CALIFORNIA STATE UNIVERSITY LONG BEACH

EE 381 – Probability and Statistic Computing

Project 03

UNIFORM DISTRIBUTION

Professor: *Duc Tran*

Team members: *Alyssa Faiferlick – Thanh Nguyen*

Class: *EE 381 – Section 13*

Due date: *10/29/2020*

I. QUESTION 1 – CDF AND PDF OF UNIFORM DISTRIBUTION

1. INTRODUCTION:

Write a python code to compute probability density function (pdf) and cumulative distribution function (cdf) using equation:

$$f(x;a,b) = \frac{1}{b-a}; a < x < b$$

And

$$F(x) = \begin{cases} 0 & x \le a \\ \frac{x-a}{b-a} & a < x < b \\ 1 & x \ge b \end{cases}$$

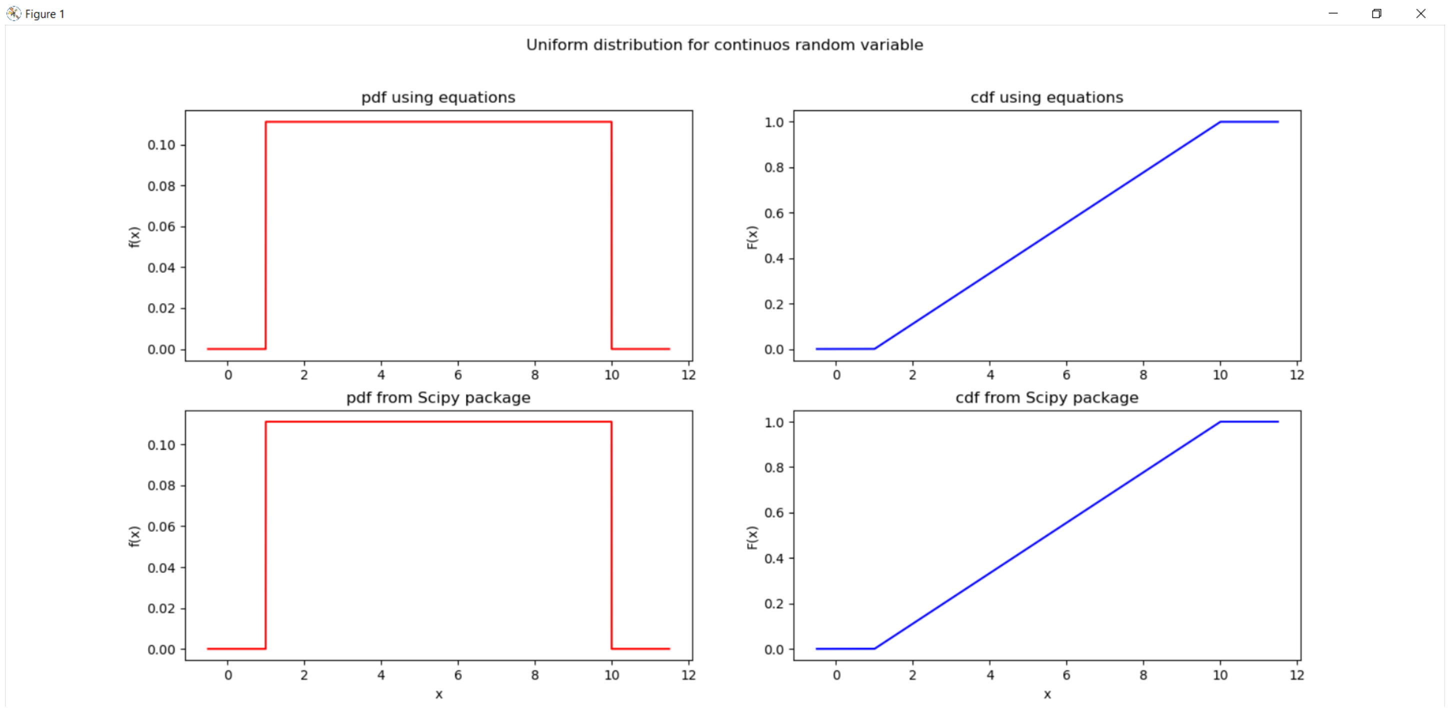
Then using stats.uniform.pdf and stats.uniform.cdf functions from Scipy package in Python to check the results.

2. PROCEDURE:

- Write a function to calculate f(x):
  - $If\ 1 < x < 10, then\ return \frac{1}{9}$
  - $If\ x \le 1\ or\ x \ge 10, then\ return\ 0$
- Write a function to calculate F(x):
  - $If\ 1 < x < 10, then\ return \frac{x-1}{9}$
  - $If\ x \le 1, then\ return\ 0$
  - $If\ x \ge 10, then\ return\ 1$
- Generate a list from -0.5 to 11.5 with the number of elements in the list is 100000 using numpy.linspace().
- Create a frame and divide it into 4 parts to plot pictures.
- Part a: For each element in the list, calculate f(x) and F(x) using these functions above, then plot (x, f(x)) and (x, F(x)).
- Part b: For each element in the list, calculate f(x) and F(x) by using uniform.pdf(x, 1, 9) and uniform.cdf(x, 1, 9), then plot (x, f(x)) and (x, F(x)).

3. RESULTS AND DISCUSSION:

- Results:



- The results are what us expected them to be. The graphs which are calculated by using equations look similar to the graphs which use pdf and cdf functions from Scipy package.

4. CONCLUSION:

- For this question, we used the uniform distribution for continuous random variable, the probability density equation and cumulative distribution equation from the lecture to calculate exactly f(x) and F(x).
- We read a lot of resources to find out how stats.uniform() method works, and learnt to use subplots to plot multiple graphs in one frame.

5. APPENDIX:

```
import matplotlib.pyplot as plt
from scipy.stats import uniform
import numpy as np

# Create cdf of a uniform random variable
def uniform_cdf(a, b, x):
    if (x > a and x < b):
        return (x-a)/(b-a)
    elif (x >= b):
           return 1
       else:
           return 0

   # Create pdf of a uniform random variable
   def uniform_pdf(a, b, x):
       if (x > a and x < b):
           return 1 / (b - a)
       else:
           return 0

   # Genrating uniform distribution
   a, b = 1, 10
   size = 100000 #the number of points
   x = np.linspace(-0.5, 11.5, size)

   # Plotting
   # divide the frame into 4 part: part a is in row 1, and part b is in row 2
   fig, axes = plt.subplots(2, 2)
   # Set title and label for graphs
   plt.suptitle("Uniform distribution for continuos random variable")
   axes[0][0].set_title("pdf using equations")
   axes[1][0].set_title("pdf from Scipy package")
   axes[0][1].set_title("cdf using equations")
   axes[1][1].set_title("cdf from Scipy package")
   for i in range(2):
       axes[1][i].set_xlabel('x')
       axes[i][0].set_ylabel('f(x)')
       axes[i][1].set_ylabel('F(x)')

   # Plotting pdf
   axes[0][0].plot(x, [uniform_pdf(a, b, x_i) for x_i in x], 'r-')
   axes[1][0].plot(x, uniform.pdf(x, a, b - 1), 'r-')

   # Plotting cdf
   axes[0][1].plot(x, [uniform_cdf(a, b, x_i) for x_i in x], 'b-')
   axes[1][1].plot(x, uniform.cdf(x, a, b - 1), 'b-')

   plt.show()
```

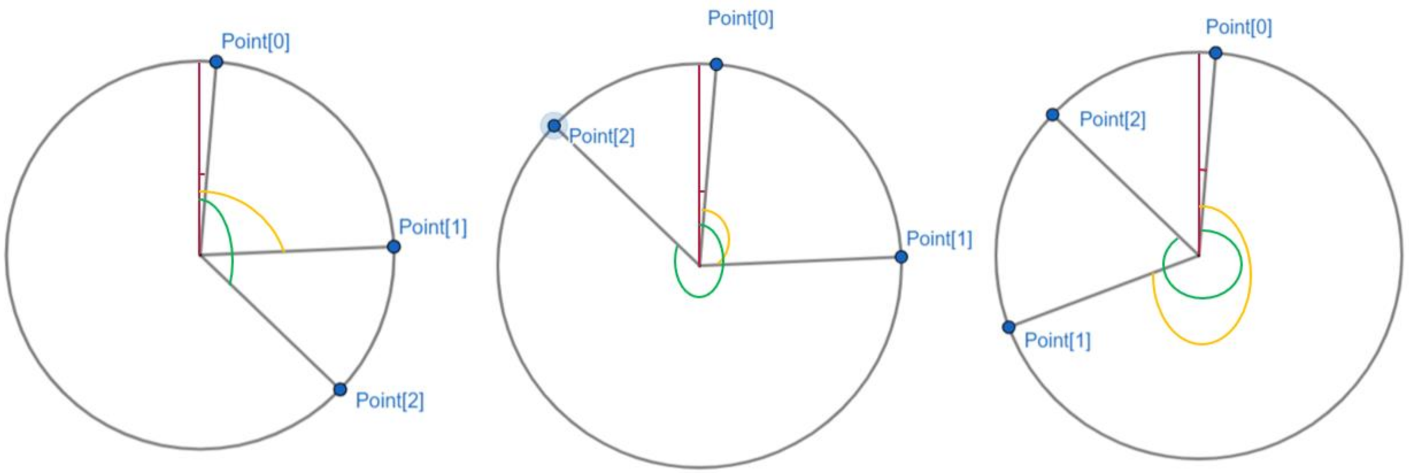
II. QUESTION 2 – SEMICIRCLE

1. INTRODUCTION:

Write Python code to find the probability that when three random points are selected from the circumference of a circle, all lie on the same semicircle.

2. PROCEDURE:

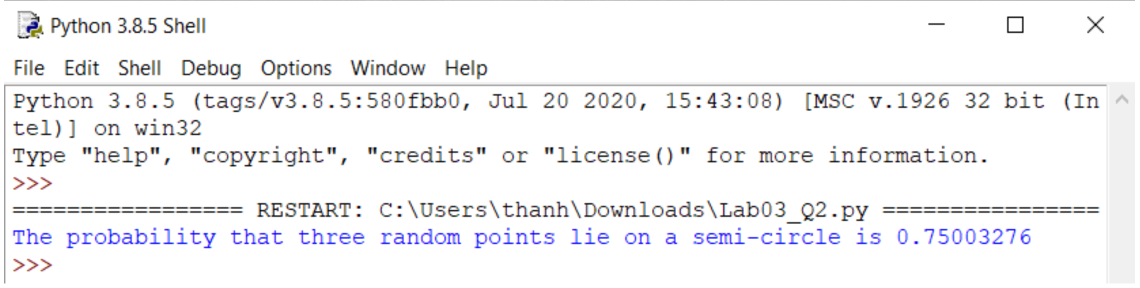
- There are three cases of 3 points:



- Generate three random points from [0, 360]
- Sort the three points from smallest to largest
- Using the three cases of how the three points could lie in a semicircle. We calculated the sum of angle of these points to see if the sum is less than are equal 180 degrees then 3 points lie on one semicircle. Notice that if the angel difference between two points is greater than 180 than we get 360 subtract the difference be the angel between these points).

3. RESULTS AND DISCUSSION:

- Results:



```
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:43:08) [MSC v.1926 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================= RESTART: C:\Users\thanh\Downloads\Lab03_Q2.py ================
The probability that three random points lie on a semi-circle is 0.75003276
>>>
```

- The probability that three random points lie on the same semicircle is ¾. Our results is 0.75 which is identical to ¾, which was what we expected.

4. CONCLUSION:

While this problem was very simple to solve on paper, I struggled to translate it into Python code. The first code I created would not compute all trials and the result was not correct. But after fixing the last trial it gave us the correct output.

5. APPENDIX:

```
"""
    Class: EE 381 - Section 13
    Project 03 - Uniform Distribution _Part 02
"""

import numpy as np
import random
from scipy.stats import uniform

  """
      Three points #1, #2, and #3 are selected at random from
      the circumference of a circle. Write Python code to
      find the probability that the three points lie on the
      same semicircle.
  """


  def findThreePoints():

      threePoints = np.random.uniform(0, 360, size = 3)
      threePoints.sort()

      return threePoints

  def semiCircleTest(N):

```

```python
    count = 0

    for i in range(0, N):

        threePoints = findThreePoints()
        temp1 = threePoints[2] - threePoints[0]
        if (temp1 <= 180):
            count += 1
        else:
            temp2 = threePoints[1] - threePoints[0]
            if (temp2 < 180):
                if (360 - temp1 + temp2 < 180):
                    count += 1
            else:
                if (360 - temp2 <= 180):
                    count += 1

    print("The probability that three random points lie on a semi-circle is", count / N)


semiCircleTest(1000000)
```