# B190900014 Н. Даваацэрэн

# Оролтонд Пуассоны, гаралтанд илтгэгч тархалттай санамсаргүй хэмжигдэхүүн үүсгэн эдгээр утгуудын тусламжтайгаар симуляци хийж өмнө бодсон характеристикуудтай тохирч байгаа эсэхийг шалга.

import numpy as np

import queue

import copy

import matplotlib.pyplot as plt

# Оролтын параметрүүд

total\_time = *int*(input("Enter time for simulation (минут): "))

IAT\_rate = *int*(input("Enter Job Arrival Rate (/минут): "))

ST\_rate = *int*(input("Enter Job Service Rate (/минут): "))

rho = IAT\_rate/ST\_rate

# Initialize Parameters

qu = queue.Queue()

curr\_process = 0

IAT = []

ST = []

AT = []

wait\_time = []

server\_busy = False

list\_wait = []

list\_delay = []

num\_processes = *int*(np.random.poisson(IAT\_rate) \* total\_time)

num\_processes\_served = 0

# Inter-Arrival-Times (IAT)

for i in range(num\_processes):

    temp = np.random.exponential(1/IAT\_rate)\*60

    IAT.append(*int*(temp))

# Service-Times (ST) (where ST[i]!=0)

while not len(ST) == num\_processes:

    temp = np.random.exponential(1/ST\_rate)\*60

    if not *int*(temp) < 1:

        ST.append(*int*(temp))

ST\_copy = copy.deepcopy(ST)

for i in range(num\_processes):

    if i == 0:

        AT.append(0)

    else:

        AT.append(AT[i-1] + IAT[i])

    wait\_time.append(0)

# Simulation of M/M/1 Queue

for i in range(total\_time\*60):

    if server\_busy:

        for item in *list*(qu.queue):

            wait\_time[item] = wait\_time[item] + 1

        ST[curr\_process] = ST[curr\_process] - 1

        if ST[curr\_process] == 0:

            server\_busy = False

            num\_processes\_served = num\_processes\_served + 1

    for j in range(num\_processes):

        if i== AT[j]:

            qu.put(j)

    if not server\_busy and not qu.empty():

        curr\_process = qu.get()

        server\_busy = True

    sum\_wait = 0

    sum\_delay = 0

    for i in range(num\_processes\_served):

        sum\_wait = sum\_wait + wait\_time[i]

        sum\_delay = sum\_delay + wait\_time[i] + ST\_copy[i]

    if num\_processes\_served == 0:

        list\_wait.append(0)

        list\_delay.append(0)

    else:

        list\_wait.append(sum\_wait/(num\_processes\_served\*60))

        list\_delay.append(sum\_delay/(num\_processes\_served\*60))

plt.plot([i+1 for i in range(total\_time\*60)], list\_wait)

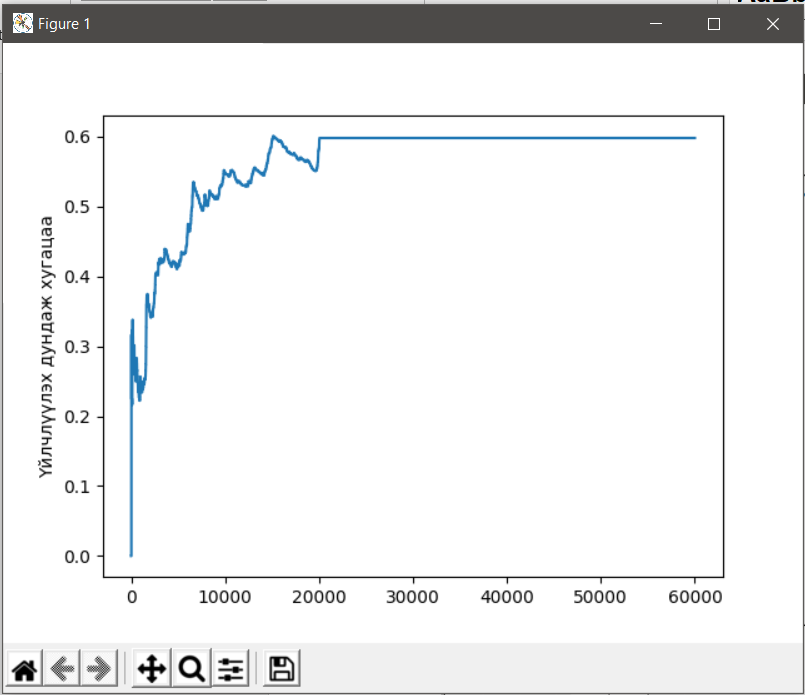
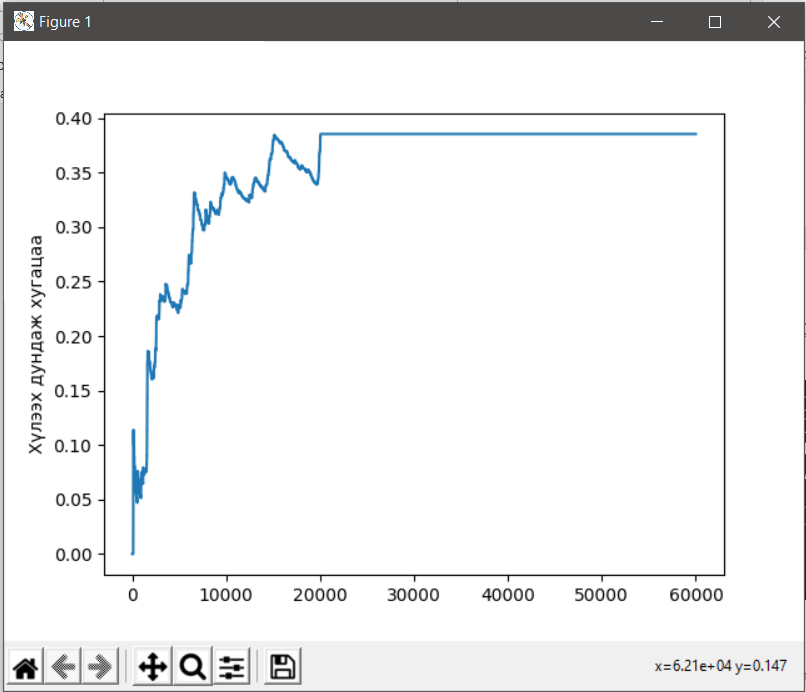
plt.ylabel("Хүлээх дундаж хугацаа")

plt.show()

plt.plot([i+1 for i in range(total\_time\*60)], list\_delay)

plt.ylabel("Үйлчлүүлэх дундаж хугацаа")

plt.show()



# A: 2a зайтай параллель шулуунууд татагдсан хавтгай руу 2l (l<a) урттай зүү хаяхад шулууныг огтолж унах магадлалыг ол.

import random

import math

import matplotlib.pyplot as plt

t=[]

y=[]

def buffon(*n*,*r*,*l*,*a*):

    data=[]

    for k in range(r):

        total = 0

        n=n+500\*k

        for i in range(n):

            x = random.uniform(0,a)

            theta = random.uniform(0,math.pi)

            x = x - l\*math.sin(theta)

            if x < 0 :

                total += 1 #Огтолж байгаа тоо

        c = 2.0\*l\*n

        d = a\*total

        print (*str*(k)+' '+*str*(c/d))

        t.append(k)

        y.append(c/d)

r=50

n=4000

l = 2 #зүүний урт

a= 2 #шулууны хоорондын зай

print ("Туршилтын тоо pi тооны үнэлгээ")

buffon(n,r,l,a)

plt.plot(t, y)

plt.xlabel('x - axis')

plt.ylabel('y - axis')

plt.title('Pi тооны үнэлгээ')

plt.show()

