Double	local	216 B
<input checked="" type="checkbox"/>	D	1x20	Double	local	368 B
<input checked="" type="checkbox"/>	K	4	Double	local	216 B
<input checked="" type="checkbox"/>	M	70	Double	local	216 B
<input checked="" type="checkbox"/>	N	1x5	Double	local	248 B
<input checked="" type="checkbox"/>	Pb	0.01	Double	local	216 B

Enter the blocking probability = 0.01

Enter the call rate = 3/60

Enter the call duration = 2

Enter no. of Channels = 50

"Blocking probability entered by user ="

0.01

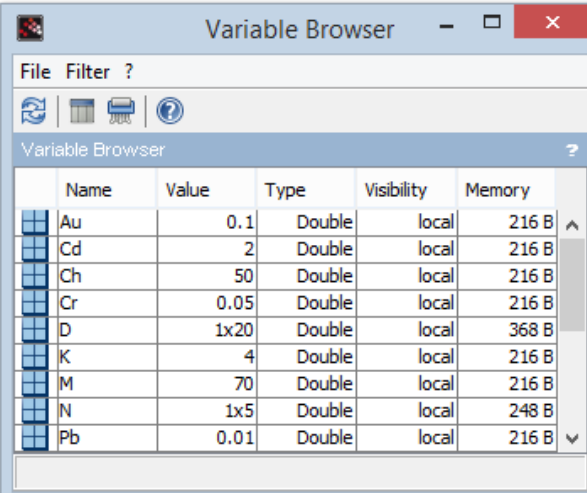
"Maximum Traffic Intensity ="

38.

"Number of Users ="

380.

-->



Variable Browser

	Name	Value	Type	Visibility	Memory
<input checked="" type="checkbox"/>	Au	0.1	Double	local	216 B
<input checked="" type="checkbox"/>	Cd	2	Double	local	216 B
<input checked="" type="checkbox"/>	Ch	50	Double	local	216 B
<input checked="" type="checkbox"/>	Cr	0.05	Double	local	216 B
<input checked="" type="checkbox"/>	D	1x20	Double	local	368 B
<input checked="" type="checkbox"/>	K	4	Double	local	216 B
<input checked="" type="checkbox"/>	M	70	Double	local	216 B
<input checked="" type="checkbox"/>	N	1x5	Double	local	248 B
<input checked="" type="checkbox"/>	Pb	0.01	Double	local	216 B

VIVA VOCE

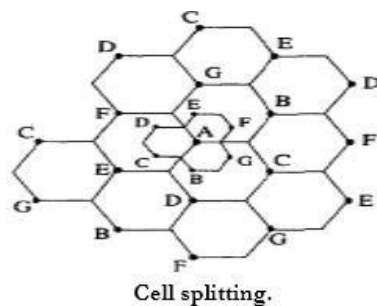
Q1. How can we improve the capacity and coverage area of a cellular system?

Ans. There are 3 techniques for improving cell capacity in cellular system, namely:

- Cell Splitting.
- Sectoring.
- Coverage Zone Approach.

A) CELL SPLITTING:

- It is process of subdividing a congested cell into smaller cells, each with its own base station and a corresponding reduction in antenna height and transmitter power.
- Cell splitting increases capacity of cellular system since it increases number of times that channels are reused, it preserves frequency reuse plan.
- It defines new cells which have smaller radius than original cells and by installing these smaller cells called microcells between existing cells, that is radius will be half of the original cell.
- Thus, capacity increases due to additional number of channels per unit area, but does not disturb the channel allocation scheme required to maintain the minimum co-channel reuse ratio Q between co-channel cells.



B) SECTORING:

- This is another method to increase cellular capacity and coverage by keeping cell radius unchanged and decreasing D/R ratio.
- In this approach, capacity improvement is achieved by reducing the number of cells in a cluster and thus increasing the frequency reuse.
- The co-channel interference in a cellular system may be decreased by replacing a single Omni-directional antenna at the base station by several directional antennas, each radiating within a specified sector.
- The factor by which the co-channel interference is reduced depends on the amount of sectoring used.

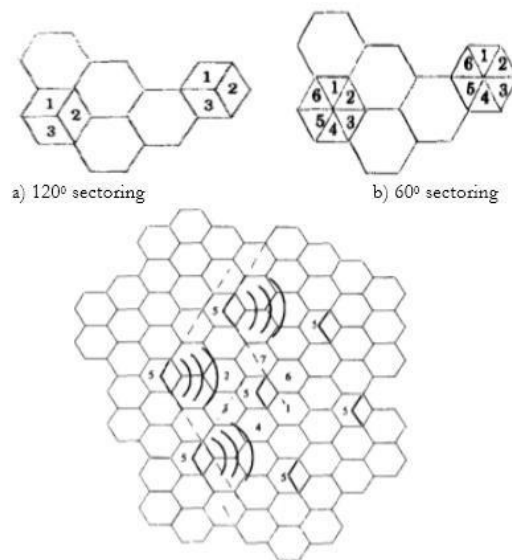


Illustration of how 120° sectoring reduces interference from

Advantages:

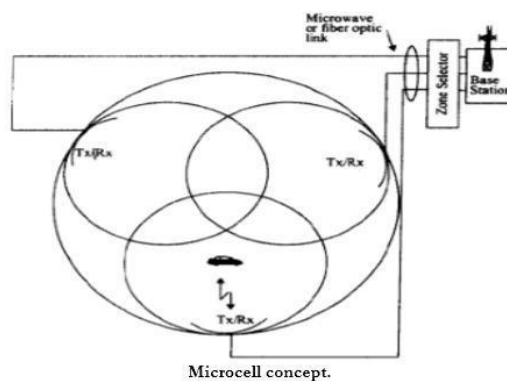
- Improvement in Signal capacity.
- Improvement in signal to interference ratio.
- Increases frequency reuse.

Disadvantages:

- Increase in number of handoffs.
- Increase in number of antenna at each base station.

C) COVERAGE ZONE/ MICROCELL ZONE CONCEPT:

- This approach was presented by Lee to solve the problem of an increased load on the switching and control link elements of the mobile system due to sectoring.
- It is based on a microcell concept for 7 cell reuse.
- In this scheme, each of the three zone sites are connected to a single base station and share the same radio equipment.
- Multiple zones and a single base station make up a cell. As a mobile travels within the cell, it is served by the zone with the strongest signal.
- This approach is superior to sectoring since antennas are placed at the outer edges of the cell, and any base station channel may be assigned to any zone by the base station.



Q2. Why the shape of cell is not circle?

Ans. Circle is the first natural choice to represent the coverage area of a base station. But while adopting this shape, adjacent cells cannot be overlaid upon a map without leaving gaps or creating overlapping regions. Hexagonal cell shape is perfect over square or triangular cell shapes in cellular architecture because it covers an entire area without overlapping i.e. they can cover the entire geographical region without any gaps.

Q3. How is frequency reuse distance measured in cellular system?

Ans. Frequency reusing is the concept of using the same radio frequencies within a given area, that are separated by considerable distance, with minimal interference, to establish communication.

- Frequency reuse offers the following benefits –
- Allows communications within cell on a given frequency
- Limits escaping power to adjacent cells
- Allows re-use of frequencies in nearby cells
- Uses same frequency for multiple conversations
- 10 to 50 frequencies per cell

For example, when **N** cells are using the same number of frequencies and **K** be the total number of frequencies used in systems. Then each **cell frequency** is calculated by using the formulae K/N .

In Advanced Mobile Phone Services (AMPS) when $K = 395$ and $N = 7$, then frequencies per cell on an average will be $395/7 = 56$. Here, **cell frequency** is 56.

Q4. Which is the standard unit used to show the signal strength in mobile?

Ans. Cellular signal strength is measured in decibels (dB), and typically range from -50 dB to -110 dB. The dB scale is logarithmic.

Q5. Why frequency reuse is required?

Ans. Frequency reuse is the process of using the same radio frequencies on radio transmitter sites within a geographic area that are separated by sufficient distance to cause minimal interference with each other. Frequency reuse allows for a dramatic increase in the number of customers that can be served (capacity) within a geographic area on a limited amount of radio spectrum (limited number of radio channels). Frequency reuse allows WiMAX system operators to reuse the same frequency at different cell sites within their system operating area.

Experiment – 3

AIM

Write a program in Scilab to Calculate maximum traffic intensity and maximum number of users accommodated in Erlang B and Erlang C system for given no of channels.

Code:

```
function n=factorial(n)
    if (n<=0)
        then
            n = 1
        else
            n = n* factorial(n-1) end
    endfunction

function p1=erlangB(A1, c1)
    pr2=0;
    pr1=A1^c1/factorial(c1);
    for k=1:c1
        pr2 = pr2+(A1^k/factorial(k));
    end
    p1=pr1/pr2;
endfunction

function [p2]=erlangC(A2, c2)
    temp_1=0;
    for k=0:c2-1
        temp_1= temp_1+A2^k/factorial(k);
    end
    denominator = A^c2+(factorial(c2)*(1-(A2/c))*temp_1);
    p2=A2^c2/denominator;
endfunction

pr_blocking = input("Enter probability of blocking");
pr_delay = input('enter probability of block call delay');
y=input("Enter call rate");
H=input("Enter the average call duration");
c=input("Enter no. of channels");
disp("no of channels");
disp(c);
Au=y*H;

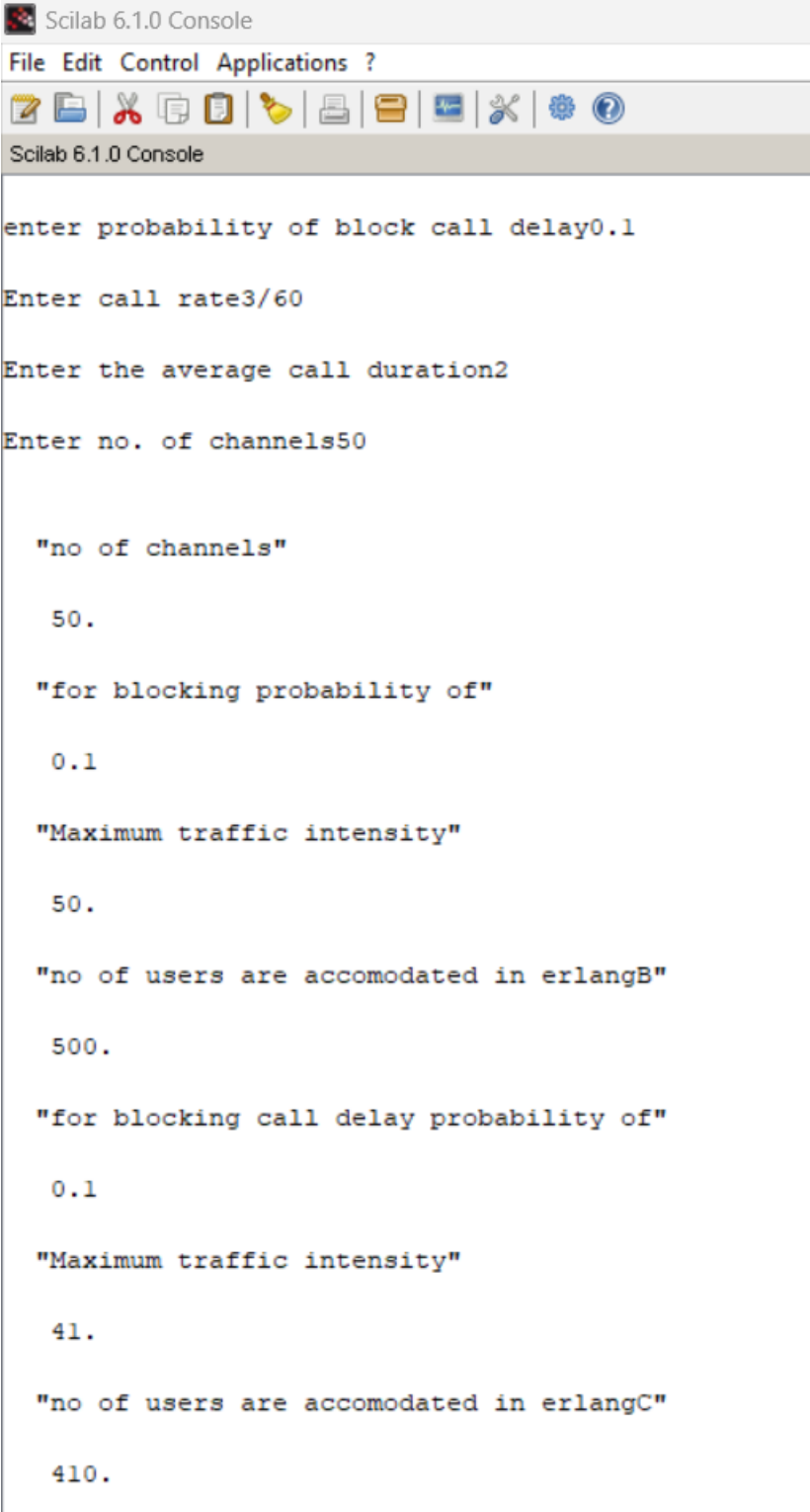
p=0;
for A=1:1:100
    while(p<pr_blocking)
        [p] = erlangB(A,c);
        A=A+1;
    end
    disp("for blocking probability of",pr_blocking); disp("Maximum traffic intensity",A-1);
    u=(A-1)/Au;
    disp('no of users are accomodated in erlangB',u);
    break;
end
```

```

p=0;
for A=1:1:100
    while(p<pr_delay)
        [p] = erlangC(A,c); A=A+1;
    end
    disp("for blocking call delay probability of",pr_blocking);
    disp("Maximum traffic intensity",A-1);
    u=(A-1)/Au;
    disp('no of users are accomodated in erlangC',u);
    break;
end

```

Output:



```

Scilab 6.1.0 Console
File Edit Control Applications ?
[Icons]
Scilab 6.1.0 Console

enter probability of block call delay0.1

Enter call rate3/60

Enter the average call duration2

Enter no. of channels50

"no of channels"

50.

"for blocking probability of"

0.1

"Maximum traffic intensity"

50.

"no of users are accomodated in erlangB"

500.

"for blocking call delay probability of"

0.1

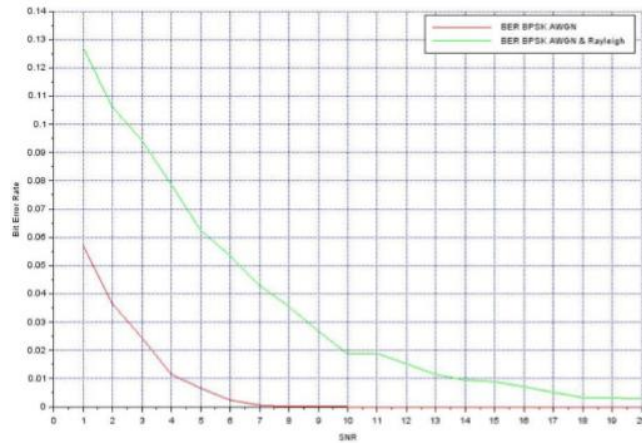
"Maximum traffic intensity"

41.

"no of users are accomodated in erlangC"

410.

```



VIVA VOCE

Q1. How is Erlang traffic calculated?

Ans. Traffic per user $A_u = \lambda H$, where λ is the request rate and H is holding time. It is calculated in Erlangs.

One Erlang: traffic in channel completely occupied.

0.5 Erlang: channel occupied 30 minutes in an hour.

Q2. What is Erlang B?

Ans. Erlang-B should be used when failure to get a free resource results in the customer being denied service. The customer's request is rejected as no free resources are available.

Q3. What is Erlang C?

Ans. Erlang-C should be used when failure to get a free resource results in the customer being added into a queue. The customers stay in the queue until a free resource can be found.

Q4. What is trunking?

Ans. Trunking : A radio system where the channel is allocated on demand by user is called "*trunked*" system. The concept of trunking allows a large number of users to share the relatively small number of channels in a cell by providing access to each user, on demand, from a pool of available channels.

Q5. What is grade of service?

Ans. Grade of Service: It is a measure of the ability of a user to access a trunked system during the busiest hour. It is the measure of congestion which is specified as the probability of a call being blocked (for Erlang B) or the probability of call being delayed beyond a certain amount of time (for Erlang C).

Experiment – 4

AIM –

Write a Program in Scilab to calculate Bit Error rate performance of BPSK modulated signal over only AWGN channel and AWGN and Rayleigh channel both.

Code:

```
clc;
n=10000;
data_stream=grand(1,n,"uin",0,1);
bpsk_stream=2*data_stream-1;
snr=1:20;
l=length(snr);
s_AWGN=0;
s_AWGN_Ray=0;
biterror_AWGN=[];
biterror_AWGN_Rayleigh=[];
for k=1:l
    h=1/sqrt(2)*(rand(1,n,'normal')+%i*(rand(1,n,'normal')));
    noise=1/sqrt(2)*(10^(-(k/20)))*(rand(1,length(bpsk_stream),'normal')+%i*(rand(1,length(bpsk_stream),'normal')));
    s_AWGN=s_AWGN+noise;
    s_AWGN_RAY=s_AWGN.*h+noise;
    received_signal=conj(h).*s_AWGN_RAY; recdata_AWGN=[];
    recdata_AWGN_Rayleigh=[];
    for i=1:n
        if (real(s_AWGN(i))>=0) output_AWGN=1;
        else
            output_AWGN=0;
        end
        recdata_AWGN(i) = output_AWGN;
    end
    for i=1:n
        if (real(s_AWGN_RAY(i))>=0) output_AWGN_Rayleigh=1;
        else
            output_AWGN_Rayleigh=0;
        end
        recdata_AWGN_Rayleigh(i) = output_AWGN_Rayleigh;
    end
    err_AWGN = 0;
    err_AWGN_Rayleigh = 0;
    for i=1:n
        if recdata_AWGN[i] ~= bpsk_stream(i) err_AWGN = err_AWGN + 1;
        end
    end
    for i=1:n
        if recdata_AWGN_Rayleigh[i] ~= bpsk_stream(i) err_AWGN_Rayleigh = err_AWGN_Rayleigh + 1;
        end
    end
    biterror_AWGN(k) = err_AWGN/n;
    biterror_AWGN_Rayleigh(k) = err_AWGN_Rayleigh/n;
end
subplot(2,1,1);
```

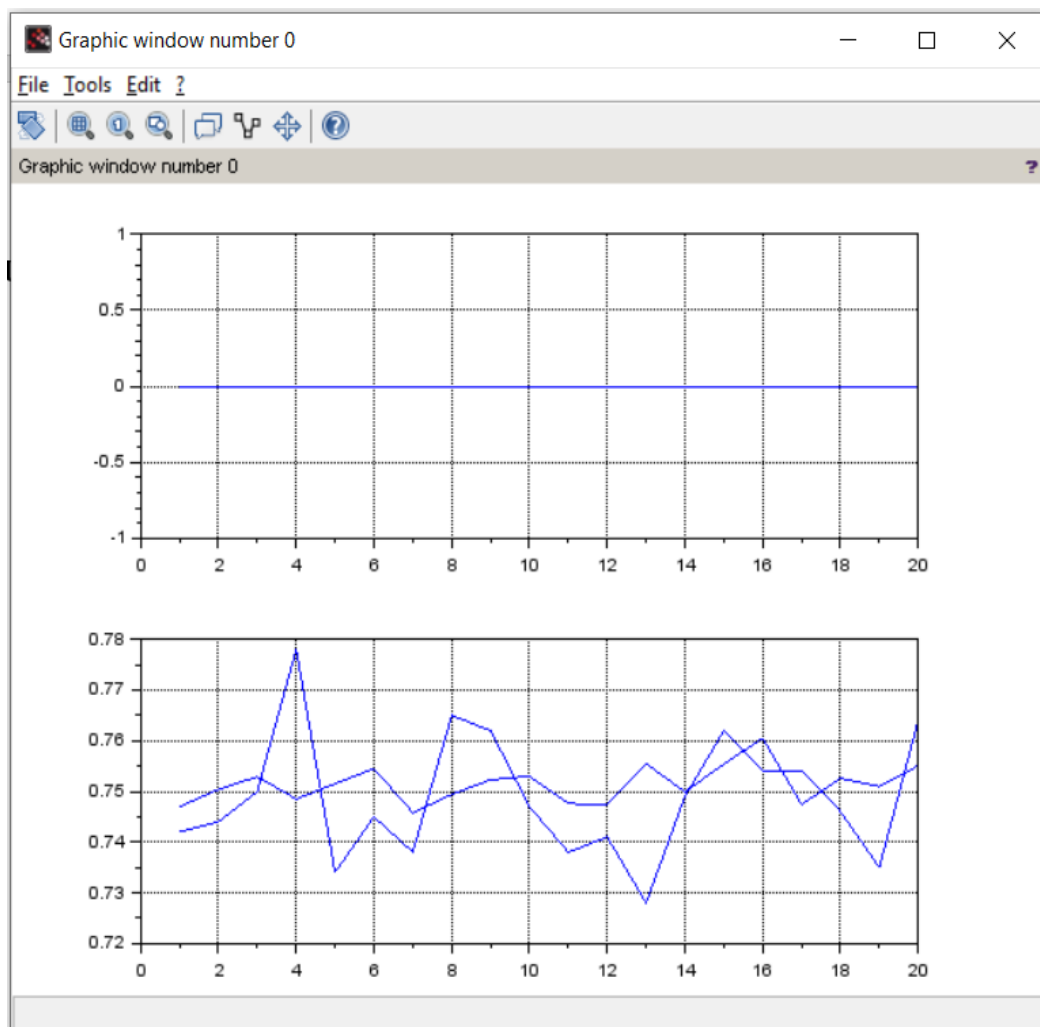
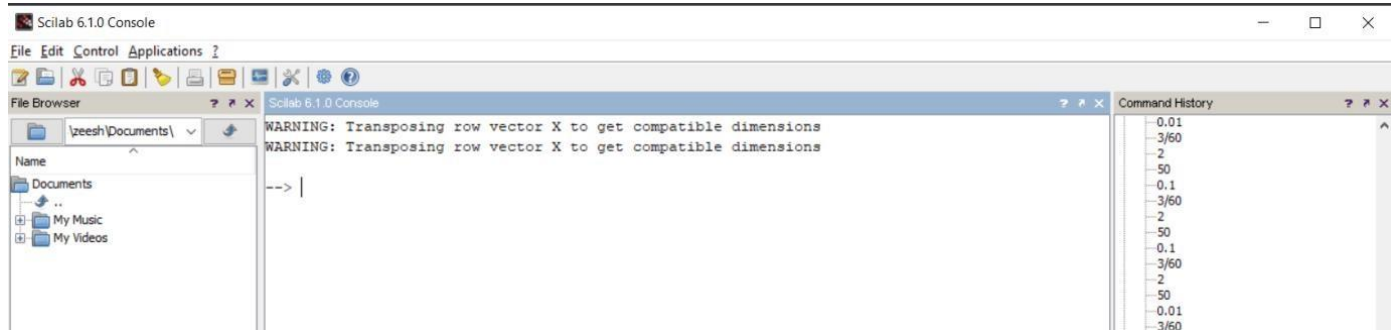


```

plot(snr,bterror_AWGN);
xgrid() subplot(2,1,2);
plot(snr,bterror_AWGN_Rayleigh);
xgrid()

```

Output:



VIVA VOCE

Q1. Explain about Binary Phase Shift Keying (BPSK)?

Ans. In BPSK, for each binary data (0 or 1), the phase of the carrier signal is changed to 0 or 180 degrees

Q2. List down the major advantages and disadvantages of analog communication techniques?

Ans. The major advantages of analog communication are:

- (a) For audio and video transmission, analog signals are mostly suited
- (b) It can be ported easily
- (c) They can be processed easily
- (d) Bandwidth usage is less than digital signals
- (e) For analog signals, the need of new graphics board is not necessary.
- (f) Analog signals can present more refined information because of its higher density

The disadvantages of analog communication are:

- (a) The signal quality of analog signal is very less as compared with digital signals
- (b) The analog cables are easily affected by external influences.
- (c) The circuit complexity is very high for analog communication
- (d) This mode of communication is not reliable
- (e) The cost of devices is high
- (f) This requires more power for transmission.

Q3. Why AWGN is called white noise?

Ans. White refers to the idea that it has uniform power across the frequency band for the information system. It is an analogy to the color white which has uniform emissions at all frequencies in the visible spectrum. Gaussian because it has a normal distribution in the time domain with an average time domain value of zero.

Q4. Give the differences between bit rate and baud rate?

Ans. Per unit time, the number of bits transmitted gives the bit rate. Per unit time, the number of signal units transmitted gives the baud rate.

Q5. Explain the aliasing effect of sampling?

Ans. When the sampling frequency becomes less than the Nyquist rate, then the original signal cannot be recovered from its samples and this effect is termed as aliasing effect. If aliasing occurs, then we cannot retrieve the original message signal from its samples.

Experiment – 5

AIM –

Program in Scilab to Generate Walsh Codes and then spread the user information using it.

Code:

```
clc;
w = [0 0;0 1];
disp('original walsh code matrix = ',w);

function [w_inv]=compliment(w)
    for i=1:1:length(w(1,:))
        for j=1:1:length(w(1,:))
            if
                w(i,j)== 0 w_inv(i,j)=1;
            else w_inv(i,j)=0;
            end
        end
    end
end
endfunction

comp = compliment(w);
disp('compliment of walsh code',comp);
w = [w w; w comp];
disp("New Matrix",w);
len=length(w(2,:));
disp("Length of new matrix :",len);

zeeshaan_input1 = [1 0 0 1 0];
zeeshaan_input2 = [0 1 1 1 0];
zeeshaan_input3 = [1 0 1 1 0];

disp("input1",input1);
disp("input2",input2);
disp("input3",input3);

Wcode1 = w(2,:);
Wcode2 = w(3,:);
Wcode3 = w(4,:);

spread = [] spread1 = [] spread2 = [] spread3 = []
for i=1:1:length(input1)
    for j=1:1:length(Wcode1)
        variable_xor1 = bitxor(input1(1,i),Wcode1(1,j));
        spread1 = [spread1 variable_xor1];
    end
end

disp("Code1 Spread",spread1);

for i=1:1:length(input2)
    for j=1:1:length(Wcode2)
        variable_xor2 = bitxor(input2(1,i),Wcode2(1,j));
        spread2 = [spread2 variable_xor2];
    end
end
```

```

end

disp("Code2 Spread",spread2);

for i=1:1:length(input3)
    for j=1:1:length(Wcode3)
        variable_xor3 = bitxor(input3(1,i),Wcode3(1,j));
        spread3 = [spread3 variable_xor3];
    end
end

disp("Code3 Spread",spread3);
spread = [spread1;spread2;spread3];
disp("Final Spread Code",spread);

```

Output:

```
"original walsh code matrix = "
```

```

0.    0.
0.    1.

```

```
"compliment of walsh code"
```

```

1.    1.
1.    0.

```

```
"New Matrix"
```

```

0.    0.    0.    0.
0.    1.    0.    1.
0.    0.    1.    1.
0.    1.    1.    0.

```

```
"Length of new matrix :"
```

```
4.
```

```
"zeeshaan_input1"
```

```
1.    0.    0.    1.    0.
```

```
"zeeshaan_input2"
```

```
0.    1.    1.    1.    0.
```

```
"zeeshaan_input3"
```

```
1.    0.    1.    1.    0.
```

"Code1 Spread"

column 1 to 18
1. 0. 1. 0. 0. 1. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 0. 1.
column 19 to 20
0. 1.

"Code2 Spread"

column 1 to 18
0. 0. 1. 1. 1. 1. 0. 0. 1. 1. 0. 0. 1. 1. 0. 0. 0. 0.
column 19 to 20
1. 1.

"Code3 Spread"

column 1 to 18
1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 0. 1. 1. 0. 0. 1. 0. 1.
column 19 to 20
1. 0.

"Final Spread Code"

column 1 to 18
1. 0. 1. 0. 0. 1. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 0. 1.
0. 0. 1. 1. 1. 1. 0. 0. 1. 1. 0. 0. 1. 1. 0. 0. 0. 0.
1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 0. 1. 1. 0. 0. 1. 0. 1.
column 19 to 20
0. 1.
1. 1.
1. 0.

→ |

VIVA VOCE

Q1. What is Scilab and what is it used for?

Ans. Scilab is a free and open-source scientific software package for numerical computation, which includes functions for data analysis, signal processing, optimization, and more. It is used in a wide range of fields, including engineering, science, and economics.

Q2. How does Scilab compare to other software packages such as MATLAB or Octave?

Ans. Scilab is similar to other software packages like MATLAB and Octave in that it is a powerful tool for numerical computation and scientific simulations. However, Scilab is free and open-source, whereas MATLAB and Octave are proprietary software.

Q3. How can I get started with using Scilab?

Ans. To get started with using Scilab, you can download it from the official website (<https://www.scilab.org/>) and install it on your computer. Then, you can explore the documentation and tutorials provided by the Scilab website to learn how to use its various functions and features.

Q4. Can Scilab be used to create graphical user interfaces (GUIs)?

Ans. Yes, Scilab includes a GUI builder called Xcos that can be used to create custom graphical user interfaces for your Scilab scripts.

Q5. Are there any resources available for learning more about Scilab?

Ans. Yes, the Scilab website (<https://www.scilab.org/>) offers a wealth of resources for learning more about Scilab, including documentation, tutorials, and examples. In addition, there are many online forums and communities where you can ask questions and discuss Scilab with other users.

Experiment – 6

AIM –

Write a program in scilab to Generate PN Sequence for CDMA Systems.

Code:

```
clc;
r(1)= 1;
r(2)= 0;
r(3)= 1;
r(4)= 0;
R = [r(1) r(2) r(3) r(4)];
PN = [];
len = length(r);
disp('length of input',len);
disp('initial bit pattern of flip flops',R);

for i=1:1:((2^len)-1)
    temp1 = r(1);
    temp2 = r(2);
    temp3 = r(3);
    temp4 = r(4);
    PN = [PN r(4)];
    temp1 = bitxor(temp3,temp4);
    r(4) = r(3);
    r(3) = r(2);
    r(2) = r(1);
    r(1) = temp1;
    R = [r(1) r(2) r(3) r(4)];
    disp('current bit pattern of flip flops',R);
end
disp('15 bit pattern',PN);

for i=1:1:((2^len)-1)
    if(PN(i)==0)
        PN(i) = -1;
    end
end
disp('After replacing 0 with -1 the 15 bit pattern',PN);

info = [1 -1 -1 1];
leninfo = length(info);
disp("length of data ",length(info));

spread = [];
for i=1:leninfo
    for j=1:length(PN)
        x = info(1,i)*PN(1,j); spread = [spread x];
    end
end
disp("spread ",spread); len_spread = length(spread);
disp("length of spreaded data", len_spread); PN = [PN PN PN PN];
disp("Updated PN : ",PN); despread = [];
```

```

for i=1:1:length(spread)
    x = spread(1,i)*PN(1,i);
    despread = [despread x];
end

disp("multiplied output",despread);
sum_col = [sum(despread(1:15)) sum(despread(16:30)) sum(despread(31:45))
sum(despread(46:60))];
disp("sum total",sum_col);

received_signal = [];
for i=1:1:length(sum_col)
    received_signal = [received_signal sum_col(i)/15];
end
disp("received signal",received_signal);

```

Output:

```

"current bit pattern of flip flops"

0.   1.   0.   0.

"current bit pattern of flip flops"

0.   0.   1.   0.

"current bit pattern of flip flops"

1.   0.   0.   1.

"current bit pattern of flip flops"

1.   1.   0.   0.

"current bit pattern of flip flops"

0.   1.   1.   0.

"current bit pattern of flip flops"

1.   0.   1.   1.

"current bit pattern of flip flops"

0.   1.   0.   1.

"current bit pattern of flip flops"

1.   0.   1.   0.

"15 bit pattern"

0.   1.   0.   1.   1.   1.   1.   0.   0.   0.   1.   0.   0.   1.   1.

"After replacing 0 with -1 the 15 bit pattern"

-1.   1.  -1.   1.   1.   1.   1.  -1.  -1.  -1.   1.  -1.  -1.   1.   1.

```



```
"length of input"
4.
"initial bit pattern of flip flops"
1.  0.  1.  0.
"current bit pattern of flip flops"
1.  1.  0.  1.
"current bit pattern of flip flops"
1.  1.  1.  0.
"current bit pattern of flip flops"
1.  1.  1.  1.
"current bit pattern of flip flops"
0.  1.  1.  1.
"current bit pattern of flip flops"
0.  0.  1.  1.
"current bit pattern of flip flops"
0.  0.  0.  1.
"current bit pattern of flip flops"
1.  0.  0.  0.
```

```
"length of input"
4.
"initial bit pattern of flip flops"
1.  0.  1.  0.
"current bit pattern of flip flops"
1.  1.  0.  1.
"current bit pattern of flip flops"
1.  1.  1.  0.
"current bit pattern of flip flops"
1.  1.  1.  1.
"current bit pattern of flip flops"
0.  1.  1.  1.
"current bit pattern of flip flops"
0.  0.  1.  1.
"current bit pattern of flip flops"
0.  0.  0.  1.
"current bit pattern of flip flops"
1.  0.  0.  0.
```

"length of data "

4.

"spread "

```
      column 1 to 18
-1.  1. -1.  1.  1.  1.  1. -1. -1. -1.  1. -1. -1.  1.  1.  1. -1.  1.
      column 19 to 36
-1. -1. -1. -1.  1.  1.  1. -1.  1.  1. -1. -1.  1. -1.  1. -1. -1. -1.
      column 37 to 54
-1.  1.  1.  1. -1.  1.  1. -1. -1. -1.  1. -1.  1.  1.  1.  1. -1. -1.
      column 55 to 60
-1.  1. -1. -1.  1.  1.
```

"length of spreaded data"

60.

"Updated PN : "

```
      column 1 to 18
-1.  1. -1.  1.  1.  1.  1. -1. -1. -1.  1. -1. -1.  1.  1. -1.  1. -1.
      column 19 to 36
  1.  1.  1.  1. -1. -1. -1.  1. -1. -1.  1.  1. -1.  1. -1.  1.  1.
      column 37 to 54
  1. -1. -1. -1.  1. -1. -1.  1.  1. -1.  1. -1.  1.  1.  1.  1. -1. -1.
      column 55 to 60
-1.  1. -1. -1.  1.  1.
```

"multiplied output"

```
      column 1 to 18
  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1. -1. -1. -1.
      column 19 to 36
-1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1. -1.
      column 37 to 54
-1. -1. -1. -1. -1. -1. -1. -1. -1.  1.  1.  1.  1.  1.  1.  1.  1.
      column 55 to 60
  1.  1.  1.  1.  1.  1.
```

"sum total"

```
 15. -15. -15.  15.
```

"received signal"

```
 1. -1. -1.  1.
```

--> |

VIVA VOCE

Q1. How Many Channels Are There In Cdma Forward Channels?

Ans. Forward channel consists of four channels which includes:

- Pilot Channel
- Sync Channel
- Paging Channel and
- Forward Traffic Channels.

Q2. What is CDMA?

Ans. CDMA stands for Code Division Multiple Access. It is a wireless technology used in transmission of signals from places with high Security and noise reduction. The principle of Spread Spectrum is used to work with CDMA. Spread signal is below the noise level and noise has no effect on the signal. CDMA is not a frequency specific to each user, instead, every channel uses the full available spectrum. Individual conversations are encoded with a pseudo-random digital sequence. A unique code is received by all mobile network users and is allowed continuous network access instead of intermittent or timed access.

Q3. What is FDMA? Explain?

Ans. Frequency Division Multiple Access (FDMA) is one of the most common analogue multiple access methods. The frequency band is divided into channels of equal bandwidth so that each conversation is carried on a different frequency. Guard bands are used between the adjacent signal spectra to minimize crosstalk between the channels.

Advantages:

In FDMA when the channel is not used, it is the channel bandwidth while rest simply is relatively narrow (30 KHz), known as System narrowband. Little or no equalization is needed. For broadcasting, time symbols are suitable analogue links. Framing for FDMA or synchronization bits are not needed for the tight filter streaming. It is required to minimize the combined interference of FDD.

- **Disadvantages:**

It does not differ significantly from analog systems; improving the capacity depends on the signal-to-interference reduction, or a signal-to-noise ratio (SNR).

- The maximum flow rate per channel is fixed and small.
- Guard bands lead to a waste of capacity.
- Hardware implies narrowband filters, which cannot be realized in VLSI and therefore increases the cost.

Q4. Differentiate between CDMA and FDMA?

Ans.

CDMA:

- Same frequency is used by every user and simultaneous transmission occurs
- Every narrowband signal is multiplied by wideband spreading signal, usually known as codeword
- Every user has a separate pseudo-codeword, i.e., orthogonal to others
- Only the desired codeword is detected by the receivers and others appear as noise
- It is mandatory for the receivers to know about the transmitter's codeword

FDMA:

- When the channel is not in use, it sits simply idle
- Bandwidth of Channel is relatively narrow (30 KHz), known as narrowband system
- Little or no equalization is needed for spreading symbol time
- Analog links are suitable for FDMA
- Framing or synchronization bits are not needed for continuous transmission
- Tight filtering is needed to minimize interference
- Combined with FDD for duplexing

Q5. Explain the following two types of hand off in CDMA system: a.) Soft handoff b.) Softer handoff.

Ans.

Soft handoff

- Soft handoff is a feature in which a cellular phone is simultaneously connected to two or more cellular phones during a single call
- It is the overlapping of repeater coverage areas, which enables every cell phone set is always well within the range of a specific repeater.
- More than one repeater can send and receive signals to transmit signals to and from mobiles.
- All repeaters are used with the same frequency channel for each mobile phone set.
- Practically no dead zones and as result, the connections seldom interrupted or dropped.

Softer handoff

- Softer handoff is a significant soft handover in which the added and removed links belong to the same node
- Macro diversity with maximum ratio combining could be performed in the same node
- The movement of handoff, when a user can be served in another cell more efficiently (less power emission, less interference), is the most obvious cause for better performance.

Experiment – 7

AIM –

Write a Program in NS3 to connect WIFI TO BUS (CSMA) Network.

Code:

```
#include "ns3/core-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/network-module.h"
#include "ns3/applications-module.h"
#include "ns3/wifi-module.h"
#include "ns3/mobility-module.h"
#include "ns3/csma-module.h"
#include "ns3/internet-module.h"

using namespace ns3;
NS_LOG_COMPONENT_DEFINE("ThirdScriptExample");
int main(int argc, char *argv[])
{
    bool verbose = true;
    uint32_t nCsma = 3;
    uint32_t nWifi = 3;
    bool tracing = false;

    CommandLine cmd;
    cmd.AddValue("nCsma", "Number of \"extra\" CSMA nodes/devices", nCsma);
    cmd.AddValue("nWifi", "Number of wifi STA devices", nWifi);
    cmd.AddValue("verbose", "Tell echo applications to log if true", verbose);
    cmd.AddValue("tracing", "Enable pcap tracing", tracing);
    cmd.Parse(argc, argv);
    // Check for valid number of csma or wifi nodes
    // 250 should be enough, otherwise IP addresses
    // soon become an issue
    if (nWifi > 250 || nCsma > 250)
    {
        std::cout << "Too many wifi or csma nodes, no more than 250 each." << std::endl;
        return 1;
    }
    if (verbose)
    {
        LogComponentEnable("UdpEchoClientApplication", LOG_LEVEL_INFO);
        LogComponentEnable("UdpEchoServerApplication", LOG_LEVEL_INFO);
    }

    NodeContainer p2pNodes;
    p2pNodes.Create(2);

    PointToPointHelper pointToPoint;
    pointToPoint.SetDeviceAttribute("DataRate", StringValue("5Mbps"));
    pointToPoint.SetChannelAttribute("Delay", StringValue("2ms"));

    NetDeviceContainer p2pDevices;
    p2pDevices = pointToPoint.Install(p2pNodes);
```

```

NodeContainer csmaNodes;
csmaNodes.Add(p2pNodes.Get(1));
csmaNodes.Create(nCsmas);

CsmaHelper csma;
csma.SetChannelAttribute("DataRate", StringValue("100Mbps"));
csma.SetChannelAttribute("Delay", TimeValue(NanoSeconds(6560)));

NetDeviceContainer csmaDevices;
csmaDevices = csma.Install(csmaNodes);

NodeContainer wifiStaNodes;
wifiStaNodes.Create(nWifi);
NodeContainer wifiApNode = p2pNodes.Get(0);

YansWifiChannelHelper channel = YansWifiChannelHelper::Default();
YansWifiPhyHelper phy = YansWifiPhyHelper();
phy.SetChannel(channel.Create());

WifiHelper wifi;
wifi.SetRemoteStationManager("ns3::AarfwifiManager");
WifiMacHelper mac;
Ssid ssid = Ssid("ns-3-ssid");
mac.SetType("ns3::StaWifiMac", "Ssid", SsidValue(ssid), "ActiveProbing",
BooleanValue(false));

NetDeviceContainer staDevices;
staDevices = wifi.Install(phy, mac, wifiStaNodes);
mac.SetType("ns3::ApWifiMac", "Ssid", SsidValue(ssid));
NetDeviceContainer apDevices;
apDevices = wifi.Install(phy, mac, wifiApNode);
MobilityHelper mobility;
mobility.SetPositionAllocator("ns3::GridPositionAllocator", "MinX", DoubleValue(0.0),
"MinY", DoubleValue(0.0),
                                "DeltaX", DoubleValue(5.0),
                                "DeltaY", DoubleValue(10.0),
                                "GridWidth", UIntegerValue(3), "LayoutType",
StringValue("RowFirst"));

mobility.SetMobilityModel("ns3::RandomWalk2dMobilityModel", "Bounds",
RectangleValue(Rectangle(-50, 50, -50, 50)));
mobility.Install(wifiStaNodes);

mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(wifiApNode);

InternetStackHelper stack;
stack.Install(csmaNodes);
stack.Install(wifiApNode);
stack.Install(wifiStaNodes);
Ipv4AddressHelper address;
address.SetBase("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer p2pInterfaces;
p2pInterfaces = address.Assign(p2pDevices);

```

```

address.SetBase("10.1.2.0", "255.255.255.0");
Ipv4InterfaceContainer csmaInterfaces;
csmaInterfaces = address.Assign(csmaDevices);

address.SetBase("10.1.3.0", "255.255.255.0");
address.Assign(staDevices);
address.Assign(apDevices);

UdpEchoServerHelper echoServer(9);

ApplicationContainer serverApps = echoServer.Install(csmaNodes.Get(nCsma));
serverApps.Start(Seconds(1.0));
serverApps.Stop(Seconds(10.0));

UdpEchoClientHelper echoClient(csmaInterfaces.GetAddress(nCsma), 9);
echoClient.SetAttribute("MaxPackets", UintegerValue(1));
echoClient.SetAttribute("Interval", TimeValue(Seconds(1.0)));
echoClient.SetAttribute("PacketSize", UintegerValue(1024));

ApplicationContainer clientApps = echoClient.Install(wifiStaNodes.Get(nWifi - 1));
clientApps.Start(Seconds(2.0));
clientApps.Stop(Seconds(10.0));
Ipv4GlobalRoutingHelper::PopulateRoutingTables();
Simulator::Stop(Seconds(10.0));
if (tracing == true)
{
    pointToPoint.EnablePcapAll("third");
    phy.EnablePcap("third", apDevices.Get(0));
    csma.EnablePcap("third", csmaDevices.Get(0), true);
}
Simulator::Run();
Simulator::Destroy();
return 0;
}

```

Output:

```

● reeha@Reeha:~/networkEng/ns-allinone-3.35/ns-3.35$ ./waf --run wifi_to_bus
Waf: Entering directory `/home/reeha/networkEng/ns-allinone-3.35/ns-3.35/build'
[2952/3001] Compiling scratch/wifi_to_bus.cc
[2962/3001] Linking build/scratch/wifi_to_bus
Waf: Leaving directory `/home/reeha/networkEng/ns-allinone-3.35/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (4.723s)
At time +2s client sent 1024 bytes to 10.1.2.4 port 9
At time +2.01799s server received 1024 bytes from 10.1.3.3 port 49153
At time +2.01799s server sent 1024 bytes to 10.1.3.3 port 49153
At time +2.03367s client received 1024 bytes from 10.1.2.4 port 9
○ reeha@Reeha:~/networkEng/ns-allinone-3.35/ns-3.35$ █

```

VIVA VOCE

Q1. What is CSMA and how does it work in wireless networks?

Ans. CSMA is a medium access control (MAC) protocol used in wireless networks to allow multiple devices to share the same communication channel. It works by allowing devices to sense the presence of other devices transmitting on the channel, and to defer their own transmission until the channel is free.

Q2. How is CSMA implemented in ns-3?

Ans. In ns-3, the CSMA protocol is implemented as a set of classes and functions in the "csma" module. These classes and functions provide the necessary functionality for simulating CSMA-based wireless networks in ns-3.

Q3. How can I use CSMA in my ns-3 simulations?

Ans. To use CSMA in your ns-3 simulations, you will need to create instances of the appropriate CSMA classes and configure them with the desired settings. You will also need to create nodes and devices that use CSMA as their MAC protocol, and connect them together to form a network.

Q4. What are some limitations or drawbacks of using CSMA in wireless networks?

Ans. One of the main limitations of CSMA is that it can lead to low utilization of the communication channel, as devices may need to wait for a long time before they are able to transmit. This can lead to poor performance in terms of throughput and delay. In addition, CSMA can be vulnerable to hidden terminal problems, where two devices may not be able to hear each other but both think the channel is free and start transmitting at the same time, leading to a collision.

Q5. Are there any alternatives to CSMA for medium access control in wireless networks?

Ans. Yes, there are many alternative MAC protocols that can be used in wireless networks, such as TDMA (Time Division Multiple Access), FDMA (Frequency Division Multiple Access), and others. These protocols can offer different trade-offs in terms of performance and complexity, and may be more suitable for certain types of networks or applications.

Experiment – 8

AIM –

Write a Program in NS3 to create WIFI Network in SIMPLE INFRASTRUCTURE MODE (of nodes).

Code:

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
#include "ns3/config-store-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>

using namespace ns3;
NS_LOG_COMPONENT_DEFINE("WifiSimpleInfra");
void ReceivePacket(Ptr<Socket> socket)
{
    while (socket->Recv())
    {
        NS_LOG_UNCOND("Received one packet!");
    }
}

static void GenerateTraffic(Ptr<Socket> socket, uint32_t pktSize, uint32_t pktCount, Time
pktInterval)
{
    if (pktCount > 0)
    {
        socket->Send(Create<Packet>(pktSize));
        Simulator::Schedule(pktInterval, &GenerateTraffic, socket, pktSize, pktCount - 1,
pktInterval);
    }
    else
    {
        socket->Close();
    }
}

int main(int argc, char *argv[])
{
    std::string phyMode("DsssRate1Mbps");
    double rss = -80;          // -dBm
    uint32_t packetSize = 1000; // bytes
    uint32_t numPackets = 1;
    double interval = 1.0;     // seconds
    bool verbose = false;

    CommandLine cmd;

    cmd.AddValue("phyMode", "Wifi Phy mode", phyMode);
    cmd.AddValue("rss", "received signal strength", rss);
```

```

cmd.AddValue("packetSize", "size of application packet sent", packetSize);
cmd.AddValue("numPackets", "number of packets generated", numPackets);
cmd.AddValue("interval", "interval (seconds) between packets", interval);
cmd.AddValue("verbose", "turn on all WifiNetDevice log components", verbose);
cmd.Parse(argc, argv); // Convert to time object
Time interPacketInterval = Seconds (interval);
// disable fragmentation for frames below 2200 bytes

Config::SetDefault("ns3::WifiRemoteStationManager::FragmentationThreshold",
StringValue("2200"));
// turn off RTS/CTS for frames below 2200 bytes
Config::SetDefault("ns3::WifiRemoteStationManager::RtsCtsThreshold",
StringValue("2200"));
// Fix non-unicast data rate to be the same as that of unicast Config::SetDefault
("ns3::WifiRemoteStationManager::NonUnicastMode", StringValue (phyMode));

NodeContainer c;
c.Create(2);
// The below set of helpers will help us to put together the wifi NICs we want
WifiHelper wifi;
if (verbose)
{
    wifi.EnableLogComponents(); // Turn on all Wifi logging
}

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper();
wifiPhy.Set("RxGain", DoubleValue(0));
wifiPhy.SetPcapDataLinkType(YansWifiPhyHelper::DLT_IEEE802_11_RADIO);

YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss("ns3::FixedRssLossModel", "Rss", DoubleValue(rss));
wifiPhy.SetChannel(wifiChannel.Create());

WifiMacHelper wifiMac;
wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager", "DataMode",
StringValue(phyMode), "ControlMode", StringValue(phyMode));

Ssid ssid = Ssid("wifi-default"); // setup sta.
wifiMac.SetType("ns3::StaWifiMac", "Ssid", SsidValue(ssid), "ActiveProbing",
BooleanValue(false));

NetDeviceContainer staDevice = wifi.Install(wifiPhy, wifiMac, c.Get(0));
NetDeviceContainer devices = staDevice;

// setup ap.
wifiMac.SetType("ns3::ApWifiMac", "Ssid", SsidValue(ssid));
NetDeviceContainer apDevice = wifi.Install(wifiPhy, wifiMac, c.Get(1));
devices.Add(apDevice);
// Note that with FixedRssLossModel, the positions below are not
// used for received signal strength.
MobilityHelper mobility;
Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator>();
positionAlloc->Add(Vector(0.0, 0.0, 0.0));
positionAlloc->Add(Vector(5.0, 0.0, 0.0));

```

```

mobility.SetPositionAllocator(positionAlloc);
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(c);

InternetStackHelper internet;
internet.Install(c);

Ipv4AddressHelper ipv4;
NS_LOG_INFO("Assign IP Addresses.");
ipv4.SetBase("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer i = ipv4.Assign(devices);

TypeId tid = TypeId::LookupByName("ns3::UdpSocketFactory");
Ptr<Socket> recvSink = Socket::CreateSocket(c.Get(0), tid);
InetSocketAddress local = InetSocketAddress(Ipv4Address::GetAny(), 80);
recvSink->Bind(local);
recvSink->SetRecvCallback(MakeCallback(&ReceivePacket));

Ptr<Socket> source = Socket::CreateSocket(c.Get(1), tid);
InetSocketAddress remote = InetSocketAddress(Ipv4Address("255.255.255.255"), 80);
source->SetAllowBroadcast(true);
source->Connect(remote);
// Tracing
wifiPhy.EnablePcap("wifi-simple-infra", devices);
// Output what we are doing
NS_LOG_UNCOND("Testing " << numPackets << " packets sent with receiver rss " << rss);
Simulator::ScheduleWithContext(source->GetNode()->GetId(),
                               Seconds(1.0), &GenerateTraffic,
                               source, packetSize, numPackets, interPacketInterval);

Simulator::Stop(Seconds(30.0));
Simulator::Run();
Simulator::Destroy();
return 0;
}

```

Output:

```

● reeha@Reeha:~/networkEng/ns-allinone-3.35/ns-3.35$ ./waf --run wifi
Waf: Entering directory `/home/reeha/networkEng/ns-allinone-3.35/ns-3.35/build'
[2953/3003] Compiling scratch/wifi.cc
[2964/3003] Linking build/scratch/wifi
Waf: Leaving directory `/home/reeha/networkEng/ns-allinone-3.35/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (4.792s)
Testing 1 packets sent with receiver rss -80
Received one packet!
○ reeha@Reeha:~/networkEng/ns-allinone-3.35/ns-3.35$

```

Q1. What is Wi-Fi and how does it work?

Ans. Wi-Fi is a wireless networking technology that allows devices to communicate over short distances using radio waves. It operates in the 2.4 GHz and 5 GHz bands, and uses a variety of protocols and standards to provide wireless connectivity for devices such as laptops, smartphones, and tablets.

Q2. How is Wi-Fi implemented in ns-3?

Ans. In ns-3, the Wi-Fi technology is implemented as a set of classes and functions in the "wifi" module. These classes and functions provide the necessary functionality for simulating Wi-Fi networks in ns-3, including support for various modes of operation, channel models, and MAC protocols.

Q3. How can I use Wi-Fi in my ns-3 simulations?

Ans. To use Wi-Fi in your ns-3 simulations, you will need to create instances of the appropriate Wi-Fi classes and configure them with the desired settings. You will also need to create nodes and devices that use Wi-Fi as their networking technology, and connect them together to form a network.

Q4. What are some limitations or challenges of using Wi-Fi in ns-3 simulations?

Ans. One of the main challenges of simulating Wi-Fi networks in ns-3 is accurately modeling the complex behavior of the physical layer, including the effects of interference, fading, and multipath. In addition, simulating large-scale Wi-Fi networks can be computationally intensive and may require significant resources.

Q5. Are there any alternatives to Wi-Fi for wireless networking in ns-3?

Ans. Yes, there are many other wireless networking technologies that can be simulated in ns-3, such as cellular networks, satellite networks, and others. These technologies can offer different trade-offs in terms of performance and complexity, and may be more suitable for certain types of networks or applications.

Experiment – 9

AIM –

Write a Program in NS3 to Create a wireless mobile ad-hoc network between three nodes.

Code:

```
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
#include "ns3/config-store-module.h"
#include "ns3/wifi-module.h"
#include "ns3/internet-module.h"
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
using namespace ns3;
NS_LOG_COMPONENT_DEFINE("WifiSimpleAdhoc");

void ReceivePacket(Ptr<Socket> socket)
{
    while (socket->Recv())
    {
        NS_LOG_UNCOND("Received one packet!");
    }
}

static void GenerateTraffic(Ptr<Socket> socket, uint32_t pktSize, uint32_t pktCount, Time
pktInterval)
{
    if (pktCount > 0)
    {
        socket->Send(Create<Packet>(pktSize));
        Simulator::Schedule(pktInterval, &GenerateTraffic,
                                socket, pktSize, pktCount - 1, pktInterval);
    }
    else
        socket->Close();
}

int main(int argc, char *argv[])
{
    std::string phyMode("DsssRate1Mbps");
    double rssi = -80; // -dBm
    uint32_t packetSize = 1000; // bytes
    uint32_t numPackets = 1;
    double interval = 1.0; // seconds
    bool verbose = false;
    CommandLine cmd;
    cmd.AddValue("phyMode", "Wifi Phy mode", phyMode);
    cmd.AddValue("rssi", "received signal strength", rssi);
    cmd.AddValue("packetSize", "size of application packet sent", packetSize);
    cmd.AddValue("numPackets", "number of packets generated", numPackets);
    cmd.AddValue("interval", "interval (seconds) between packets", interval);
```

```

cmd.AddValue("verbose", "turn on all WifiNetDevice log components", verbose);
cmd.Parse(argc, argv); // Convert to time object
Time interPacketInterval = Seconds(interval);

// disable fragmentation for frames below 2200 bytes
Config::SetDefault("ns3::WifiRemoteStationManager::FragmentationThreshold",
StringValue("2200"));
// turn off RTS/CTS for frames below 2200 bytes
Config::SetDefault("ns3::WifiRemoteStationManager::RtsCtsThreshold",
StringValue("2200"));
// Fix non-unicast data rate to be the same as that of unicast
Config::SetDefault("ns3::WifiRemoteStationManager::NonUnicastMode",
StringValue(phyMode));
NodeContainer c;
c.Create(2);
// The below set of helpers will help us to put together the wifi NICs we want
WifiHelper wifi;
if (verbose)
    wifi.EnableLogComponents(); // Turn on all Wifi logging
    // wifi.SetStandard (WIFI_PHY_STANDARD_80211b);

YansWifiPhyHelper wifiPhy = YansWifiPhyHelper();
// This is one parameter that matters when using FixedRssLossModel
// set it to zero; otherwise, gain will be added
wifiPhy.Set("RxGain", DoubleValue(0));
// ns-3 supports RadioTap and Prism tracing extensions for 802.11b
wifiPhy.SetPcapDataLinkType(YansWifiPhyHelper::DLT_IEEE802_11_RADIO);
YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay("ns3::ConstantSpeedPropagationDelayModel");
// The below FixedRssLossModel will cause the rss to be fixed regardless
// of the distance between the two stations, and the transmit power
wifiChannel.AddPropagationLoss("ns3::FixedRssLossModel", "Rss", DoubleValue(rss));
wifiPhy.SetChannel(wifiChannel.Create());
// Add a mac and disable rate control
WifiMacHelper wifiMac;
wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager", "DataMode",
StringValue(phyMode), "ControlMode", StringValue(phyMode));
// Set it to adhoc mode
wifiMac.SetType("ns3::AdhocWifiMac");
NetDeviceContainer devices = wifi.Install(wifiPhy, wifiMac, c);
// Note that with FixedRssLossModel, the positions below are not
// used for received signal strength.
MobilityHelper mobility;
Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator>();
positionAlloc->Add(Vector(0.0, 0.0, 0.0));
positionAlloc->Add(Vector(5.0, 0.0, 0.0));
mobility.SetPositionAllocator(positionAlloc);
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.Install(c);
InternetStackHelper internet;
internet.Install(c);
Ipv4AddressHelper ipv4;
NS_LOG_INFO("Assign IP Addresses.");
ipv4.SetBase("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer i = ipv4.Assign(devices);

```

```

TypeId tid = TypeId::LookupByName("ns3::UdpSocketFactory");
Ptr<Socket> recvSink = Socket::CreateSocket(c.Get(0), tid);
InetSocketAddress local = InetSocketAddress(Ipv4Address::GetAny(), 80);

recvSink->Bind(local);
recvSink->SetRecvCallback(MakeCallback(&ReceivePacket));
Ptr<Socket> source = Socket::CreateSocket(c.Get(1), tid);
InetSocketAddress remote = InetSocketAddress(Ipv4Address("255.255.255.255"), 80);
source->SetAllowBroadcast(true);
source->Connect(remote);
// Tracing
wifiPhy.EnablePcap("wifi-simple-adhoc", devices);
// Output what we are doing
NS_LOG_UNCOND("Testing " << numPackets << " packets sent with receiver rss " << rss);
Simulator::ScheduleWithContext(source->GetNode()->GetId(), Seconds(1.0),
&GenerateTraffic, source, packetSize, numPackets, interPacketInterval);
Simulator::Run();
Simulator::Destroy();
return 0;
}

```

Output:

```

reeha@Reeha:~/networkEng/ns-allinone-3.35/ns-3.35$ ./waf --run adhoc
Waf: Entering directory `/home/reeha/networkEng/ns-allinone-3.35/ns-3.35/build'
[2956/3005] Compiling scratch/adhoc.cc
[2966/3005] Linking build/scratch/adhoc
Waf: Leaving directory `/home/reeha/networkEng/ns-allinone-3.35/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (4.791s)
Testing 1 packets sent with receiver rss -80
aborted. cond="it == m_phyEntities.end()", msg="Unsupported Wi-Fi modulation class", +1.000034000s 1 file=../src/wifi/model/wifi-phy.cc, line=881
terminate called without an active exception

reeha@Reeha:~/networkEng/ns-allinone-3.35/ns-3.35$ ./waf --run wifi
Waf: Entering directory `/home/reeha/networkEng/ns-allinone-3.35/ns-3.35/build'
[2953/3003] Compiling scratch/wifi.cc
[2964/3003] Linking build/scratch/wifi
Waf: Leaving directory `/home/reeha/networkEng/ns-allinone-3.35/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (4.792s)
Testing 1 packets sent with receiver rss -80
Received one packet!
reeha@Reeha:~/networkEng/ns-allinone-3.35/ns-3.35$

```

VIVA VOCE

Q1. What is a mobile ad hoc network and how does it work?

Ans. A mobile ad hoc network (MANET) is a type of wireless network that allows devices to communicate with each other without the need for a fixed infrastructure. In a MANET, devices can dynamically form a network and route traffic between each other as they move around.

Q2. How are mobile ad hoc networks implemented in ns-3?

Ans. In ns-3, mobile ad hoc networks are implemented as a set of classes and functions in the "adhoc" module. These classes and functions provide the necessary functionality for simulating MANETs in ns-3, including support for various routing protocols and MAC protocols.

Q3. How can I use mobile ad hoc networks in my ns-3 simulations?

Ans. To use mobile ad hoc networks in your ns-3 simulations, you will need to create instances of the appropriate ad hoc networking classes and configure them with the desired settings. You will also need to create nodes and devices that use ad hoc networking, and connect them together to form a network.

Q4. What are some limitations or challenges of using mobile ad hoc networks in ns-3 simulations?

Ans. One of the main challenges of simulating mobile ad hoc networks in ns-3 is accurately modeling the complex behavior of the network as nodes move around and the topology changes. In addition, simulating large-scale mobile ad hoc networks can be computationally intensive and may require significant resources.

Q5. Are there any alternatives to mobile ad hoc networks for wireless networking in ns-3?

Ans. Yes, there are many other wireless networking technologies that can be simulated in ns-3, such as cellular networks, Wi-Fi networks, and others. These technologies can offer different trade-offs in terms of performance and complexity, and may be more suitable for certain types of networks or applications.