Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації та управління

Організація обчислювальних процесів

Комп’ютерний практикум №1

«Перевірка генератора випадкових чисел на відповість закону розподілу»

Виконав:

студент групи ІС-73

Коноплянка Д. С.

Перевірила:

Новікова П.А.

Київ 2020 р.

**Завдання до практичної роботи**

* Згенерувати 10000 випадкових чисел трьома вказаними нижче способами. **45 балів**.
* Згенерувати випадкове число за формулою , де  - випадкове число, рівномірно розподілене в інтервалі (0;1). Числа можна створювати за допомогою вбудованого в мову програмування генератора випадкових чисел. Перевірити на відповідність експоненційному закону розподілу . Перевірку зробити при різних значеннях λ.
* Згенерувати випадкове число по формулах:

,

де ξi  - випадкове число, рівномірно розподілене в інтервалі (0;1). Числа ξi можна створювати за допомогою убудованого в мову програмування генератора випадкових чисел. Перевірити на відповідність нормальному закону розподілу:

.

Перевірку зробити при різних значеннях µ і σ.

* Згенерувати випадкове число за формулою

,

=513, *с*=231.

Перевірити на відповідність рівномірному закону розподілу в інтервалі (0;1). Перевірку зробити при різних значеннях параметрів *а* і *с*.

* Для кожного побудованого генератора випадкових чисел побудувати гістограму частот, знайти середнє і дисперсію цих випадкових чисел. По виду гістограми частот визначити вид закону розподілу. **20 балів.**
* Відповідність заданому закону розподілу перевірити за допомогою критерію згоди χ2. **30 балів**
* Зробити висновки щодо запропонованих способів генерування випадкових величин. **5 балів**

**Лістинг для експоненційного розподілу**

import random

import numpy as np

import matplotlib.pyplot as plt

from plotHelper import scatterplot

randNumbers = []

randNumbersY = []

lamb = 0.01

x2Nab = 0

X2Krit = 37.57

for that in range(10000):

    randNumbers.append(random.uniform(0.001,1))

i = 0

while i < len(randNumbers):

    randNumbers[i] = (-1 / lamb) \* np.log(randNumbers[i])

    i+=1

k = 0

while k < len(randNumbers):

    randNumbersY.append(1 - np.exp(-1 \* lamb \* randNumbers[k]))

    k+=1

scatterplot(randNumbers, randNumbersY, "MyNumbers", "FXNumbers", "Plot")

n, x, bars = plt.hist(randNumbers, bins = 20)

result = zip(x, n)

plt.show()

xn = []

for item in result:

    xn.append(item[0] \* item[1])

mean = sum(xn) / sum(n)

print("Avarage: ", mean)

i = 0

dispSum = 0

while i < len(xn):

    dispSum += pow((mean - x[i]), 2) \* n[i]

    i += 1

disp = dispSum / (sum(n))

print("Dispersion: ", disp)

p = []

i = 0

while i < len(x) - 1:

    p.append(np.exp(- lamb \* x[i]) - np.exp(- lamb \* x[i+1]))

    i += 1

ni = []

i = 0

while i < len(p):

    ni.append(sum(n) \* p[i])

    i += 1

result = []

i = 0

while i < len(p):

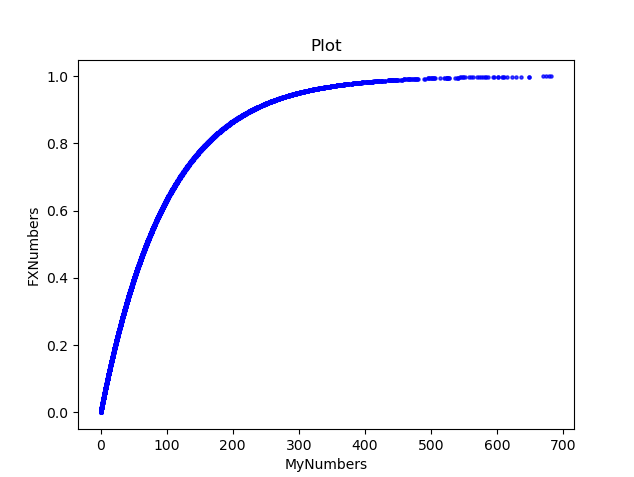
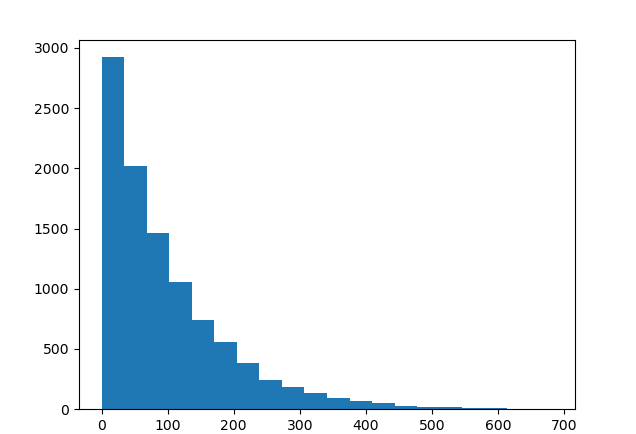
    result.append(pow(n[i] - ni[i], 2) / ni[i])

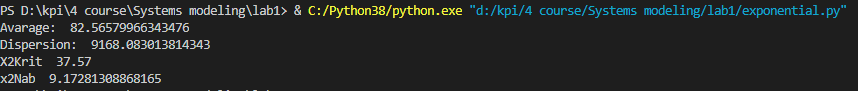
    i += 1

x2Nab = sum(result)

print("X2Krit ", X2Krit)

print("x2Nab ", x2Nab)



**Лістинг для нормального розподілу**

import random

import numpy as np

import matplotlib.pyplot as plt

from scipy import stats

from plotHelper import scatterplot

randNumbers = []

randNumbersY = []

sigma = 1

alpha = 0.01

x2Nab = 0

X2Krit = 37.57

for that in range(10000):

    miu = -6

    for i in range(12):

        miu+= random.uniform(0.001,1)

    randNumbers.append(sigma \* miu + alpha)

k = 0

while k < len(randNumbers):

    randNumbersY.append( (1 / (sigma \* np.sqrt(2 \* np.pi))) \* np.exp(- (pow(randNumbers[k] - alpha,2) / (2 \* pow(sigma, 2)))))

    k+=1

scatterplot(randNumbers, randNumbersY, "MyNumbers", "FXNumbers", "Plot")

n, x, bars = plt.hist(randNumbers, bins = 20)

plt.show()

print(x)

result = zip(x, n)

xn = []

for item in result:

    xn.append(item[0] \* item[1])

mean = sum(xn) / sum(n)

print("Avarage: ", mean)

i = 0

dispSum = 0

while i < len(xn):

    dispSum += pow((mean - x[i]), 2) \* n[i]

    i += 1

disp = dispSum / (sum(n) - 1)

print("Dispersion: ", dispSum)

otklonenie = np.sqrt(disp)

h = (max(x) - min(x)) / len(n)

print("H: ", h)

u = []

i = 0

while i < len(x):

    u.append((x[i]-mean) / otklonenie)

    i += 1

fu = []

for item in u:

    fu.append(np.exp(-pow(item, 2) / 2) / np.sqrt(2\*np.pi))

ni0 = []

for item in fu:

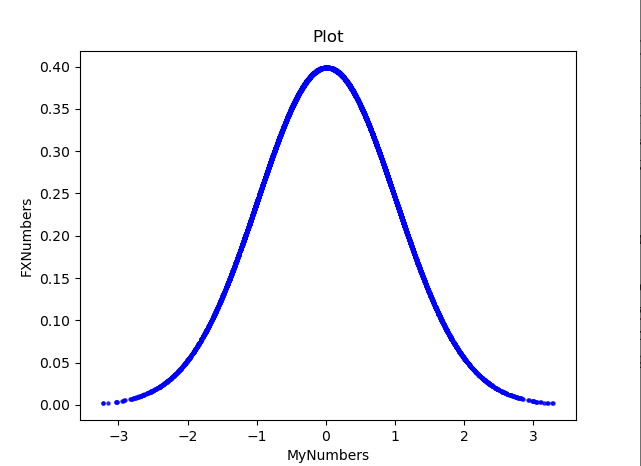
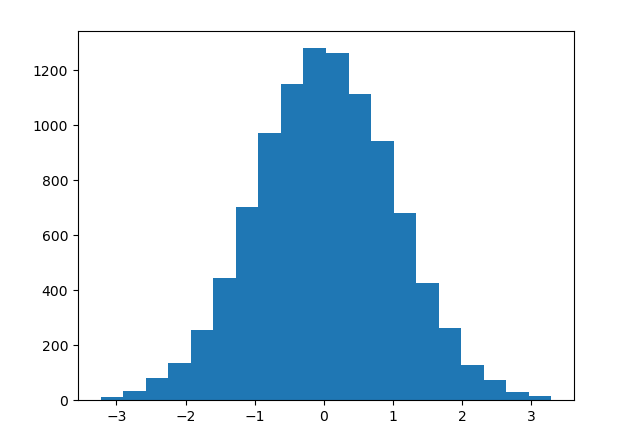
    ni0.append((sum(n) \* h \* item) / otklonenie)

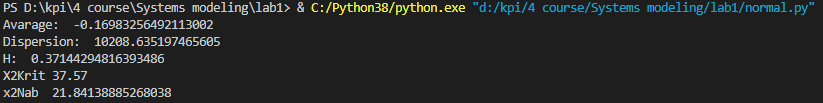
for i in range(len(n)):

    x2Nab += pow((n[i] - ni0[i]), 2) / ni0[i]

print("X2Krit", X2Krit)

print("x2Nab ", x2Nab)



**Лістинг для рівномірного розподілу**

import numpy as np

import matplotlib.pyplot as plt

from scipy.stats import kstest

from plotHelper import scatterplot

a = pow(5, 13)

c = pow(2, 31)

z = 2

x2Nab = 0

X2Krit = 37.57

randNumbers = []

for i in range(10000):

    z = (a \* z) %c

    randNumbers.append(z / c)

test = kstest(randNumbers, 'uniform')

print(test)

n, x, bars = plt.hist(randNumbers, bins = 20)

plt.show()

result = zip(x, n)

xn = []

for item in result:

    xn.append(item[0] \* item[1])

mean = sum(xn) / sum(n)

print("Avarage: ", mean)

xnT = []

i = 0

dispSum = 0

while i < len(xn):

    value = pow((x[i] - mean), 2) \* n[i]

    xnT.append(value)

    dispSum += value

    i += 1

disp = dispSum / (sum(n))

print("Dispersion: ", disp)

otklonenie = np.sqrt(disp)

aStar = mean - np.sqrt(3) \* otklonenie

bStar = mean + np.sqrt(3) \* otklonenie

randNumbersY = []

for z in randNumbers:

    randNumbersY.append(1 / (bStar - aStar))

scatterplot(randNumbers, randNumbersY)

plotnost = 1 / (bStar - aStar)

xdiff = np.diff(x)

h = np.mean(xdiff)

h = h / 2

nFirst = sum(n) \* plotnost \* (x[0] + h - aStar)

nLast = sum(n) \* plotnost \* (bStar - (x[len(x) - 1] - h))

ns = []

ns.append(nFirst)

i = 1

while i < len(x) - 1:

    ns.append(sum(n) \* plotnost \* (x[i+1] - x[i]))

    i += 1

ns.append(nLast)

i = 0

while i < len(n):

    x2Nab += pow(n[i] - ns[i], 2) / ns[i]

    i += 1

print("X2Krit", X2Krit)

print("x2Nab", x2Nab)

