

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

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №0**  
по дисциплине «Моделирование»

Выполнил:  
Студент гр. ИВ-622  
Курзин А.С.

Проверила:  
Ассистент Кафедры ВС  
Петухова Я.В.

Новосибирск 2020

## **СОДЕРЖАНИЕ**

ПОСТАНОВКА ЗАДАЧИ.....	3
ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ.....	3
ГЕНЕРАТОРЫ СЛУЧАЙНЫХ ЧИСЕЛ.....	5
РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ.....	5
ЗАКЛЮЧЕНИЕ.....	9
ЛИСТИНГ.....	10

## ПОСТАНОВКА ЗАДАЧИ

Необходимо взять готовую реализацию трёх генераторов псевдослучайных чисел и убедиться в их равномерном распределении, используя такие параметры как  $\chi^2$  и автокорреляция.

## 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  $X$ .

Гипотеза  $H'_0$ :  $X^n$  порождается ранее упомянутой функцией  $F'(x)$ . Разделим  $[a, b]$  на  $k$  непересекающихся интервалов  $[a_i, b_i], i = 1 \dots k$ .

Пусть  $n_j$  — количество наблюдений в  $j$ -м интервале;

$p_j = F(b_j) - F(a_j)$  - вероятность попадания наблюдения в  $j$ -й интервал при выполнении гипотезы  $H'_0$ ;

$E_j = n p_j$  - ожидаемое число попаданий в  $j$ -й интервал; тогда распределение Хи-квадрат с числом степеней свободы  $k-1$  будет иметь следующую статистику:

$$\chi^2 = \sum_{j=1}^k \frac{(n_j - E_j)^2}{E_j} \sim \chi_{k-1}^2$$

В зависимости от значения критерия  $\chi^2$  гипотеза  $H_0$  может приниматься либо отвергаться:

- $\chi_1^2 < \chi^2 < \chi_2^2$  - гипотеза  $H_0$  выполняется.
- $\chi^2 \leq \chi_1^2$  — попадает в левый «хвост» распределения. Следовательно, теоретические и практические значения очень близки. Если, к примеру, происходит проверка генератора случайных чисел, который сгенерировал  $n$  чисел из отрезка  $[0, 1]$  и выборка  $X^n$  распределена равномерно на  $[0, 1]$ , то генератор

нельзя называть случайным (гипотеза случайности не выполняется), т.к. выборка распределена слишком равномерно, но гипотеза  $H_0$  выполняется.

- $\chi^2 \geq \chi^2_{\alpha}$  — попадает в правый «хвост» распределения, гипотеза  $H_0$  отвергается.

Автокорреляционная функция – предназначена для оценки корреляции между смещенными копиями последовательностей.

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

$Ex$  - математическое ожидание – среднее значение случайной величины при стремлении количества выборок или количества её измерений (иногда говорят – количества испытаний) к бесконечности:

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

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

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

$x_{i+\tau}$  – множество значений другой случайной величины (полученной из значений прошлой случайной величины, но с некоторым смещением);

$n$  – мощность множества случайных величин;

$\tau$  – смещение последовательности.

## ГЕНЕРАТОРЫ СЛУЧАЙНЫХ ЧИСЕЛ

В качестве генератора случайных чисел были взяты:

- Стандартный генератор для языка Python
- Второй генератор случайных чисел альтернативный стандартному – SystemRandom. В модуле random есть класс SystemRandom, внутри реализации которого идет обращение как раз к os.urandom. В этом альтернативном генераторе реализованы те же функции. Она выдает так же псевдослучайные данные, но они зависят дополнительно и от операционной системы. Результаты используются в криптографии. Есть недостаток – то что функции SystemRandom отрабатывают в несколько раз дольше.
- Встроенный в NumPy генератор псевдослучайных чисел в подмодуле random. Числа являются псевдослучайными, в том плане что, они сгенерированы детерминистически из порождающего элемента (seed number), но рассредоточены в статистическом сходстве с случайным образом. Для генерации NumPy использует особенный алгоритм, который имеет название Mersenne Twister – это генератор случайных чисел на основе алгоритма «Вихря Мерсенна». Вихрь Мерсенна — генератор псевдослучайных чисел основывается на свойствах простых чисел Мерсенна (отсюда название) и обеспечивает быструю генерацию высококачественных по критерию случайности псевдослучайных чисел. Вихрь Мерсенна лишён многих недостатков, присущих другим ГПСЧ, таких как малый период, предсказуемость, легко выявляемые статистические закономерности.

## РЕЗУЛЬТАТЫ РАБОТЫ ПРОГРАММЫ

При помощи генератора псевдослучайных чисел было сгенерировано 10 000, и 1 00 000 случайных значений в диапазоне от 0 до 1. Диапазон от 0 до 1 был поделен на 1 00 и 1 000 равномерных отрезков. Далее считаем количество попаданий на отрезки.

```

Стандартный рандом:
Всего точек:10000
Всего интервалов:1000
Хи квадрат:1059.0000000000018
Всего точек:1000000
Всего интервалов:1000
Хи квадрат:978.4779999999993
Всего точек:1000000
Всего интервалов:100
Хи квадрат:110.09759999999997

```

Рисунок 1– Результаты вычислений «хи-квадрат» для ГСЧ: Стандартный random

```

SystemRandom:
Всего точек:10000
Всего интервалов:1000
Хи квадрат:1015.1999999999997
Всего точек:1000000
Всего интервалов:1000
Хи квадрат:973.8359999999997
Всего точек:1000000
Всего интервалов:100
Хи квадрат:96.95940000000002

```

Рисунок 2 – Результаты вычислений «хи-квадрат» для ГСЧ: SystemRandom

```

NumPy:
Всего точек:10000
Всего интервалов:1000
Хи квадрат:979.3999999999992
Всего точек:1000000
Всего интервалов:1000
Хи квадрат:919.39000000000011
Всего точек:1000000
Всего интервалов:100
Хи квадрат:110.7326

```

Рисунок 3 – Результаты вычислений «хи-квадрат» для ГСЧ: NumPY

Таблица №1

N - количество чисел		N = 10 000	N = 1 000 000	N = 1 000 000
k - количество интервалов		k = 1000	k = 1 000	k = 100
Генераторы случайных чисел	Стандартный random языка Python	$\chi^2_{\text{эксн, random}} = 1059$	$\chi^2_{\text{эксн, random}} = 978.48$	$\chi^2_{\text{эксн, random}} = 110.098$
	SystemRandom	$\chi^2_{\text{эксн, SystemRandom}} = 1015.2$	$\chi^2_{\text{эксн, SystemRandom}} = 973.83$	$\chi^2_{\text{эксн, SystemRandom}} = 96.96$
	NumPy (Marsenne twister)	$\chi^2_{\text{эксн, NumPy}} = 979.4$	$\chi^2_{\text{эксн, NumPy}} = 929.39$	$\chi^2_{\text{эксн, NumPy}} = 110.73$

Была рассчитана автокорреляция для каждого ГСЧ при N = 1.000.000 и k = 1.000, где  $a(\tau)$  – автокорреляция,  $\tau$  – смещение интервалов. В таблице №1 показаны результаты генерации случайных чисел каждым ГСЧ с изменением количества чисел и количества интервалов.

```

t: 1
a(t): 0.00010639002976257232
t: 2
a(t): 0.0001531011200339764
t: 3
a(t): -2.526631517712201e-06
t: 4
a(t): 4.890486070151586e-05
t: 5
a(t): 0.00011497246716318977
t: 6
a(t): 8.90450744154203e-05
t: 7
a(t): 8.942178737909984e-05
t: 8
a(t): 2.8275804883855447e-05
t: 9
a(t): -9.589755198619878e-06
t: 10
a(t): -0.00013847919438545294
t: 11
a(t): 3.6816721869393716e-05
t: 12
a(t): 1.893196965780562e-05
t: 13
a(t): 0.0001785454738490241
t: 14
a(t): 3.5071511301123114e-05
t: 15
a(t): 0.00021921420182309138
t: 16
a(t): -0.00019234487412636613
t: 17
a(t): 2.165977064603795e-05
t: 18
a(t): 9.253385153656658e-05
t: 19
a(t): -0.0001019551031168983
t: 20
a(t): -6.065511204239002e-05

```

Рисунок 4 – Результат расчета коэффициента корреляции генератором: NumPY

```

t: 1
a(t): 0.00010291303734568405
t: 2
a(t): -4.12562571198859e-05
t: 3
a(t): 4.8275327769102225e-05
t: 4
a(t): -0.00010361024716407265
t: 5
a(t): 5.918559601754087e-06
t: 6
a(t): 5.122101432146405e-05
t: 7
a(t): 4.861363635746295e-05
t: 8
a(t): 6.022226045519005e-05
t: 9
a(t): 4.8593324447173345e-05
t: 10
a(t): 4.1223009120397364e-05
t: 11
a(t): 5.248771555848274e-05
t: 12
a(t): -8.653071317065792e-05
t: 13
a(t): 0.0001659487681962547
t: 14
a(t): -5.956536659101994e-05
t: 15
a(t): -6.019763862936095e-05
t: 16
a(t): 0.00010264276112142204
t: 17
a(t): -0.00012747774308036398
t: 18
a(t): -0.00012857861162284006
t: 19
a(t): 2.0667351866022685e-05
t: 20
a(t): -8.798963592830245e-05

```

Рисунок 5 – Результат расчета коэффициента корреляции генератором: SystemRandom

```
t: 1
a(t): 4.133635459390662e-05
t: 2
a(t): -7.019543080487497e-05
t: 3
a(t): 7.391313343756057e-05
t: 4
a(t): -5.0457459840988267e-05
t: 5
a(t): -5.2453308366165746e-05
t: 6
a(t): 0.00012133899834866426
t: 7
a(t): 3.976695253710333e-05
t: 8
a(t): 2.5532399273116586e-05
t: 9
a(t): -1.7804669934585084e-06
t: 10
a(t): -5.756187947247577e-05
t: 11
a(t): 1.5204482583575554e-05
t: 12
a(t): -8.623909507708797e-05
t: 13
a(t): -8.010060331563146e-05
t: 14
a(t): 7.642128531715837e-05
t: 15
a(t): -9.877692137467345e-05
t: 16
a(t): -7.3861273082270945e-06
t: 17
a(t): 3.855889533340398e-05
t: 18
a(t): -0.0001976470912450474
t: 19
a(t): 0.00010680436379290887
t: 20
a(t): 0.00015402516140643386
```

Рисунок 6 – Результат расчета коэффициента корреляции  
Генератором: Стандартный random

Исходя из выше представленных данных можно сделать вывод, что числа действительно являются случайными, так как значения автокорреляции стремятся к нулю.



## ЗАКЛЮЧЕНИЕ

В рамках лабораторной работы были изучены методы тестирования качества работы генератора псевдослучайных чисел и проведены эксперименты по критерий Пирсона и автокорреляции.

Из результатов эксперимента, мы можем увидеть, что критерии «хи-квадрат» примерно равны  $\chi^2_{\text{эксп, random}} = 945.899$ ,  $\chi^2_{\text{эксп, SystemRandom}} = 1026.049$ ,  $\chi^2_{\text{эксп, NumPy}} = 1028.827$ , также можем заметить, что «хи-квадрат» табличного равен  $\chi^2_{\text{таб}} = 1142.848$ . Таким образом,  $\chi^2_{\text{эксп}} < \chi^2_{\text{таб}}$  и отсюда мы можем сделать вывод о том, что гипотезы о равновероятном распределении в генераторах случайных чисел принимается. В противном случае, если бы значение  $\chi^2_{\text{эксп}}$  было бы равно или больше чем  $\chi^2_{\text{таб}}$ , то гипотеза была бы отклонена. Следовательно, для всех тех трех различных генераторов, которые мы взяли, гипотеза принимается.

Так же стоит заметить, что значение автокорреляции приближено к нулю, что говорит о том, что статистической взаимосвязи между исходной и сдвинутой последовательности нет. Коэффициент автокорреляции напрямую зависит от количества точек на интервалах. Если уменьшить количество интервалов, то точек в каждом интервале становится больше и график будет более равномерным, а коэффициент корреляции приближается к нулевому значению. Если уменьшить количество точек, то количество точек попавших на интервал становится меньше.

## ЛИСТИНГ

```
import random
import numpy as np
from math import sqrt
import typing as tp

def get_float_rand() -> float:
    return float(random.random())

def get_SystemRandom() -> float:
    r = random.SystemRandom()
    rand = r.random()
    return float(rand)

def get_NumPy_rand() -> float:
    rnd = np.random.sample()
    return float(rnd)

def mat_waiting(N: int, x: tp.Sequence[int]) -> float:
    i = 0
    Ex = 0
    while i < N:
        Ex += x[i] * (1 / N)
        i += 1

    return float(Ex)

def sample_variance(N: int, x: tp.Sequence[int]) -> float:
    i = 0
    Sx = 0
    Ex = mat_waiting(N, x)
    while i < N:
        Sx += (x[i] + Ex) ** 2
        i += 1

    Sx = Sx / N

    return float(Sx)

def autocorrelation(offset: int, N: int, x: tp.Sequence[int], Ex: float, S2x: float) -> float:
    R = 0
    i = 0
```

```

while i < N - offset:
    R += (x[i] - Ex) * (x[i+offset] - Ex)
    i += 1

R = R / ((N - offset) * S2x)

return float(R)

N = 1000000
K = 1000
y = range(N)
x = []
array = [0] * K
ksi = 0
offset = 1
for k in y:
    rand = get_float_rand()
    x.append(rand)
    inter = rand / (1 / K)
    array[int(inter)] += 1

print('Всего точек:' + str(N))
print('Всего интервалов:' + str(K))

avg = 0
for n in x:
    avg += n
avg = avg / (N)
print('Сред. арифм. :' + str(avg))

i = 0
summary = 0
while i < len(x):
    summary += (x[i] - avg) ** 2
    i += 1
deviation = sqrt((1 / N) * summary)
print('Сред. квадр. откл. :' + str(deviation))

p_i = 1 / K

i = 0
while i < K:
    ksi += (((array[i] - p_i * N) ** 2) / (p_i * N))
    i += 1
print('Кси квадрат:' + str(ksi))

```

```
Ex = mat_waiting(N, x)
S2x = sample_variance(N, x)
i = 0
offset = 1
while i < K / 2:
    R = autocorrelation(offset, N, x, Ex, S2x)
    print('offset: ' + str(offset))
    print('R: ' + str(R))

    i += 1
    offset += 1
```