```
#########################
##Monte Carlo Algorithm###
#########################
#Week 4 Statistical Methods in Experimental Physics
#author Qiyu Chen
#date 20240314 PI day

import random
import math
import matplotlib.pyplot as plt
tot=100000

#############################################################################
#exer3.3 started
'''
This is exer 3.3 using transform method
'''
random_numbers=[math.sqrt(random.random()*1) for _ in range(tot)]

plt.hist(random_numbers, bins=100, density=True)
plt.title('exer3.3  f(x)=2x/xm^2  x_max=1  transform')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.savefig('Desktop/MCP/exer3.3_transform.jpg')
plt.cla()

'''
This is exer 3.3 using Acceptance-Rejection Method(ARM)
'''
'''First generate x randomly note here we take x_max = 1'''

def f(x):
    return 2*x

x_min=0
x_max=1
random_x=[x_min+(x_max-x_min)*random.random() for _ in range(tot)]

'''Second generate u randomly'''

f_min=0
f_max=2
random_u=[f_min+(f_max-f_min)*random.random() for _ in range(tot)]

correct_x=[]
for i, j in zip(random_x, random_u):
    if j < f(i):
        correct_x.append(i)

'''Finally plot correct_x'''

plt.hist(correct_x, bins=100, density=True)
plt.title('exer3.3  f(x)=2x/xm^2  x_max=1  ARM')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.savefig('Desktop/MCP/exer3.3_ARM.jpg')
plt.cla()
#exer3.3 finished
#############################################################################


#############################################################################
#exer3.4 started
for n in range(1,21):
    all_z=[(sum([random.random() for _ in range(n)])-1/2*n)/math.sqrt(n/12) for _ in range(tot)]
    plt.hist(all_z, bins=100, density=True)
    plt.title('exer3.4 n='+str(n)+' distribution of z')
    plt.xlabel('Value')
    plt.ylabel('Frequency')
    plt.savefig('Desktop/MCP/exer3.4_n='+str(n)+'.jpg')
    plt.cla()
#exer3.4 finished
#############################################################################
```

```
##########################################################################
#exer3.5 started
import numpy
'''expotienal distribution'''
all_t=[numpy.random.exponential(scale=1) for _ in range(tot)]

'''gaussian distribution'''
all_x=[numpy.random.normal(loc=0, scale=0.5) for _ in range(tot)]

all_y=[i+j for i, j in zip(all_t, all_x)]
plt.hist(all_y, bins=100, density=True)
plt.title('exer3.5 distribution of y')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.savefig('Desktop/MCP/exer3.5.jpg')
plt.cla()

'''sigma>>tau'''
'''expotienal distribution'''
all_t=[numpy.random.exponential(scale=1) for _ in range(tot)]

'''gaussian distribution'''
all_x=[numpy.random.normal(loc=0, scale=50) for _ in range(tot)]

all_y=[i+j for i, j in zip(all_t, all_x)]
plt.hist(all_y, bins=100, density=True)
plt.title('exer3.5 distribution of y  sigma >> tau')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.savefig('Desktop/MCP/exer3.5_sigma_larger_than_tau.jpg')
plt.cla()

'''sigma<<tau'''
'''expotienal distribution'''
all_t=[numpy.random.exponential(scale=1) for _ in range(tot)]

'''gaussian distribution'''
all_x=[numpy.random.normal(loc=0, scale=0.005) for _ in range(tot)]

all_y=[i+j for i, j in zip(all_t, all_x)]
plt.hist(all_y, bins=100, density=True)
plt.title('exer3.5 distribution of y  sigma << tau')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.savefig('Desktop/MCP/exer3.5_sigma_smaller_than_tau.jpg')
plt.cla()
#exer3.5 finished
##########################################################################


##########################################################################
#exer3.6 started
'''3.6(b)'''
all_x=[math.tan(math.pi*(random.random()-1/2)) for _ in range(tot)]
plt.hist(all_x, bins=100, density=True, range=(-10,10))
plt.title('exer3.6(b) distribution of x')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.savefig('Desktop/MCP/exer3.6(b).jpg')
plt.cla()

'''3.6(c)'''
n=10
average_x=[sum([math.tan(math.pi*(random.random()-1/2)) for _ in range(n)])/n for _ in range(tot)]
plt.hist(average_x, bins=100, density=True, range=(-10,10))
plt.title('exer3.6(c) distribution of xbar')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.savefig('Desktop/MCP/exer3.6(c).jpg')
plt.cla()

#exer3.6 finished
##########################################################################
```

```
##########################################################################
#exer3.7 started
'''exer3.7(a)'''
N=6
nu=3.0
n_out=[]
tot=10000
for _ in range(tot):
    n_0=1
    for _ in range(N):
        n=0
        for _ in range(0, n_0):
            n+=numpy.random.poisson(nu)
        n_0=n
    n_out.append(n_0)

plt.hist(n_out, bins=50, density=True, range=(0,5000))
plt.title('exer3.7(a) distribution of n_out')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.savefig('Desktop/MCP/exer3.7(a).jpg')
plt.cla()
average=sum(n_out)/tot
n_out_minus_aver_square=[(n-average)**2 for n in n_out]
s_out=sum(n_out_minus_aver_square)/(tot-1)
print(average, s_out)

'''exer3.7(b)'''
n_out=[]
for _ in range(tot):
    n_0=1
    for i in range(N):
        n=0
        if i==0:
            for _ in range(0,n_0):
                n+=numpy.random.poisson(6.0)
        else:
            for _ in range(0, n_0):
                n+=numpy.random.poisson(nu)
        n_0=n
    n_out.append(n_0)

plt.hist(n_out, bins=50, density=True, range=(0,5000))
plt.title('exer3.7(b) distribution of n_out')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.savefig('Desktop/MCP/exer3.7(b).jpg')
plt.cla()
average=sum(n_out)/tot
n_out_minus_aver_square=[(n-average)**2 for n in n_out]
s_out=sum(n_out_minus_aver_square)/(tot-1)
print(average, s_out)

'''exer3.7(c)'''
nu=3.0
N=12
'''consider first six dynodes'''
n_out=[]
for _ in range(tot):
    n_0=1
    for i in range(6):
        n=0
        if i==0:
            for _ in range(0,n_0):
                n+=numpy.random.poisson(6.0)
        else:
            for _ in range(0, n_0):
                n+=numpy.random.poisson(nu)
        n_0=n
    n_out.append(n_0)
'''
Now we have a distribution, and we want to generate random numbers to fit this distribution
So, we use ARM
'''
'''first we want to fit a distribution of 6 dynodes'''
```

```
n_out_1=[]
for _ in range(tot):
    n_0=1
    for _ in range(6):
        n=0
        for _ in range(0, n_0):
            n+=numpy.random.poisson(nu)
        n_0=n
    n_out_1.append(n_0)

'''histogram is the ditribution function that we search for'''

'''Monte Carlo simulation'''
def random_generate(s):
    l=len(s)
    p=random.randint(0,l-1)
    return s[p]

n_out_final=[]
for i in range(tot):
    n_transient=0
    for j in range(0,n_out[i]):
        n_transient+=random_generate(n_out_1)
    n_out_final.append(n_transient)

'''Plot it'''
plt.hist(n_out_final, bins=50, density=True)
plt.title('exer3.7(c) distribution of n_out')
plt.xlabel('Value')
plt.ylabel('Frequency')
plt.savefig('Desktop/MCP/exer3.7(c).jpg')
plt.cla()
print(sum(n_out_final)/len(n_out_final))
#exer3.7 finished
################################################################################

#######
##end##
#######
```