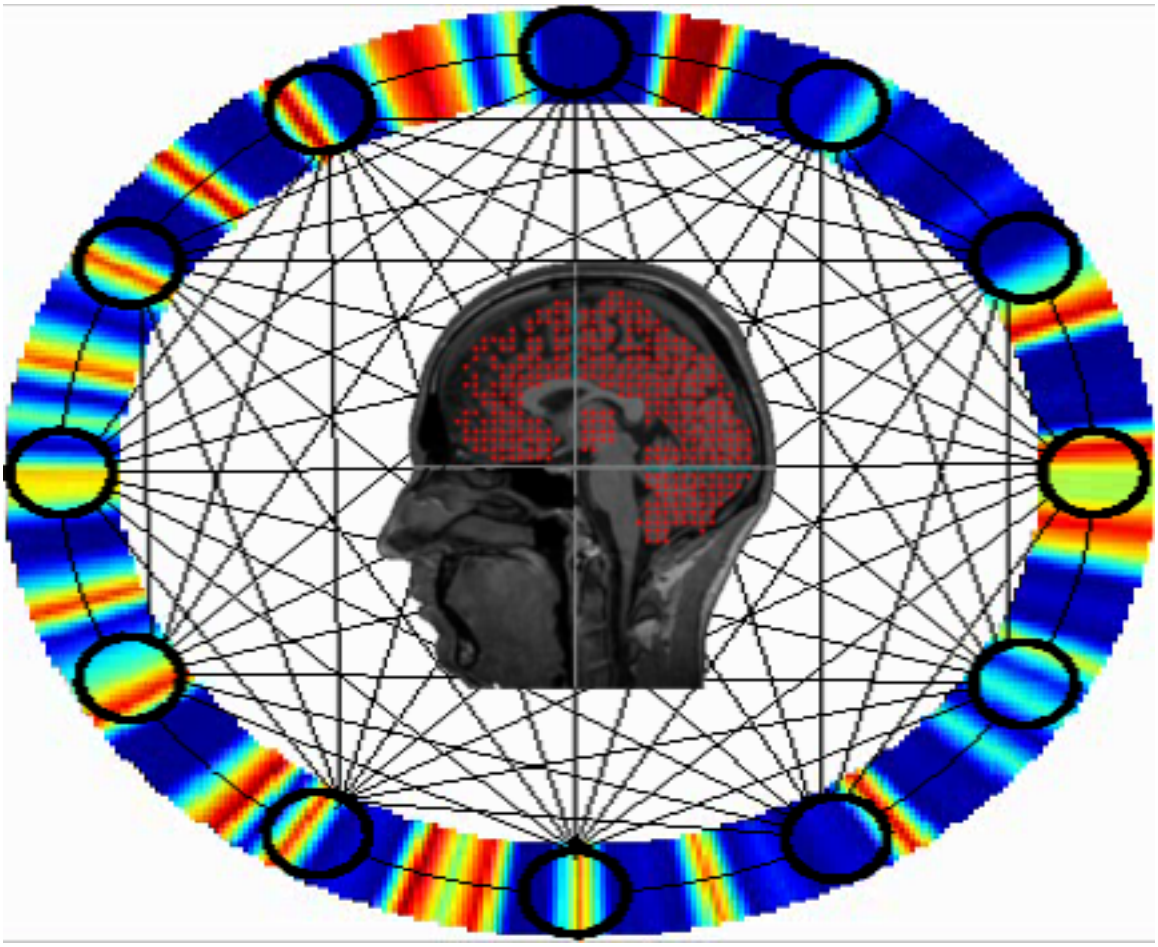
Modelling and Simulation File



Varun Gupta

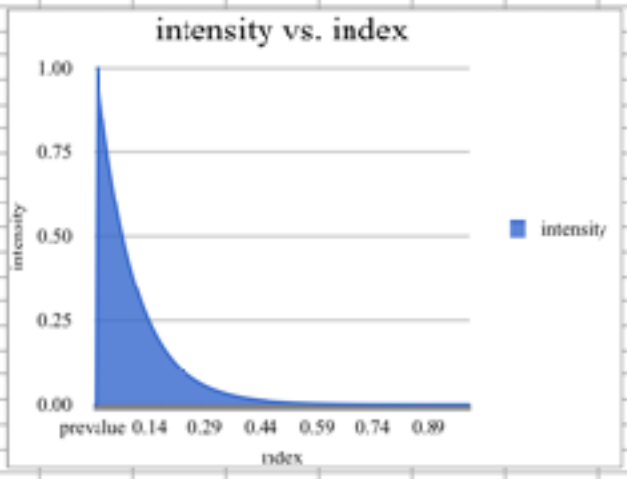
20UIT2618

List of Practicals

1. *Simulate exponential decay using MS-Excel*
2. *Simulate chi Square test*
3. *Simulate ks test*
4. *Generate random numbers using mid square method*
5. *Implement a linear congruential random number generator*
6. *Simulate monte carlo simulation*
7. *Simulate single server queueing system*
8. *Assignment-1*
9. *Assignment-2*

Transient-single-exponential

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1		**Simulation of an Exponential Decay**													
2															
3		Lifetime (τ):	1.00	ms											
4		Total time:	10.00	ms											
5		Initial intensity (I0):	1.00												
6															
7															
8															
9															
10															
11		Enter an x value to find y at that time.													
12		x value:	1												
13		y value:	3.6?9E-01												
14															
15															
16															
17															
18															
19															
20															
21															
22															
23															
24															



Chi-Square

```
from scipy import stats
import numpy as np
```

```
print(stats.chisquare([16, 18, 16, 14, 12, 
12]))
print(stats.chisquare([16, 18, 16, 14, 12, 
12], f_exp=[16, 16, 16, 16, 16, 8]))
```

```
Output:
Power_divergenceResult(statistic=2.0, 
pvalue=0.8491450360846096)
Power_divergenceResult(statistic=3.5, 
pvalue=0.623876277495822)
 ? > ? > ? ~/Desktop/Modelling-Lab ▶ ?  ? 
master ▶
```

KS Test

```
from scipy import stats
import numpy as np


x = np.linspace(-15, 15, 9)
print(stats.kstest(x, 'norm'))
```

Output:
KstestResult(statistic=0.444356027159
2436, pvalue=0.038850142705171 16)

Mid Square

```
import math

seed = input("Enter the seed number\n")
l = len(seed)
seed = int(seed)
n = int(input("Enter the count of numbers to
generate\n"))

print("The output numbers are: ")

div = round(math.pow(10, l/2))
rem = round(math.pow(10, l))

for i in range(n):
    seed = math.floor(seed*seed/div) % rem
    print(seed)
```

```
Output:
Enter the seed number
4662
Enter the count of numbers to generate
5
The output numbers are:
7342
9049
```

8844
2163
6785

Linear Congruential Generator

```
x = int(input("Please enter the seed value\n"))
a = int(input("Please enter the value of a\n"))
c = int(input("Please enter the value of c\n"))
m = int(input("Please enter the value of m\n"))
```

```
if(m < 0):
    print("Modulus negative")
    exit(0)
if(a ⩽ 0 or a ⩾ m):
    print("a out of bounds")
    exit(0)
if(c < 0 or c ⩾ m):
    print("c out of bounds")
    exit(0)
if(x < 0 or x ⩾ m):
    print("Seed out of bounds")
    exit(0)
```

```
n = int(input("Please enter the count of number you
want to generate\n"))
```

```
for i in range(n):
    x = (a*x + c) % m
    print(x)
```

```
Output:
Please enter the seed value
5
Please enter the value of a
4
Please enter the value of c
3
Please enter the value of m
10
Please enter the count of number you want to generate
10
3
5
3
5
3
5
3
5
3
5
```

Random Walk-monte carlo

```
""" Simulates a random walk """
import random


def dirn(pos):
    direction = None
    if pos > 0:
        direction = "up"
    else:
        direction = "down"
    return direction


T = int(input("Number of Samples?\n"))
n = int(input("Number of steps you want to simulate?\n"))
p = float(input("Odds of Going up?\n"))*100

vals = []

for t in range(0, T):

    pos = 0

    for i in range(n):
        if random.randint(1, 100) <= p:
            pos += 1
        else:
```

```
            pos -= 1

    vals.append(pos)

    direction = dirn(pos)
    pos = abs(pos)
    print("Final position is " + str(pos) + " " + direction)

avg = 0
for el in vals:
    avg += el
avg /= T
avg = round(avg)

var = 0
for el in vals:
    var += ((el-avg)*(el-avg))
var /= n
var = round(var)

print("\nThe mean of the outputs is: " + str(avg) + " " +
dirn(avg))
print("The variance of the output is " + str(var))
```

```
Output:
Number of Samples?
5
Number of steps you want to simulate?
100
Odds of Going up?
0.51
Final position is 8 up
Final position is 2 up
Final position is 10 up
Final position is 2 up
```

```
Final position is 4 down
```

```
The mean of the outputs is: 4 up
The variance of the output is 1
```

Single Server Queueing system

```
import random


class Job:
    def __init__(self, at, st):
        self.at = at
        self.st = st
        self.wt = None
        self.et = None

    def arrive(self, q, et):
        self.et = et
        q.append(self)

    def depart(self, t):
        self.wt = t-self.at


Q = []

t = int(input("Time for which you want to simulate\n"))
n = int(input("Number of Jobs in the system\n"))
x = int(input("Enter the max waiting time\n"))
jobs = []
```

```
for i in range(n):
    jobs.append(Job(random.randint(0, t), random.randint(0, x)))

t = 0

count = 0
while(count < n):
    for j in jobs:
        if t == j.at:
            j.arrive(Q, t)
    if not not Q and Q[0].et + Q[0].st <= t:
        Q.pop(0).depart(t)
        count += 1
    t += 1


avg_wt_time = 0
for el in jobs:
    avg_wt_time += el.wt
avg_wt_time = avg_wt_time/n

print("Average wait time is " + str(avg_wt_time))
```

Output:

```
Time for which you want to simulate
100
Number of Jobs in the system
10
Enter the max waiting time
5
Average wait time is 2.9
```

Assignment-1

```
import numpy as np

""" Question 1 """
A = np.matrix('1 3 4 2 ; 2 0 1 6; 4 1 2 7')
print(A)

# Matrix Size
aH = A.shape[0]
aW = A.shape[1]
print(aH)
print(aW)

# Matrix Transpose
Atr = A.transpose()
print(Atr)

print("\nxxxxxxxxxxxxxxxxxxxxxxxxx\n")


""" Question 2 """

B = np.matrix('2 2 3; 4 0 6; 8 1 5')
C = np.matrix('1 1 2; 6 3 5; 1 9 1')

# print(B)
# print(C)

D = np.subtract(B, C)
```

```
# print(D)
E = np.add(B, C)
# print(E)
F = np.add(E, 2)
# print(F)
G = B*C
# print(G)
H = np.multiply(B, C)
# print(H)

print("\nxxxxxxxxxxxxxxxxxxxxxxxx\n")


""" Question 3 """
A1 = np.matrix('2 7 6 8 9 10')
B1 = np.matrix('6 4 3 2 3 4')

C1 = np.multiply(A1, B1)
D1 = np.divide(A1, B1)

# print(C1)
# print(D1)

print("\nxxxxxxxxxxxxxxxxxxxxxxxx\n")


""" Question 4 """
r1 = np.matrix('7 3 5')
s1 = np.matrix(' 2 4 3')

q1 = np.power(r1, s1)
q2 = np.power(r1, 2)

# print(q1)
# print(q2)
```

```
print("\nxxxxxxxxxxxxxxxxxxxxxxxxx\n")

""" Question 5 """

A = np.matrix('1 3 4 2; 2 0 1 6; 4 1 2 7; 0 3 6 4')
print(np.diag(A))
print(A.sum(axis=0))
print(A.sum(axis=1))
print(A.sum())
A[1, :] += 2
A[:, 2] += 2
A[1, 2] -= 2  # Here value was added twice
print(A)
```

Assignment-2

```
import numpy as np

mat = np.matrix('3 11 6 5; 4 7 10 2; 13 9 0 8')
print(mat)
mat[2, 0] = 20
print(mat)
print(mat[1, 3]-mat[0, 1])
v = np.matrix('4 15 8 12 34 2 50 23 11')
u = v[0, 2:6]
print(u)
B = mat[:, 2]
print(B)
C = mat[2, :]
print(C)
F = mat[1:3, 1:3]
print(F)
```