Holly Do
COMP 680

Problem Set 06

Some problems in this section ask for a “procedure” or a “method” to produce a value (or estimate of a value). You may use Python for those problems. The scipy.stats distribution objects, as well as other modules, may be useful in that regard.

1. [16 pts] Let U be a Standard Uniform random variable. Show all the steps required to generate exponential variable with parameter λ =0.05

1.1. [4 pts] an Exponential random variable with the parameter λ = .0.05

Because Exponential distribution is continuous so we use Inverse Transform Method.

F(x) = 1- $e^{-\lambda x}$

So, $F(x)^{-1} = -\frac{\ln(1-x)}{\lambda}$

1/ We generate u value from Uniform U(0,1) using python

2/ Then we plug this u seed value into the inverse of cdf to get back a value which is x, a sample generated from the exponential distribution.

```
u = st.uniform().rvs()
lam = 0.05
-mth.log(1 - u)/lam
```

Example: x = 3.1766404208069603

1.2. [4 pts] a Bernoulli random variable with the probability of success p = 0.4

Bernoulli is a discrete distribution so we use Bucket Method.

1/ We generate u value from Uniform U(0,1) using python

```
u = st.uniform().rvs()
```

2/ We assign this value into buckets to distribute among the probabilities.

```
1 if u <= 0.4 else 0
```

Example:

u = 0.31310802762074286

x = 1

1.3. [8 pts] a Geometric random variable with parameter p =0.4

Geometric is a discrete distribution so we use Bucket Method.

1/ We generate u value from Uniform U(0,1) using python

2/ We assign this value into buckets to distribute among the probabilities.

```
import scipy.stats as st

def geometric():

    sum1 = 1
    while True:
        u = st.uniform().rvs()
        x = 1 if u <= 0.4 else 0
        if(x == 1):
            break
        else:
            sum1 = sum1+1
    return sum1

geometric()
```

Example: x = 6

2. [16 pts] Let x be a random variable with the density

$$f(x) = \frac{1}{12}x^{\frac{1}{3}}, 0 \leq x \leq 8$$

2.1. [10 pts] Show all the steps to generate samples from using the Inverse Transform Method

This function is a continuous function so we use Inverse Transform Method.

F(x) =1/16 * $x^{\frac{4}{3}}$ 0<=x<=8

So: $F(x)^{-1} = (16x)^{\frac{3}{4}}$

1/ We generate u value from Uniform U(0,1) using python

```
u = st.uniform().rvs()
```

2/ Then we plug this u seed value into the inverse of cdf to get back a value which is x, a sample generated from the exponential distribution.

```
import scipy.stats as st
import math as mth
u = st.uniform().rvs()
print(u)
print (mth.pow(16 * u, 3/4))
```

Example:

u = 0.1759202993053799

x = 2.206860437802502

2.2. [6 pts] Suppose your Standard Uniform generator yields random sample

. What value from does that correspond to?

```
u = 0.3455
print (mth.pow(16 * u, 3/4))
```

x = 3.6051723018916464

3. [16 pts] Suppose a random variable has the following cdf function

$$F(x) = \frac{2}{\pi}\int_{-1}^{x}\frac{1}{1+t^2}dt \quad , -1 \le x \le 1$$

3.1. [10 pts] Describe the steps to generate samples from using the Rejection Method.

PDF = f(x) = $\frac{2}{\pi}$(1/1+x^2)

1. Find domain, range bounding box: -1<=x<=1, 0<=f(x) <= $\frac{2}{\pi}$

2. Generate from x from U(-1, 1) and y from U[0, $\frac{2}{\pi}$]

```
import numpy as np
u = 2*np.random.uniform() -1              #equal to np.random.uniform(-1, 1)
v = 2./mth.pi * np.random.uniform()       #equal to np.random.uniform(0, mth.pi/2
```

3. Compute f(x)

```
def p6_3_f(x):
    return (2/mth.pi) * (1/(1+x**2))
```

4. If, y > f(x), go back to 2, repeat 6. Otherwise, accept

```
import numpy as np
import math as mth

def p6_3_f(x):
    return (2/mth.pi) * (1/(1+x**2))
# p6_3_f(0)

def p6_3_sample():
    while(True):
        x, y = 2.*np.random.uniform() -1, 2./mth.pi * np.random.uniform()
        fx = p6_3_f(x)
        if y <= fx: return x

p6_3_sample()
```

-0.40273526321684505

3.2. [3 pts] Let u, v and be a pair of random samples taken

from a Standard Uniform distribution. Generate the potential from .
Generate the from . Would the point be accepted or rejected?

```
x, y = 2.*0.2388 -1, 2./mth.pi * 0.7533
fx = p6_3_f(x)
print(y, fx)
```

0.47956567452449905 = y 0.5001326829555027= fx

For y < f(x), the point is accepted.

3.3. [3 pts] Let u, v and be a pair of random samples taken

from a Standard Uniform distribution. Would the point be accepted or rejected?
U= 0.2063
V=0.8097

```
x, y = 2.*0.2063 -1, 2./mth.pi * 0.8097
fx = p6_3_f(x)
print(y, fx)
```

0.5154710296860306 = y   0.4733096110684434 = fx

For y > f(x), the point is rejected.

4. [16 pts] The Jamaispay insurance company has 1256 clients. The probability of

any client filing claims is Poisson, with rate .023/year (multiple claims in 1 year
are possible). Assuming a claim is filed, the value of the claim is uniformly
distributed between \$500.00 and \$5000.00.

4.1. [8 pts] Show the steps for a procedure to estimate the expected value of

(total) claims payments in any given year. You must write a short Python program to describe your procedure.

1 customer filling claim : Poisson lamda = 0.023
Value of claim is uniform 500 to 5000
Client number = 1256

1. Calculate first year expense for all the clients:

```python
import numpy as np
import scipy.stats as st
nclients = 1256
min1 = 500
max1 =5000
def year1():
    total_1_year_all = 0.0
    for _ in range(nclients):
        nclaims = int(st.poisson(mu=0.023).rvs())
        total_1_year_all += sum(st.uniform.rvs(min1, max1 - min1) for _ in
range(nclaims))
    return total_1_year_all
```

2. Repeat n trials and find the average expense for all the clients per any year:

```python
def any_year(ntrials):
    total = 0
    for _ in range(ntrials):
        total += year1()
    avg = total/ntrials
    return avg

n = 100
expected_payments = any_year(n)
print (expected_payments)
```

4.2. [8 pts] The Insurance Governing Board requires that Jamaispay keep a

reserve equal to the expected maximum claim payout over a 10-year period. Show the steps for a procedure to estimate the reserve amount for

Jamai spays. You must write a short Python to describe your procedure.

1/ Return max among 10 years samples

```
max_amount = 0
years = 10
def max(years):
    for i in range (years):
        u = year1()
        if (u>max_amount):
            max_amount = u
    return max_amount
```

2/ Return expected max by running trials and get the average

```
def avg_max(years):
    for i in range (years):
        total += max(years)
    avg_max = total/years
    return avg_max
```

5. [18 pts] The HiRoller casino has a new game called InARow.

The InARow has a pile of 100 tokens, numbered from 1 to 100.
The pile of tokens is shuffled (mixed). Each player at the table gets 5 tokens.
If a player gets sequence of 5 in a row (for example 91,92,93,94,95), that is a winning hand. More than 1 player can win. In the following problems, assume you have a procedure shuffle() that produces random permutations of an array of numbers from 1 to 100.

5.1. [4 pts] The casino would like to decide how to set the payoffs and entry

charge. Suppose the table has exactly 4 players.
Describe a procedure to estimate the expected number of winning hands on any round of InARow. You should use the shuffle procedure to generate deals for InARow. You must write a short Python program to describe your procedure.

1st person's deal

```
deck = np.arange(100)+1
```

```
def deal1():
    np.random.shuffle(deck)
    return deck[0:5]
```

1/ First round winning hands

```
deck = np.arange(100)+1

def winning_hands(players):
    np.random.shuffle(deck)
    wins = 0
    for i in range(0,5*players,5):
        hand = deck[i: i+5]
        hand.sort()
        if (hand[4]- hand[0] == 4):
            wins += 1
    return wins
```

```
0
```

2/Average the wins among 100000 trials to get average winning hands of any round

```
def avg_winning_hands(trials, players):
    total = 0
    for i in range (trials):
        total += winning_hands(players)
    avg_wins = total/trials
    return avg_wins

expected_wins = avg_winning_hands(100000, 4)
print (expected_wins)
```

```
1e-05
```

5.2. [6 pts] Suppose a table has any number of players from 4 to 8.

The number of players is uniformly distributed among the possibilities. Furthermore, players may come and go after any deal.
Using Standard Uniform Procedure S, and the shuffle procedure as the

source of randomness, show the steps of a method to estimate the average number of wins in this situation
You must write a short Python program to describe your method.

EACH PERSON DOES NOT HAVE FRESH DECK OF CARD
ONE SHUFFLE THEN TAKE 20 FIRST CARDS DIVIDE TO 4 PLAYERS

```
import scipy.stats as st
import random

deck = np.arange(100)+1

#1st round
def winning_hands_2():
    np.random.shuffle(deck)
    wins = 0
    players = random.randint(4,8)
    for i in range(0,5*players,5):
        hand = deck[i: i+5]
        hand.sort()
        if (hand[4]- hand[0] == 4):
            wins += 1
    return wins

hands = 100000

def avg_winning_hands_2(hands):
    total = 0
    for i in range(hands):
        total += winning_hands_2()
    avg_wins_2 = total/hands

    return avg_wins_2

expected_wins = avg_winning_hands_2(100000)
print (expected_wins)
```

```
2e-05
```

5.3.[4 pts] Using the Chebyshev method, decide on a minimum number of simulation runs to produce a .001 accuracy with probability .95 for problem 5.2

Chebyshev's inequality formula with e = 0.001 and confidence 0.95 so a = 1 – 0.95 = 0.05

N≥ $\frac{1}{4a\epsilon^2}$

¼ IS P TIMES Q MAX BECAUSE THIS SIMULATION INCLUDES BERNOULLI TRIALS

$\frac{1}{4a\epsilon^2}$= 1 / (4 * 0.05 * (0.001**2))

N≥ 5000000

5.4. [4 pts] Using the Central Limit Theorem method, estimate the number of simulation runs to produce a .001 accuracy with probability .95 for problem 5.2

Chebyshev's inequality formula with e = 0.001 and a = 1 – 0.95 = 0.05

$$N \geq \frac{1}{4}\left(\frac{\Phi^{-1}\left(\frac{a}{2}\right)}{\epsilon}\right)^2$$

CDF of standard normal distribution

$$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{t^2}{2}}\,dt$$

There is no simple closed-form expression for inverse CDF of the standard normal distribution compute using Python:

```
from scipy.stats import norm

quantile = norm.ppf(0.05/2)
n = 0.25*((quantile/0.001)**2)
```

```
print (n)
```

960364.7051735318

N≥ 960365

6. [18 pts] Use Monte Carlo integration to estimate the following integrals.

[You will not like doing this by hand. If you do choose to write Python
programs, please include them as part of showing your work

LOOK AT A AND B TO DETERMINE WHAT P(X) TO USE

1/ Define general functions to be used to estimate integral using Monte Carlo method:

```
nmax = 1000

def mc_uinform(g,a=0,b=1,nsims=nmax):
    mean1 = (np.sum(g(st.uniform(a,b-a).rvs(nsims)))) / nsims
    return (b-a)*(mean1)

def mc_norm(g,nsims=nmax):
    dist = st.norm()
    x = dist.rvs(nsims)
    px = dist.pdf(x)
    return np.sum(g(x) / px ) / nsims

def mc_exp(g,nsims=nmax):
    dist = st.expon()
    x = dist.rvs(nsims)
    px = dist.pdf(x)
    return np.sum(g(x) / px ) / nsims
```

6.1

$$\int_0^1 \left| \sin\left(\frac{1}{x}\right) \right| dx$$

This is bounded [0, 1] so we use uniform distribution to estimate the integral.

```python
def f6p1(x):
    return np.absolute(np.sin(1./x))

y = mc_uinform(f6p1,a=0,b=1,nsims=nmax)

print(y)
```

```
0.7795657700464369
```

6.2

$$\int_0^5 \left| \sin\left(\frac{1}{x}\right) \right| dx$$

This is bounded [0, 5] so we use uniform distribution to estimate the integral.

```python
def f6p2(x):
    return np.absolute(np.sin(1./x))

y = mc_uinform(f6p2,a=0,b=5,nsims=nmax)

print(y)
```

```
2.322515658933906
```

6.3

$$\int_0^1 \sin\left(\frac{1}{x}\right) dx$$

This is bounded [0, 1] so we use uniform distribution to estimate the integral.

```python
def f6p3(x):
    return np.sin(1/x)

y = mc_uinform(f6p3,a=0,b=1,nsims=nmax)
```

```
print(y)
```

0.5169755280045899

6.4

$$\int_{-2}^{2} e^{-x^2}\ dx$$

This is bounded [-2, 2] so we use uniform distribution to estimate the integral.

```
def f6p4(x):
    return np.exp(-(x**2))

y = mc_uinform(f6p4,a=-2,b=2,nsims=nmax)

print(y)
```

1.7678179790519835

6.5

$$\int_{-\infty}^{\infty} e^{-x^2}\ dx$$

This is bounded $[-\infty\ ,\ \infty\ ]$ so, we use normal distribution to estimate the integral.

```
def f6p5(x):
    return np.exp(-x**2)

y = mc_norm(f6p5,nsims=nmax)

print(y)
```

1.786339769931065

6.6

$$\int_0^\infty e^{-\sqrt{x}}\ dx$$

This is bounded [0, ∞] so, we use exponential distribution for x

```
def f6p6(x):
    return np.exp(-np.sqrt(x))

y = mc_exp(f6p6,nsims=nmax)

print(y)
```

```
1.404408247316111
```