Q.1)If a cellular mobile telephone system uses two 25khz simplex channel to provide full duplex Voice and control channel and having a total 36Mhz of bandwidth. Compute the number of Channels available/channel.If it uses 4 cell reuse,7 cell reuse and 12 cell reuse pattern. Also,Determine equitable distribution of control channel and voice channel if the system has k=4 and 1Mhz of the allocated spectrum is dedicated to control channels.

In [ ]:

```
import math
```

In [ ]:

```
bw = int(input("Enter the total Bandwidth of channel in MHz:"))
bw1 = bw*(10**6)
simplex_chnl = int(input("Enter the Simplex Channel Bandwidth in KHz:"))
duplex_channel = simplex_chnl*2
duplex = duplex_channel*(10**3)
duplex1 = duplex/1000
print("Channel Bandwidth is:",duplex1,"kHz/channel")
tot_channel = bw1/duplex
print("Total available  channels are:",tot_channel , "channels")
```

```
Channel Bandwidth is: 50.0 kHz/channel
Total available  channels are: 720.0 channels
```

In [ ]:

```
n=int(input("Enter the reuseable cells:"))
channel_avail=tot_channel/n
channel_avail=math.ceil(channel_avail)
print("Total number of channels available per cell:",channel_avail,"channels")

#control        channels
spectrum=int(input("Enter the frequency spectrum for control channel in MHz:"))
spectrum1=spectrum*(10**6)
S=spectrum1/duplex
S=math.ceil(S)
print("Number of control channels are",S,"out of",tot_channel,"available channels")

#Number of      control channel per     cells
con_channel=S/n
con_channel=math.ceil(con_channel)
print("Control  channels for",n,"resulable cells are:",con_channel)
voc_avail=channel_avail-con_channel
print("We can have",con_channel,"control channels and",voc_avail,"voice channels per cell")
```

```
Total number of channels available per cell: 180 channels
Number of control channels are 20 out of 720.0 available channels
Control channels for 4 resulable cells are: 5
We can have 5 control channels and 175 voice channels per cell
```

In [ ]:

```
n=int(input("Enter the reuseable cells:"))
channel_avail=tot_channel/n
channel_avail=math.ceil(channel_avail)
print("Total number of channels available per cell:",channel_avail,"channels")
#control channels
spectrum=int(input("Enter the frequency spectrum for control channel in MHz:"))
spectrum1=spectrum*(10**6)
S=spectrum1/duplex
S=math.ceil(S)
print("Number of control channels are",S,"out of",tot_channel,"available channels")
#Number of control channel per cells
con_channel=S/n
con_channel=math.ceil(con_channel)
print("Control channels for",n,"resulable cells are:",con_channel)
voc_avail=channel_avail-con_channel
print("We can have",con_channel,"control channels and",voc_avail,"voice channel per cell")
```

```
Total number of channels available per cell: 103 channels
Number of control channels are 20 out of 720.0 available channels
Control channels for 7 resulable cells are: 3
We can have 3 control channels and 100 voice channel per cell
```

In [ ]:

```
n=int(input("Enter the reuseable cells:"))
channel_avail=tot_channel/n
channel_avail=math.ceil(channel_avail)
```

```
print("Total number of channels available per cell:",channel_avail,"channels")
#control channels
spectrum=int(input("Enter the frequency spectrum for control channel in MHz:"))
spectrum1=spectrum*(10**6)
S=spectrum1/duplex
S=math.ceil(S)
print("Number of control channels are",S,"out of",tot_channel,"available channels")
#Number of control channel per cells
con_channel=S/n
con_channel=math.ceil(con_channel)
print("Control channels for",n,"resulable cells are:",con_channel)
voc_avail=channel_avail-con_channel
print("We can have",con_channel,"control channels and",voc_avail,"voice channel per cell")
```

```
Total number of channels available per cell: 60 channels
Number of control channels are 20 out of 720.0 available channels
Control channels for 12 resulable cells are: 2
We can have 2 control channels and 58 voice channel per cell
```

Q.2) Consider geographical area of a cellular system is 480Km2.A total of 910 radio channels are available for traffic handling suppose, area of a cell is 8 Km2. How many times would the cluster size of 7 have to be replicated in order to cover the entire service area? Calculate the number of channels per cell and system capacity. If the cluster size is decreased from 7 to 4 then does it result into increase in system capacity?

In [ ]:

```
import matplotlib.pyplot as plt
import math
```

In [ ]:

```
#Taking inputs
area = int(input("Enter the geographical area of the cell in square kilometers:"))
cov = int(input("Enter the coverage area of the cell in square kilometers:"))
cha = int(input("Enter the radio channels are available for traffic handling:"))
```

In [ ]:

```
#For first cell size
k = int(input("Enter the cluster size:"))
area_cov = k * cov
print("Total coverage area of the cluster is:",area_cov)
area_cell = area/area_cov
area_cell = math.ceil(area_cell)
print("The number of times that the cluster has to be replicated to cover the entire service area of cellular
cap = cha/k
cap = math.ceil(cap)
print("Cell Capacity is:",cap," channels/cell")
sys = cha * area_cell
sys = math.ceil(sys)
print("System capacity is:",sys," channels")
```

```
Total coverage area of the cluster is: 8
The number of times that the cluster has to be replicated to cover the entire service area of cellular syste
m: 60
Cell Capacity is: 910  channels/cell
System capacity is: 54600  channels
```

In [ ]:

```
#For second cell size
k1 = int(input("Enter the cluster size:"))
area_cov1 = k1 * cov
print("Total coverage area of the cluster is:",area_cov1)
area_cell1 = area/area_cov1
area_cell1 = math.ceil(area_cell1)
print("The number of times that the cluster has to be replicated to cover the entire service area of cellular
cap1 = cha/k1
cap1 = math.ceil(cap1)
print("Cell Capacity is:",cap1," channels/cell")
sys1 = cha * area_cell1
sys1 = math.ceil(sys1)
print("System capacity is:",sys1," channels")
```

```
Total coverage area of the cluster is: 24
The number of times that the cluster has to be replicated to cover the entire service area of cellular syste
m: 20
Cell Capacity is: 304  channels/cell
System capacity is: 18200  channels
```

In [ ]:

```
#For third cell size
k2 = int(input("Enter the cluster size:"))
```

```
area_cov2 = k2 * cov
print("Total coverage area of the cluster is:",area_cov2)
area_cell2 = area/area_cov2
print("The number of times that the cluster has to be replicated to cover the entire service area of cellular
cap2 = cha/k2
cap2 = math.ceil(cap2)
print("Cell Capacity is:",cap2," channels/cell")
sys2 = cha * area_cell2
sys2 = math.ceil(sys2)
print("System capacity is:",sys2," channels")
```

```
Total coverage area of the cluster is: 32
The number of times that the cluster has to be replicated to cover the entire service area of cellular syste
m: 15.0
Cell Capacity is: 228  channels/cell
System capacity is: 13650  channels
```

In [ ]:

```
#For fourth cell size
k3 = int(input("Enter the cluster size:"))
area_cov3 = k3 * cov
print("Total coverage area of the cluster is:",area_cov3)
area_cell3 = area/area_cov3
area_cell3 = math.ceil(area_cell3)
print("The number of times that the cluster has to be replicated to cover the entire service area of cellular
cap3 = cha/k3
cap3 = math.ceil(cap3)
print("Cell Capacity is:",cap3," channels/cell")
sys3 = cha * area_cell3
sys3 = math.ceil(sys3)
print("System capacity is:",sys3," channels")
```

```
Total coverage area of the cluster is: 56
The number of times that the cluster has to be replicated to cover the entire service area of cellular syste
m: 9
Cell Capacity is: 130  channels/cell
System capacity is: 8190  channels
```

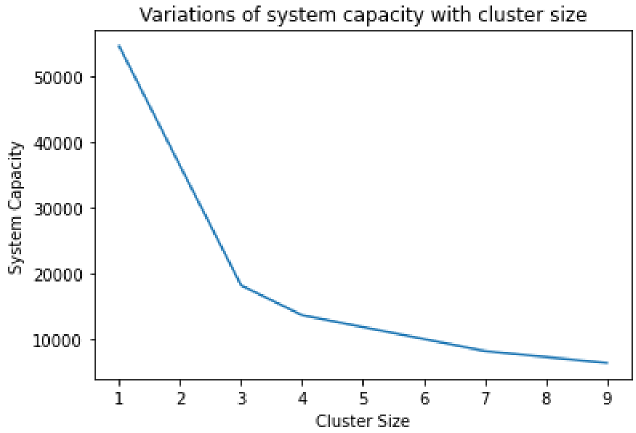
In [ ]:

```
#For fifth cell size
k4 = int(input("Enter the cluster size:"))
area_cov4 = k4 * cov
print("Total coverage area of the cluster is:",area_cov4)
area_cell4 = area/area_cov4
area_cell4 = math.ceil(area_cell4)
print("The number of times that the cluster has to be replicated to cover the entire service area of cellular
cap4 = cha/k4
cap4 = math.ceil(cap4)
print("Cell Capacity is:",cap4," channels/cell")
sys4 = cha * area_cell4
sys4 = math.ceil(sys4)
print("System capacity is:",sys4," channels")
```

```
Total coverage area of the cluster is: 72
The number of times that the cluster has to be replicated to cover the entire service area of cellular syste
m: 7
Cell Capacity is: 102  channels/cell
System capacity is: 6370  channels
```

In [ ]:

```
#Plotting the graph
m = [k, k1, k2, k3, k4]
n = [sys, sys1, sys2, sys3, sys4]
plt.plot(m,n)
plt.ylabel("System Capacity")
plt.xlabel("Cluster Size")
plt.title("Variations of system capacity with cluster size")
plt.show()
```


Variations of system capacity with cluster size
System Capacity
50000
40000
30000
20000
10000
1
2
3
4
5
6
7
8
9
Cluster Size

Name: Abhishek Singh

Roll Number: 201902034 (60)

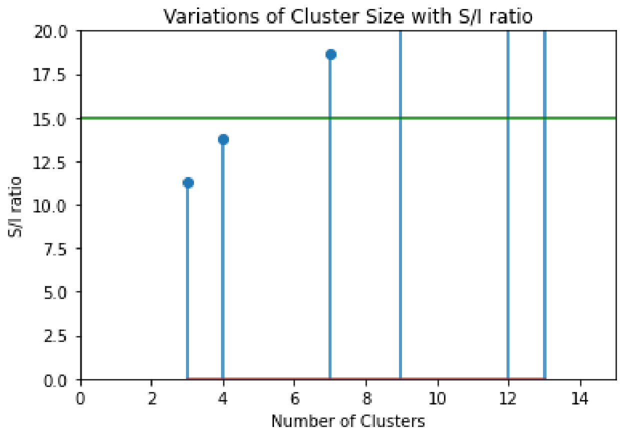
Subject: Mobile Communication System

In [ ]:
```
import math
import matplotlib.pyplot as plt
import numpy as np
```

In [ ]:
```
N = [3,4,7,9,12,13]
SI_vals = []
S_I = (int(input("Enter the value of minimum required S/I : ")))
n = (int(input("Enter the value of n : ")))
for i in N:
  q = math.sqrt(3*i)
  print("Co-channel re-use ratio is",round(q, 2))
  s_i1 = (q**n)/6
  print("Signal-to-interference is",round(s_i1, 2))
  s_i1_db = 10*math.log10(s_i1)
  print("Signal-to-interference in dB is",round(s_i1_db, 2))
  SI_vals.append(s_i1_db)
  if s_i1_db > S_I:
    print("Since this is greater than minimum required S/I N = ",i," can be used\n")
  else:
    print("Since this is less than the minimum required S/I N = ",i," cannot be used\n")
```

```
Co-channel re-use ratio is 3.0
Signal-to-interference is 13.5
Signal-to-interference in dB is 11.3
Since this is less than the minimum required S/I N =  3  cannot be used

Co-channel re-use ratio is 3.46
Signal-to-interference is 24.0
Signal-to-interference in dB is 13.8
Since this is less than the minimum required S/I N =  4  cannot be used

Co-channel re-use ratio is 4.58
Signal-to-interference is 73.5
Signal-to-interference in dB is 18.66
Since this is greater than minimum required S/I N =  7  can be used

Co-channel re-use ratio is 5.2
Signal-to-interference is 121.5
Signal-to-interference in dB is 20.85
Since this is greater than minimum required S/I N =  9  can be used

Co-channel re-use ratio is 6.0
Signal-to-interference is 216.0
Signal-to-interference in dB is 23.34
Since this is greater than minimum required S/I N =  12  can be used

Co-channel re-use ratio is 6.24
Signal-to-interference is 253.5
Signal-to-interference in dB is 24.04
Since this is greater than minimum required S/I N =  13  can be used
```

In [ ]:
```
plt.stem(N,SI_vals,use_line_collection = True)
plt.axhline(y = 15,color = 'g')
plt.ylabel("S/I ratio")
plt.xlabel("Number of Clusters")
plt.title("Variations of Cluster Size with S/I ratio")
plt.xlim([0, 15]);
plt.ylim([0, 20]);
plt.show()
```



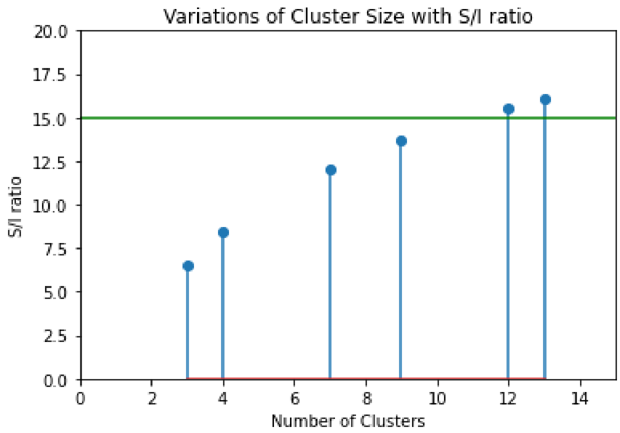
In [ ]:

```
N = [3,4,7,9,12,13]
SI_vals = []
S_I = (int(input("Enter the value of minimum required S/I : ")))
n = (int(input("Enter the value of n : ")))
for i in N:
  q = math.sqrt(3*i)
  print("Co-channel re-use ratio is",round(q, 2))
  s_i1 = (q**n)/6
  print("Signal-to-interference is",round(s_i1, 2))
  s_i1_db = 10*math.log10(s_i1)
  print("Signal-to-interference in dB is",round(s_i1_db, 2))
  SI_vals.append(s_i1_db)
  if s_i1_db > S_I:
    print("Since this is greater than minimum required S/I N = ",i," can be used\n")
  else:
    print("Since this is less than the minimum required S/I N = ",i," cannot be used\n")
```

```
Co-channel re-use ratio is 3.0
Signal-to-interference is 4.5
Signal-to-interference in dB is 6.53
Since this is less than the minimum required S/I N =  3  cannot be used

Co-channel re-use ratio is 3.46
Signal-to-interference is 6.93
Signal-to-interference in dB is 8.41
Since this is less than the minimum required S/I N =  4  cannot be used

Co-channel re-use ratio is 4.58
Signal-to-interference is 16.04
Signal-to-interference in dB is 12.05
Since this is less than the minimum required S/I N =  7  cannot be used

Co-channel re-use ratio is 5.2
Signal-to-interference is 23.38
Signal-to-interference in dB is 13.69
Since this is less than the minimum required S/I N =  9  cannot be used

Co-channel re-use ratio is 6.0
Signal-to-interference is 36.0
Signal-to-interference in dB is 15.56
Since this is greater than minimum required S/I N =  12  can be used

Co-channel re-use ratio is 6.24
Signal-to-interference is 40.59
Signal-to-interference in dB is 16.08
Since this is greater than minimum required S/I N =  13  can be used
```

In [ ]:

```
plt.stem(N,SI_vals,use_line_collection = True)
plt.axhline(y = 15,color = 'g')
plt.ylabel("S/I ratio")
plt.xlabel("Number of Clusters")
plt.title("Variations of Cluster Size with S/I ratio")
plt.xlim([0, 15]);
plt.ylim([0, 20]);
plt.show()
```



Q.2) Consider a cellular system with S/I ratio of 18dB. The frequency reuse factor is N=7. Calculate the worst case for signal to co-channel interference ratio. Is the frequency reuse factor 7 still being acceptable? If not, what is it? Assume path loss exponent as 4.

In [ ]:

```
path_loss = int(input("Enter the value of Path Loss Exponent:"))
freq_reuse = int(input("Enter the frequency reuse factor:"))
Q = round(((3*freq_reuse)**0.5),2)
print("Hence, the value of Co-channel Reuse Ratio for N=",freq_reuse,"is:",Q)
S = (((Q)**path_loss)/6) #six cochannel
S_dB = round(10*math.log(S,10),2)
print("Hence, S/I ratio is:",S_dB,"dB")
if(S_dB > 18):
    print("The design is accepted for Path Loss Exponent,η = ",path_loss)
else:
    print("The design is not accepted for Path Loss Exponent,η = ",path_loss)
```

```
Hence, the value of Co-channel Reuse Ratio for N= 7 is: 4.58
Hence, S/I ratio is: 18.65 dB
The design is accepted for Path Loss Exponent,η =  4
```

Q.3) If the acceptable S/I is now 20 dB, will the cluster size determined in problem 2 be adequate? If not, then what should be the cluster size?

In [ ]:

```
path_loss = int(input("Enter the value of Path Loss Exponent:"))
freq_reuse = int(input("Enter the frequency reuse factor:"))
Q = round(((3*freq_reuse)**0.5),2)
print("Hence, the value of Co-channel Reuse Ratio for N=",freq_reuse,"is:",Q)
S = round((((Q)**path_loss)/6),2) #six cochannel
S_dB = round(10*math.log(S,10),2)
print("Hence, S/I ratio is:",S_dB,"dB")

if(S_dB > 20):
    print("The design is accepted for Path Loss Exponent,η = ",path_loss)
else:
    print("The design is not accepted for Path Loss Exponent,η = ",path_loss)
```

```
Hence, the value of Co-channel Reuse Ratio for N= 7 is: 4.58
Hence, S/I ratio is: 12.04 dB
The design is not accepted for Path Loss Exponent,η =  3
```

Q.4) Determine the S/I ratio at the mobile receiver located at the boundary of its omnidirectional operating cell, under the influence of interfering signals from six cochannel interfering cells in the first tier in a cellular system designed with N=7, N=9 and N= 12

In [ ]:

```
N=[7,9,12]
SI_plt=[]
si=18 #S/I in dB
SI=10**(si/10)
n=int(input("Enter the path loss exponent:\n "))
for i in N :
    io=(i-1)
    Q=round(np.sqrt(3*i),2)
    print("The Frequency reuse factor(Q) is {}".format(Q))
    SI_new=round((Q**n)/io,2)
    SIdB=round(10*np.log10(SI_new),2)
    print("The Value of S/I for cluster size {} is {} dB".format(i,SIdB))
    SI_plt.append(SIdB)
    if SIdB>si :
```

```
        print("The design is acceptable\n")
    else :
        print("The design is rejected\n")
plt.stem(N,SI_plt,use_line_collection = True)
plt.axhline(y = 15,color = 'r')
plt.ylabel("S/I ratio")
plt.xlabel("Number of Clusters")
plt.title("Variations of Cluster Size with S/I ratio")
plt.xlim([0, 15]);
plt.ylim([0, 25]);
plt.show()
```

```
The Frequency reuse factor(Q) is 4.58
The Value of S/I for cluster size 7 is 18.65 dB
The design is acceptable

The Frequency reuse factor(Q) is 5.2
The Value of S/I for cluster size 9 is 19.61 dB
The design is acceptable

The Frequency reuse factor(Q) is 6.0
The Value of S/I for cluster size 12 is 20.71 dB
The design is acceptable
```

MCS Lab 3
Abhishek Singh 201902034(60)

In [ ]:
```python
import math
import numpy as np
```

In [ ]:
```python
def erlang(C):
    erlang_dict={"2":"0.105",
    "4":"0.701",
    "5":"1.13",
    "10":"3.96",
    "20":"11.1",
    "24":"14.2",
    "40":"27.3",
    "70":"53.7",
    "100":"80.9"}
    if str(C) in erlang_dict:
        return erlang_dict[str(C)]
    else:
        return print("No")

C = int(input("Enter Number of truncated Channels ")) # =5
Au = 0.1
GOS = 0.005
A = float(erlang(C))
tot_users = A/Au
print("Number of users = ",str(math.ceil(tot_users)))
```

```
Number of users =  12
```

A hexagonal cell within a 4-cell system has a radius of 1.387 km. A total of 60 channels are used within the entire system. If the load per user is 0.029 Erlangs, and λ = 1call / hour, compute the following for an Erlang C system that has a 5% probability of a delayed call: (a) How many users per square kilometer will this system support? (a) What is the probability that a delayed call will have to wait for more than 10s? (c) What is the probability that a call will be delayed for more than 10 seconds?

In [ ]:
```python
r = float(input("Enter the radius of cell in km :")) # =1.387
area = 2.598*r*r
N = int(input("Enter the no of cells in a cluster :")) # =4
tot_ch = int(input("Enter total no of channels :")) # =60

c = tot_ch/4
A = 9
Au = 0.029
lam = 1
t = 10
delay_prob = 0.05
user_per_sqkm = (A/Au)/area
print("User per square km = ",user_per_sqkm)

hold_time = Au*3600/lam
hold_prob = math.exp(-(c-A)*t/hold_time)
print("Probability that cell call will have to wait for 10 sec = ",hold_prob)
print("Probability that cell call will get delayed by 10 sec = ",hold_prob*delay_prob)
```

```
User per square km =  62.09440109773648
Probability that cell call will have to wait for 10 sec =  0.5628665888428616
Probability that cell call will get delayed by 10 sec =  0.02814332944214308
```

A certain city has an area of 1,300 square miles and is covered by a cellular system using a 7-cell reuse pattern. Each cell has a radius of 4 miles and the city is allocated 40 MHz of spectrum with a full duplex channel bandwidth of 60 kHz. Assume a GOS of 2% for an Erlang B system is specified. If the offered traffic per user is 0.03 Erlangs, compute (a) the number of cells in the service area, (b) the number of channels per cell, (c) traffic intensity of each cell, (d) the maximum carried traffic; (e) the total number of users that can be served for 2% GOS, (f') the number of mobiles per channel, and (g) the theoretical maximum number of users that could be served at one time by the system

In [ ]:
```python
Cov_area = int (input("Enter the total coverage area (in sq. miles) :") ) # =1300
R = int(input("Enter the cell radius (in miles):") ) # =4
freq_reuse = int (input ("Enter the frequency reuse factor:") ) # =7
bw = int(input ("Enter the bandwidth (in kHz):") ) # =60
freq = int (input("Enter the frequency of the spectrum (in MHz) :") ) # =40
Area = np.round( (2.5981*R*R) , 4)

print ("Area covered per cell is:", Area, "sq.miles")
N = int( (Cov_area/Area) )
print("Hence, total number of cells in a cluster are:", N, "cells")
tot_channels = int((freq*(10**6) / (bw*(10**3) ) ) )
print("Hence, total number of channels are:", tot_channels, "channels")
C_c = int((freq* (10**6)/(bw*(10**3) *freq_reuse) ) )
print ("Hence, total number of channels per cell are: ", C_c, "channels/cell")

traffic = 84 #From Erlang B chart with C and GoS of 2%
print ("Hence, Traffic Intensity per cell is:", traffic, "Erlangs/cell")
Max_traffic = int(N*traffic)
print( "Hence, maximum carried traffic is: ", Max_traffic, "Erlangs")
tot_users = int(Max_traffic/0.03) #0.03 is given in question
print ("Hence, he total number of users are: ", tot_users, "users")
mob_per_channel = int(tot_users/tot_channels)

print ("Hence, total number of mobiles per channel are:", mob_per_channel, "mobiles/channel")
user_at_a_time = int(C*N)
print( "Hence, theoretical maximum number of users that could be served at one time by the system is: ", user_at_a_time, "users")
```

```
Area covered per cell is: 41.5696 sq.miles
Hence, total number of cells in a cluster are: 31 cells
Hence, total number of channels are: 666 channels
Hence, total number of channels per cell are:  95 channels/cell
Hence, Traffic Intensity per cell is: 84 Erlangs/cell
Hence, maximum carried traffic is:  2604 Erlangs
Hence, he total number of users are:  86800 users
Hence, total number of mobiles per channel are: 130 mobiles/channel
Hence, theoretical maximum number of users that could be served at one time by the system is:  155 users
```

MCS Lab 4
Abhishek Singh 201902034(60)

The radius of the split cell is one half of that of the cell before splitting. Show that the overall system capacity increases by four times and power decays at function of square

In [ ]:

```
nC = int(input("Enter no. of cells :")) # =7
nCh = int(input("Enter no. of channels per cells :")) # =10
rFac = float(input("Enter the radius multiplication factor :")) # =0.5
n = int(input("Enter the path loss exponent :")) # =4
sys_cap_old = nC*nCh
sys_cap_new = nC*nCh*(rFac**2)**-1

print("Old system capacity is {} \nNew system capacity is {}".format(sys_cap_old,sys_cap_new))
print("Thus, it can be seen that the channel capacity increases with channel spiltting")
```

```
Old system capacity is 70
New system capacity is 280.0
Thus, it can be seen that the channel capacity increases with channel spiltting
```

Determine (a) The channel capacity for a cellular system service area comprised of seven macrocell with 16 channels per macrocell (b) Channel capacity if each macrocell is split into four minicells (c) Channel capacity if each minicell is further split into four microcells

In [ ]:

```
m = int(input("no of macro cells")) # =7
p = int(input("no of channels per macrocell")) # =16
C = m*p
print("The total channel capacity of the cellular system is", C)

n = int(input("no of minicells per macrocell")) # =4
C = m*p*n
print("The total channel capacity after splitting into minicells is", C)

q = int(input("no of microcells per minicell")) # =4
C = m*p*n*q
print("The total channel capacity after splitting into microcells is", C)
```

```
The total channel capacity of the cellular system is 112
The total channel capacity after splitting into minicells is 448
The total channel capacity after splitting into microcells is 1792
```

Consider a cellular system with a radius of 2 Km which is split into smaller cells with a radius of 1 Km. Let each cell site be assigned 120 channels regardless of the cell size. How many times will the number of channels contained in a 6X6 Km2 area centered around small cell ‘S’ be increased with cell splitting as compared to without cell splitting?

In [ ]:

```
n = int(input("No. of large cells within the given area")) # =4
Nch = int(input("No. of channels in each large cell")) # =120
r1 = int(input("Radius of large cell")) # =2
r2 = int(input("Radius of split cell")) # =1

tot_Nch_before_split = n*Nch
no_split_cells = (r1/r2)**2 * n - 1
tot_Nch_after_split = Nch*no_split_cells

print("No. of Channels before Cell Splitting {}\nNo of Channels after Cell Splitting {}".format(tot_Nch_before_sp
```

```
No. of Channels before Cell Splitting 480
No of Channels after Cell Splitting 1800.0
```

Abhishek Singh
MCS lab 05
201902034(60)

Case 1:
A cellular system is designed with a directional antenna cellular configuration. A cluster pattern
of size 7 is deployed. Let the mobile receiver be located at the boundary of its operating cell, and be
under the influence of interfering signals from two co-channel interfering cells in the first tier. Compute
the worst-case signal to cochannel interference ratio S/I at the mobile receiver. If the S/I value for a
practical system requires 6dB higher than the theoretical value of 18 dB then comment on the results
obtained. Assume the path loss exponent as 4.
Case 2: Solve for N=4
Case 3: Solve for 6-sector N=7 one co-channel interfering cells in the first tier.
Case 4: Solve for 6-sector N=4 one co-channel interfering cells in the first tier

In [ ]:

```
import math

def sector6(N, eta, Q):
    SI_6 = 1/ ((Q+0.7)**(-eta))
    SI_6_db = 10*math. log10(SI_6)

    return SI_6_db
def sector3(N, eta, Q) :
    SI_3 = ((Q**-eta)+((Q+0.7)**-eta))**-1
    SI_3_db = 10*math.log10(SI_3)
    return SI_3_db

print ("Case 1: N = 7, eta = 4, for 3 Sectors: ")
N = 7
eta = 4
Q = (3*N) **0.5
print ("S/I ratio in dB is: ", sector3(N, eta, Q), "\n")

print ("Case 2: N = 4, eta = 4, for 3 Sectors: ")
N = 4
eta = 4
Q = (3*N)**0.5
print ("S/I ratio in dB is: ", sector3(N, eta, Q), "\n")

print ("Case 3: N = 7, eta = 4, for 6 Sectors: ")
N = 7
eta = 4
Q = (3*N)**0.5
print("S/I ratio in dB is: ", sector6(N, eta, Q),"\n")

print ("Case 4: N = 4, eta = 4, for 6 Sectors: ")
N = 4
eta = 4
Q = (3*N)**0.5
print("S/I ratio in dB is: ", sector6(N, eta, Q), "\n")
```

```
Case 1: N = 7, eta = 4, for 3 Sectors:
S/I ratio in dB is:  24.495603468889296

Case 2: N = 4, eta = 4, for 3 Sectors:
S/I ratio in dB is:  19.884131977262278

Case 3: N = 7, eta = 4, for 6 Sectors:
S/I ratio in dB is:  28.91382915403679

Case 4: N = 4, eta = 4, for 6 Sectors:
S/I ratio in dB is:  24.780852756259875
```

Q2)
A TDMA digital cellular system is designed to accept a S/I value of 15 db. Find the optimum value of N
for a) Omnidirectional antenna design b) Six-sector 600 directional design Comment on the use of cell
sectoring in this case

In [ ]:

```
N1 = 7 # 3 sector
N2 = 4 # 6 sector
tf_ch = 312
print ("System A => N1 = 7, Three Sectors")
print ("System B => N2 = 4, Six Sectors\n")

print("For System A")
print ("Number of traffic channels per cell =", round(tf_ch/N1) )
print("Number of Traffic channels per sector =", round((tf_ch/N1)/3), "\n")
print("For System B")
print ("Number of traffic channels per cell =", round(tf_ch/N2) )

print ("Number of Traffic channels per sector =", round((tf_ch/N2)/6), "\n")
#Comparison
print("It is seen that N=7 pattern offers higher")
print ("Spectrum efficiency than N = 4 pattern in the above system configuration")
```

```
System A => N1 = 7, Three Sectors
System B => N2 = 4, Six Sectors

For System A
Number of traffic channels per cell = 45
Number of Traffic channels per sector = 15

For System B
Number of traffic channels per cell = 78
Number of Traffic channels per sector = 13

It is seen that N=7 pattern offers higher
Spectrum efficiency than N = 4 pattern in the above system configuration
```

Abhishek Singh
MCS lab 06
201902034(60)

Q)
Consider a transmitter which radiates a sinusoidal carrier frequency of 1850 MHz. For a vehicle moving 60 mph, compute the received carrier frequency if the mobile is moving
(a) directly towards the transmitter
(b) directly away from the transmitter
(c) in a direction which is perpendicular to the direction of arrival of the transmitted signal

In [ ]:

```
import math as mt
```

In [ ]:

```
c = 3e8
fc = (float(input("Enter Carrier Frequery in Hz: ")))*10**6
#fc = 1850#e6
#v = 60
v = float(input("Enter Vehicle Speed in mph: "))
vc = v*0.447
lamC = c/fc
x=90
fd = vc/lamC
ft = fc + fd
fa = fc - fd

#towards
print("For a vehicle moving towards transmitter, Doppler Shift is positive")
print("Hence, F = Fc + Fd =", ft, "Hz")
#away
print("For a vehicle moving towards transmitter, Doppler Shift is negative")
print ("Hence, F = Fc - Fd =", fa, "Hz")
#perpendicular to signal
print("Vehicle perpendicular to direction of signal, No Doppler Shift")
print("Hence, F = Fc + Fd = Fc + 0 =", fc, "MHz")
```

```
For a vehicle moving towards transmitter, Doppler Shift is positive
Hence, F = Fc + Fd = 1850000165.39 Hz
For a vehicle moving towards transmitter, Doppler Shift is negative
Hence, F = Fc - Fd = 1849999834.61 Hz
Vehicle perpendicular to direction of signal, No Doppler Shift
Hence, F = Fc + Fd = Fc + 0 = 1850000000.0 MHz
```

Abhishek Singh
MCS lab 06
201902034(60)

Implement following outdoor models for path loss estimation if distance d is varied from 1 to 20 Km in step size of 1Km
f-center frequency in Hz (1500-2000MHz)
f1-center frequency1 in Hz (1800MHz)
hb- base station height (upto 50)
hm- mobile station height (upto 5)
a = 3.2*log(11.75*hm)^2 - 4.97
a hm= (1.1log(fc)-0.7)hm-(1.56log(fc)-0.8)
'IMT 2000 model'
PL = 32.4+20*log(d)+20*log(f)-10log(Gt)-10log(Gr)
'Cost 231 model suburban area'
PL=46.3+(33.9*log(f1)-(13.82*log(hb)-(a*hm)+(44.9-6.55*log(hb)*log(d))))
'Cost 231 model urban area
PL=46.3+(33.9*log(f1)-(13.82*log(hb)-(a*hm)+(44.9-6.55*log(hb)*log(d))))
'Hata model urban area'
PL=69.55+(26.16*log(f))+(44.9-6.55*log(hb))*log(d)-13.82*log(hb)-a
'Hata model suburban area'
PL= pl4-2*log((f/28)^2)-5.4
'Hata model rural area'
PL= pl4-(4.78*(log(f))^2)+(18.33*log(f))-40.94

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (10,10)
```

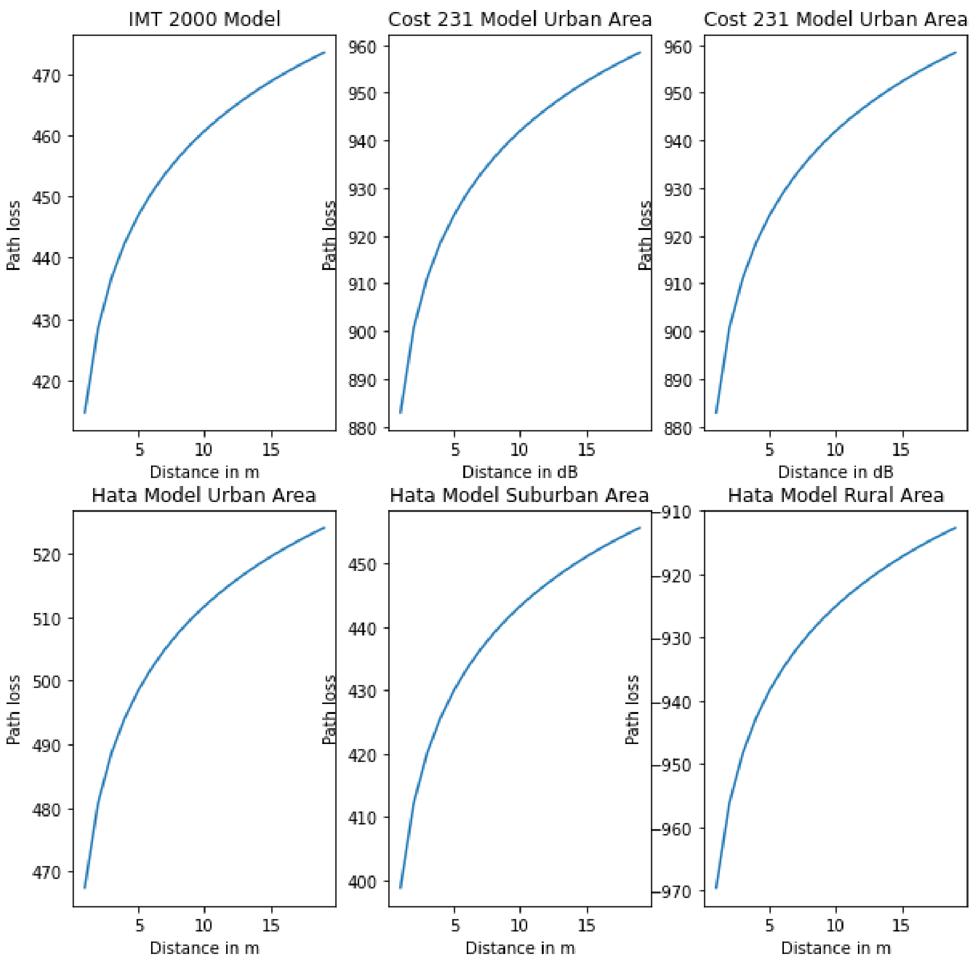
In [ ]:

```
f = int(input('Enter the center frequency= '))
f1 = int(input('Enter the center frequency = '))
d = range(1,20,1)
hm = 5
hb = 50
a = 3.2*np.log(11.75*hm)**2 - 4.97

pl = 32.4+20*np.log(d)+20*np.log(f);
plt.subplot(2,3,1)
plt.plot(d,pl)
plt.xlabel('Distance in m')
plt.ylabel('Path loss')
plt.title('IMT 2000 Model')

pl2=46.3+(33.9*np.log(f1)-(13.82*np.log(hb)-(a*hm)+(44.9-6.55*np.log(hb)*np.log(d))));
plt.subplot(2,3,2)
plt.plot(d,pl2)
plt.xlabel('Distance in dB')
plt.ylabel('Path loss')
plt.title('Cost 231 Model Urban Area')

pl3=46.3+(33.9*np.log(f1)-(13.82*np.log(hb)-(a*hm)+(44.9-6.55*np.log(hb)*np.log(d))));
plt.subplot(2,3,3)
plt.plot(d,pl3)
plt.xlabel('Distance in dB')
plt.ylabel('Path loss')
plt.title('Cost 231 Model Urban Area')

pl4=69.55+(26.16*np.log(f))+(44.9-6.55*np.log(hb))*np.log(d)-13.82*np.log(hb)-a;
plt.subplot(2,3,4)
plt.plot(d,pl4)
plt.xlabel('Distance in m')
plt.ylabel('Path loss')
plt.title('Hata Model Urban Area')
```

```python
pl5= pl4-2*np.log((f/28)**2)-5.4;
plt.subplot(2,3,5)
plt.plot(d,pl5)
plt.xlabel('Distance in m')
plt.ylabel('Path loss')
plt.title('Hata Model Suburban Area')

pl6= pl4-(4.78*(np.log(f))**2)+(18.33*np.log(f))-40.94;
plt.subplot(2,3,6)
plt.plot(d,pl6)
plt.xlabel('Distance in m')
plt.ylabel('Path loss')
plt.title('Hata Model Rural Area')
#plt.tight_layout()
```

Out[ ]:

```
Text(0.5, 1.0, 'Hata Model Rural Area')
```



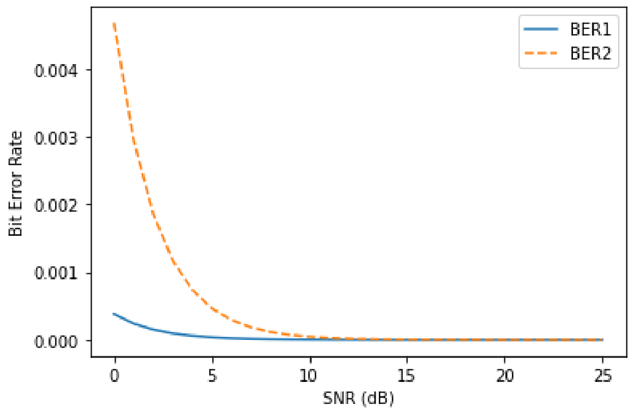
Abhishek Singh 60 201902034 lab_08

```
import math
import matplotlib.pyplot as plt
import numpy as np
```

```
def nCr(n, r):
    return (math.factorial(n)/(math.factorial(r)*math.factorial(n - r)))
#nCr(10,3)
```

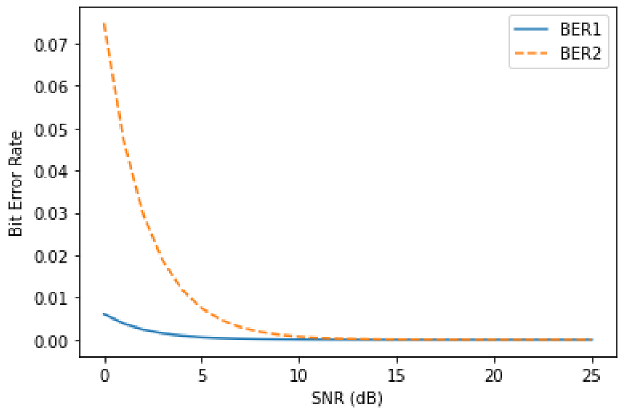
```
L=3
N=256
SNR_db = np.arange(0,26)
SNR = np.power(10,(SNR_db/10))
a = nCr(5,3)
b = 2*N*SNR
BER1 = np.power((a*(1/b)),2)

L=4
a = nCr(7,4)
b = 2*N*SNR
BER2 = np.power((a*(1/b)),2)

print(SNR)

plt.plot(SNR_db,BER1)
plt.plot(SNR_db,BER2,"--")
plt.xlabel("SNR (dB)")
plt.ylabel('Bit Error Rate')
plt.legend(["BER1", "BER2"])
plt.show()
```

```
[  1.           1.25892541   1.58489319   1.99526231   2.51188643
   3.16227766   3.98107171   5.01187234   6.30957344   7.94328235
  10.          12.58925412  15.84893192  19.95262315  25.11886432
  31.6227766   39.81071706  50.11872336  63.09573445  79.43282347
 100.         125.89254118 158.48931925 199.5262315  251.18864315
 316.22776602]
```




```
L=3
N=64
SNR_db = np.arange(0,26)
SNR = np.power(10,(SNR_db/10))
a = nCr(5,3)
b = 2*N*SNR
BER1 = np.power((a*(1/b)),2)

L=4
a = nCr(7,4)
b = 2*N*SNR
BER2 = np.power((a*(1/b)),2)

print(SNR)

plt.plot(SNR_db,BER1)
plt.plot(SNR_db,BER2,"--")
plt.xlabel("SNR (dB)")
plt.ylabel('Bit Error Rate')
plt.legend(["BER1", "BER2"])
plt.show()
```

```
[  1.           1.25892541   1.58489319   1.99526231   2.51188643
   3.16227766   3.98107171   5.01187234   6.30957344   7.94328235
  10.          12.58925412  15.84893192  19.95262315  25.11886432
  31.6227766   39.81071706  50.11872336  63.09573445  79.43282347
 100.         125.89254118 158.48931925 199.5262315  251.18864315
 316.22776602]
```



In [ ]:

```
def Calculate():
    #  To compute number of encoded Class Ia bits
    UncodedClassIa_Bits = int(input("Number of uncoded Class Ia bits = "))
    Crc_Bits = int(input("Number of CRC bits = "))
    EncodedClassIa_Bits = UncodedClassIa_Bits + Crc_Bits
    print("Number of encoded Class Ia bits = {} + {} = {} bits".format(UncodedClassIa_Bits, Crc_Bits, EncodedClassIa_Bits ))

    #  To compute number of encoded Class Ib bits
    UncodedClassIb_Bits = int(input("Number of uncoded Class Ib bits = "))
    Tail_Bits = int(input("Number of tail bits = "))
    EncodedClassIb_Bits = UncodedClassIb_Bits + Tail_Bits
    print("Number of encoded Class Ib bits = {} + {} = {} bits".format(UncodedClassIb_Bits, Tail_Bits, EncodedClassIb_Bits))

    #  To compute number of concatenation of encoded Class Ia + Class Ib bits
    Concatenated_Bits = EncodedClassIa_Bits + EncodedClassIb_Bits
    print("Therefore, number of concatenated bits = {} bits + {} bits = {} bits".format(EncodedClassIa_Bits, EncodedClassIb_Bits, Concatenated_Bits))

    #  To compute number of convolutionally encoded bits
    FECCoderRate_Convolutional = float(input("FEC coder rate of convolutional encoder = "))
    ConvolutionallyEncoded_Bits = (1 / FECCoderRate_Convolutional) * Concatenated_Bits
    print("Number of convolutionally encoded bits = {} × {} bits = {} bits".format((1 / FECCoderRate_Convolutional), Concatenated_Bits, ConvolutionallyEncoded_Bits))

    #  To compute number of encoded bits to be transmitted
    ClassII_Bits = int(input("Number of Class II bits = "))
    Encoded_BitsToTx = ConvolutionallyEncoded_Bits + ClassII_Bits
    print("Number of encoded bits to be transmitted = {} + {} = {} bits".format(ConvolutionallyEncoded_Bits, ClassII_Bits, Encoded_BitsToTx))

    #  To determine achievable gross channel data rate
    Duration_Tx = int(input("Duration of transmission = "))
    Gross_ChannelBitRate = Encoded_BitsToTx / Duration_Tx
    print("Therefore, gross channel bit rate = {} / {} kbps".format(Encoded_BitsToTx, Duration_Tx))
    print("                                  = {} kbps".format(Gross_ChannelBitRate))

if __name__ == "__main__":
  Calculate()
```

```
Number of encoded Class Ia bits = 50 + 3 = 53 bits
Number of encoded Class Ib bits = 132 + 4 = 136 bits
Therefore, number of concatenated bits = 53 bits + 136 bits = 189 bits
Number of convolutionally encoded bits = 2.0 × 189 bits = 378.0 bits
Number of encoded bits to be transmitted = 378.0 + 78 = 456.0 bits
Therefore, gross channel bit rate = 456.0 / 20 kbps
                                  = 22.8 kbps
```

Abhishek Singh 60 201902034

In [ ]:

```
Bc = 1250 # kHz
Rb = 9.6 # kbps
Sr_min_dB = 3 #dB
Sr_min_rat = round(10**(Sr_min_dB/10))
M_max = (Bc/Rb)*(1/Sr_min_rat)
print("The maximum no. of simultaneous users are",round(M_max),"users")

Sr_max_dB = 9 #dB
Sr_max_rat = round(10**(Sr_max_dB/10))
M_min = (Bc/Rb)*(1/Sr_max_rat)
print("The minimum no. of simultaneous users are",round(M_min),"users")
```

```
The maximum no. of simultaneous users are 65 users
The minimum no. of simultaneous users are 16 users
```