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 7
Листинг 8

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 χ^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:

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

, 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

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

Если $x^2_{\text{эксп}} \leq x^2_{\text{таб}}$, 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 распределение 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.

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

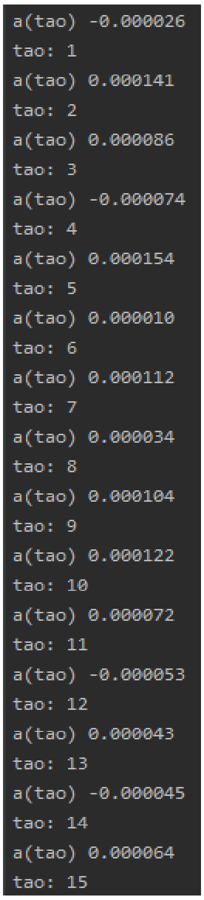
```
Генератор модуля NumPy
N: 1000000
K: 1000
Хи квадрат: 937.6499999999978
```

Рисунок 2 –rand() из модуля NumPy

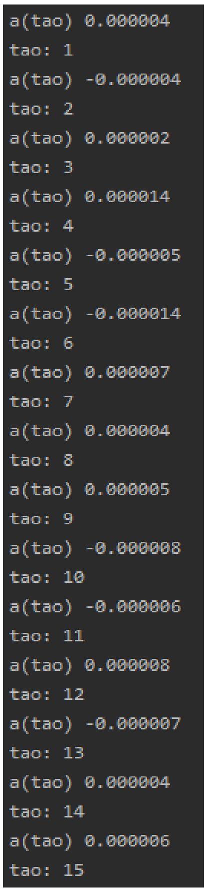
```
Генератор SystemRandom()
N: 1000000
K: 1000
Хи квадрат: 952.4699999999992
```

Рисунок 3 – SystemRandom()

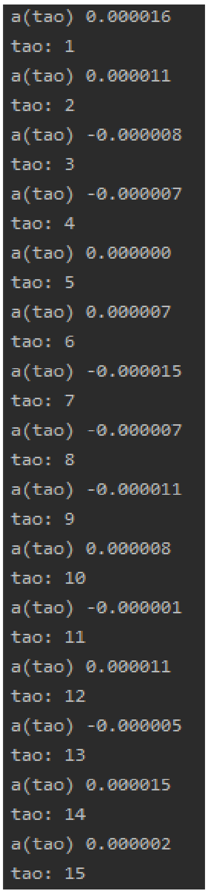
Так же была рассчитана автокорреляция для каждого ГСЧ при N = 1.000.000 и k = 1.000, где $\alpha(\tau)$ – автокорреляция, τ– смещение интервалов.



```
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
```



```
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
```



```
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 4, 5, 6 –Random(), Rand(), 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()	χ^2 = 993.72	χ^2 = 99.975	χ^2 = 987.6
NumPy Rand()	χ^2 = 953.382	χ^2 = 95.783	χ^2 = 979.399
SystemRamdom()	χ^2 = 998.081	χ^2 = 91.015	χ^2 = 1004.799
Табличное значение χ^2	χ^2 = 1142.848	χ^2 = 134,642	χ^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 χ^2 = 970.919, χ^2= 937.649 и χ^2 = 952.469 соответственно для каждого из них принимается гипотеза о равновероятном распределении в ГСЧ исходя их табличных значений для каждого из случая.

Из наших опытов так же видно, что функция автокорреляции стремится к нулю при каждом изменении τ (смещение последовательности). Что 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.

Во время проведения экспериментов 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.

Листинг

gen_1.py

```python
import random

print(f'Генератор Random()')
N = int(input("N: "))
K = int(input("K: "))
y = range(N)
x_1 = []
numbers_1 = [0] * K
ksi = 0


def gen_1():
    for k in y:
        rand_1 = random.uniform(0, 1)
        x_1.append(rand_1)
        inter_1 = rand_1 / (1 / K)
        numbers_1[int(inter_1)] += 1

    p_i = 1 / K
    i = 0
    ksi = 0
    while i != K:
        ksi += (((numbers_1[i] - p_i * N) ** 2) / (p_i * N))
        i += 1
    print(f'Хи квадрат: {ksi}')

    i = 0
    Ex = 0
    while i != N:
        Ex += x_1[i]
        i += 1
    Ex = (1 / N) * Ex

    i = 0
    S2x = 0
    while i != N:
        S2x += ((x_1[i]) ** 2) - (Ex ** 2)
        i += 1
    S2x = S2x / N

    offset = 1
    while offset <= K / 2:
        a = 0
        i = 0
        while i < N - offset:
            a += (x_1[i] - Ex) * (x_1[i + offset] - Ex)
            i += 1
        a = a / (N - offset) * S2x
        print(f'a(tao) %f' % a)
        print(f'tao: {offset}')
        offset += 1
```

gen_2.py

```
from numpy.random import rand

print(f'Генератор модуля NumPy')
N = int(input("N: "))
K = int(input("K: "))
y = range(N)
x_2 = []
numbers_2 = [0] * K
ksi = 0
offset = 1


def gen_2():
    for k in y:
        rand_2 = rand()
        x_2.append(rand_2)
        inter_2 = rand_2 / (1 / K)
        numbers_2[int(inter_2)] += 1

    p_i = 1 / K
    i = 0
    ksi = 0
    while i != K:
        ksi += (((numbers_2[i] - p_i * N) ** 2) / (p_i * N))
        i += 1
    print(f'Хи квадрат: {ksi}')

    i = 0
    Ex = 0
    while i != N:
        Ex += x_2[i]
        i += 1
    Ex = (1 / N) * Ex

    i = 0
    S2x = 0
    while i != N:
        S2x += ((x_2[i]) ** 2) - (Ex ** 2)
        i += 1
    S2x = S2x / N

    offset = 1
    while offset <= K / 2:
        a = 0
        i = 0
        while i < N - offset:
            a += (x_2[i] - Ex) * (x_2[i + offset] - Ex)
            i += 1
        a = a / (N - offset) * S2x
        print(f'a(tao) %f' % a)
        print(f'tao: {offset}')
        offset += 1
```

gen_3.py

```python
import random

print(f'Генератор SystemRandom()')
N = int(input("N: "))
K = int(input("K: "))
y = range(N)
x_3 = []
numbers_3 = [0] * K
ksi = 0


def gen_3():
    for k in y:
        rand_3 = random.SystemRandom().random()
        x_3.append(rand_3)
        inter_3 = rand_3 / (1 / K)
        numbers_3[int(inter_3)] += 1

    p_i = 1 / K
    i = 0
    ksi = 0
    while i != K:
        ksi += (((numbers_3[i] - p_i * N) ** 2) / (p_i * N))
        i += 1
    print(f'Хи квадрат: {ksi}')

    i = 0
    Ex = 0
    while i != N:
        Ex += x_3[i]
        i += 1
    Ex = (1 / N) * Ex

    i = 0
    S2x = 0
    while i != N:
        S2x += ((x_3[i]) ** 2) - (Ex ** 2)
        i += 1
    S2x = S2x / N

    offset = 1
    while offset <= K / 2:
        a = 0
        i = 0
        while i < N - offset:
            a += (x_3[i] - Ex) * (x_3[i + offset] - Ex)
            i += 1
        a = a / (N - offset) * S2x
        print(f'a(tao) %f' % a)
        print(f'tao: {offset}')
        offset += 1
```