

Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и информатики»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №0**  
По дисциплине «Моделирование»

Выполнил:  
Студент гр. ИВ-622  
Неудахин В. Д.

Проверила:  
Ассистент Кафедры ВС  
Петухова Я. В.

Новосибирск 2020

## Оглавление

Постановка задачи .....	3
Теоретические сведения .....	3
Ход работы .....	5
Заключение .....	9
Листинг .....	10

## Постановка задачи

Необходимо взять готовую реализацию трех генераторов псевдослучайных чисел на отрезке  $[0, 1]$  и убедиться в их равномерном распределении, используя такие параметры, как распределение Пирсона и автокорреляции.

## Теоретические сведения

Пусть  $X$  – исследуемая случайная величина. Требуется проверить гипотезу  $H_0$  о том, что данная случайная величина подчиняется закону распределения  $F(x)$ . Для этого необходимо произвести выборку из  $n$  независимых наблюдений над случайной величиной  $X$ :  $X^n = (x_1, \dots, x_n)$ ,  $x_i \in [a, b]$ ,  $\forall i = 1 \dots n$ , и по ней построить эмпирический закон распределения  $F'(x)$  случайной величины  $X$ .

Критерий согласия Пирсона – метод, который позволяет оценить значимость различий между фактическим (выявленным в результате исследования) количеством исходов или качественных характеристик выборки, попадающих в каждую категорию, и теоретическим количеством, которое можно ожидать в изучаемых группах при справедливости нулевой гипотезы.

Процедура проверки гипотез с использованием критериев типа  $\chi^2$  предусматривает группирование наблюдений. Область определения случайной величины разбивают на  $k$  непересекающихся интервалов граничными точками  $(x_0, x_1, x_2, \dots, x_k)$ , где  $x_0$  – нижняя грань области определения случайной величины;  $x_k$  – верхняя грань. В соответствии с заданным разбиением подсчитывают число  $n_i$  выборочных значений, попавших в  $i$ -й интервал, и вероятности попадания в интервал, соответствующие теоретическому закону с функцией распределения. Рассчитывается «хи-квадрат» по формуле:

$$\chi^2 = \frac{(n_1 - p_1 N)^2}{p_1 N} + \frac{(n_2 - p_2 N)^2}{p_2 N} + \dots + \frac{(n_k - p_k N)^2}{p_k N}$$

, где

- $p_i$  – теоретическая вероятность попадания чисел в  $i$ -ый интервал
- $k$  – количество интервалов
- $N$  – общее количество сгенерированных чисел
- $n_i$  – количество попавших чисел в интервал

- $\chi^2$  – критерий, который позволяет определить, удовлетворяет ли генератор случайных чисел требованиям равномерного распределения или нет

Если  $\chi_{\text{экс}}^2 \leq \chi_{\text{таб}}^2$ , то гипотеза не противоречит опытным данным, иначе отвергается.

Условие применения критерия Пирсона является наличие в каждом интервале не менее пяти наблюдений. И следует помнить, что для равномерного распределения все значения вероятности одинаковы. Далее надо провести корреляционный анализ. Корреляционный анализ – популярный метод статистического исследования, который используется для выявления степени зависимости одного показателя от другого.

Автокорреляция – статистическая взаимосвязь между последовательностями величин одного ряда, взятыми со сдвигом, например, для случайного процесса – со сдвигом по времени.

$$a(\tau) = \frac{\sum_{i=1}^{N-\tau} (x_i - Ex) * (x_{i+\tau} - Ex)}{(N - \tau) * S^2(x)}, \text{ где}$$

$E(x)$  – математическое ожидание – среднее значение случайной величины при стремлении количества выборок или количества ее измерений:

$$E(x) = \frac{1}{N} \sum_{i=1}^N X_i$$

$S^2(x)$  – выборочная дисперсия случайной величины – это оценка теоретической дисперсии распределения, рассчитанная на основе данных выборки:

$$S^2(x) = \frac{1}{N} \sum_{i=1}^N (x_i - x)^2 = \frac{1}{N} \sum_{i=1}^N x_i^2 - (x)^2$$

$x_{i+\tau}$  – множество значений другой случайной величины (полученной из значений прошлой случайной величины, но с некоторым смещением)  $n$  – мощность множества случайных величин  $\tau$  – смещение последовательности.

## Ход работы

Для выполнения данной лабораторной работы было выбрано три генератора случайных чисел, независящих друг от друга:

- В качестве первого генератора был взят генератор случайных чисел из стандартной библиотеки `random()` языка программирования Python.
- Вторым генератором случайных чисел был генератор `rand()` из библиотеки модуля NumPy для Python.
- А третьим генератором стал `SystemRandom()`.

Для наглядности работы данных ГСЧ каждым из них было сгенерировано 1.000.000 случайных чисел в диапазоне от 0 до 1. Так же этот диапазон был поделён на 1000 равномерных интервалов. И посчитан «хи квадрат»:

```
Генератор Random()
N: 1000000
K: 1000
Хи квадрат: 970.9199999999984
```

Рисунок 1 –`random()` из стандартной библиотеки

```
Генератор модуля NumPy
N: 1000000
K: 1000
Хи квадрат: 937.6499999999978
```

Рисунок 2 –`rand()` из модуля NumPy

```
Генератор SystemRandom()
N: 1000000
K: 1000
Хи квадрат: 952.4699999999992
```

Рисунок 3 – `SystemRandom()`

Так же была рассчитана автокорреляция для каждого ГСЧ при  $N = 1.000.000$  и  $k = 1.000$ , где  $\alpha(\tau)$  – автокорреляция,  $\tau$  – смещение интервалов.

```
a(tao) -0.000026
tao: 1
a(tao) 0.000141
tao: 2
a(tao) 0.000086
tao: 3
a(tao) -0.000074
tao: 4
a(tao) 0.000154
tao: 5
a(tao) 0.000010
tao: 6
a(tao) 0.000112
tao: 7
a(tao) 0.000034
tao: 8
a(tao) 0.000104
tao: 9
a(tao) 0.000122
tao: 10
a(tao) 0.000072
tao: 11
a(tao) -0.000053
tao: 12
a(tao) 0.000043
tao: 13
a(tao) -0.000045
tao: 14
a(tao) 0.000064
tao: 15
```

Рисунок 4 - Random()

```
a(tao) 0.000004
tao: 1
a(tao) -0.000004
tao: 2
a(tao) 0.000002
tao: 3
a(tao) 0.000014
tao: 4
a(tao) -0.000005
tao: 5
a(tao) -0.000014
tao: 6
a(tao) 0.000007
tao: 7
a(tao) 0.000004
tao: 8
a(tao) 0.000005
tao: 9
a(tao) -0.000008
tao: 10
a(tao) -0.000006
tao: 11
a(tao) 0.000008
tao: 12
a(tao) -0.000007
tao: 13
a(tao) 0.000004
tao: 14
a(tao) 0.000006
tao: 15
```

Рисунок 5 - Rand()

```
a(tao) 0.000016
tao: 1
a(tao) 0.000011
tao: 2
a(tao) -0.000008
tao: 3
a(tao) -0.000007
tao: 4
a(tao) 0.000000
tao: 5
a(tao) 0.000007
tao: 6
a(tao) -0.000015
tao: 7
a(tao) -0.000007
tao: 8
a(tao) -0.000011
tao: 9
a(tao) 0.000008
tao: 10
a(tao) -0.000001
tao: 11
a(tao) 0.000011
tao: 12
a(tao) -0.000005
tao: 13
a(tao) 0.000015
tao: 14
a(tao) 0.000002
tao: 15
```

Рисунок 6 – SystemRand()

Исходя из этих данных можно сделать вывод, что числа действительно являются случайными, так как значения  $\alpha(\tau)$  стремятся к нулю.

Далее было еще несколько попыток генерации случайных чисел каждым ГСЧ с изменением количества чисел и количества интервалов. Все результаты показаны в таблице 1.

Таблица 1 – показатели критерия «хи квадрат» при разных N и k

ГСЧ	N = 1.000.000 k = 1.000	N = 1.000.000 k = 100	N = 10.000 k = 1.000
Random()	$\chi^2 = 993.72$	$\chi^2 = 99.975$	$\chi^2 = 987.6$
NumPy Rand()	$\chi^2 = 953.382$	$\chi^2 = 95.783$	$\chi^2 = 979.399$
SystemRamdom()	$\chi^2 = 998.081$	$\chi^2 = 91.015$	$\chi^2 = 1004.799$
Табличное значение $\chi^2$	$\chi^2 = 1142.848$	$\chi^2 = 134,642$	$\chi^2 = 1142.848$



## Заключение

В данной лабораторной работе были найдены три независимых друг от друга генератора случайных чисел языка программирования Python и реализованы три программы для генерации псевдослучайных чисел, расчетов автокорреляции и критерия согласия Пирсона.

Из результатов опытов следует сказать, что критерии «хи квадрат» которые равны  $\chi^2 = 970.919$ ,  $\chi^2 = 937.649$  и  $\chi^2 = 952.469$  соответственно для каждого из них принимается гипотеза о равновероятном распределении в ГСЧ, исходя из табличных значений для каждого из случая. Так же было выявлено какую информацию дает Критерий согласия Пирсона, а именно:

1. Чем больше отклонение, тем менее равномерным становится распределение чисел на интервалах.
2. Так как значение, используемое в критерии согласия Пирсона, не превышало критического значения, следует, что гипотезу о равномерном распределении нельзя отвергнуть.

Основываясь на проведённых опытах видно, что функция автокорреляции стремится к нулю при каждом изменении  $\tau$  (смещение последовательности). Это говорит о слабой силе корреляции. Так же стоит заметить, что коэффициент автокорреляции напрямую зависит от количества точек на интервалах. К примеру, если уменьшить количество интервалов, то, следовательно, точек на меньшем количестве интервалов будет больше и коэффициент автокорреляции будет стремиться к нулю. А если увеличить количество интервалов, а точек уменьшить, то на каждый интервал будет попадать неравное количество точек и как следствие коэффициент автокорреляции будет возрастать.

## Листинг

gen\_1.py

```
1
2 import random
3
4 print(f'Генератор Random() ')
5 N = int(input("N: "))
6 K = int(input("K: "))
7 y = range(N)
8 x_1 = []
9 numbers_1 = [0] * K
10 ksi = 0
11
12
13 def gen_1():
14     for k in y:
15         rand_1 = random.uniform(0, 1)
16         x_1.append(rand_1)
17         inter_1 = rand_1 / (1 / K)
18         numbers_1[int(inter_1)] += 1
19
20 p_i = 1 / K
21 i = 0
22 ksi = 0
23 while i != K:
24     ksi += (((numbers_1[i] - p_i * N) ** 2) / (p_i * N))
25     i += 1
26 print(f'Хи квадрат: {ksi}')
27
28 i = 0
29 Ex = 0
30 while i != N:
31     Ex += x_1[i]
32     i += 1
33 Ex = (1 / N) * Ex
34
35 i = 0
36 S2x = 0
37 while i != N:
38     S2x += ((x_1[i]) ** 2) - (Ex ** 2)
39     i += 1
40 S2x = S2x / N
41
42 offset = 1
43 while offset <= K / 2:
44     a = 0
45     i = 0
46     while i < N - offset:
47         a += (x_1[i] - Ex) * (x_1[i + offset] - Ex)
48         i += 1
49     a = a / (N - offset) * S2x
50     print(f'a(tao) %f' % a)
51     print(f'tao: {offset}')
52     offset += 1
```

## gen\_2.py

```
1  from numpy.random import rand
2
3  print(f'Генератор модуля NumPy')
4  N = int(input("N: "))
5  K = int(input("K: "))
6  y = range(N)
7  x_2 = []
8  numbers_2 = [0] * K
9  ksi = 0
10 offset = 1
11
12
13 def gen_2():
14     for k in y:
15         rand_2 = rand()
16         x_2.append(rand_2)
17         inter_2 = rand_2 / (1 / K)
18         numbers_2[int(inter_2)] += 1
19
20     p_i = 1 / K
21     i = 0
22     ksi = 0
23     while i != K:
24         ksi += ((numbers_2[i] - p_i * N) ** 2) / (p_i * N)
25         i += 1
26     print(f'Хи квадрат: {ksi}')
27
28     i = 0
29     Ex = 0
30     while i != N:
31         Ex += x_2[i]
32         i += 1
33     Ex = (1 / N) * Ex
34
35     i = 0
36     S2x = 0
37     while i != N:
38         S2x += ((x_2[i]) ** 2) - (Ex ** 2)
39         i += 1
40     S2x = S2x / N
41
42     offset = 1
43     while offset <= K / 2:
44         a = 0
45         i = 0
46         while i < N - offset:
47             a += (x_2[i] - Ex) * (x_2[i + offset] - Ex)
48             i += 1
49         a = a / (N - offset) * S2x
50         print(f'a(tao) %f' % a)
51         print(f'tao: {offset}')
52         offset += 1
```

## gen\_3.py

```
1 import random
2
3 print(f'Генератор SystemRandom()')
4 N = int(input("N: "))
5 K = int(input("K: "))
6 y = range(N)
7 x_3 = []
8 numbers_3 = [0] * K
9 ksi = 0
10
11
12 def gen_3():
13     for k in y:
14         rand_3 = random.SystemRandom().random()
15         x_3.append(rand_3)
16         inter_3 = rand_3 / (1 / K)
17         numbers_3[int(inter_3)] += 1
18
19     p_i = 1 / K
20     i = 0
21     ksi = 0
22     while i != K:
23         ksi += (((numbers_3[i] - p_i * N) ** 2) / (p_i * N))
24         i += 1
25     print(f'Хи квадрат: {ksi}')
26
27     i = 0
28     Ex = 0
29     while i != N:
30         Ex += x_3[i]
31         i += 1
32     Ex = (1 / N) * Ex
33
34     i = 0
35     S2x = 0
36     while i != N:
37         S2x += ((x_3[i]) ** 2) - (Ex ** 2)
38         i += 1
39     S2x = S2x / N
40
41     offset = 1
42     while offset <= K / 2:
43         a = 0
44         i = 0
45         while i < N - offset:
46             a += (x_3[i] - Ex) * (x_3[i + offset] - Ex)
47             i += 1
48         a = a / (N - offset) * S2x
49         print(f'a(tao) %f' % a)
50         print(f'tao: {offset}')
51         offset += 1
```