Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования
«Национальный исследовательский университет «Высшая школа экономики»

Кафедра «Компьютерная безопасность»

ОТЧЁТ
К ЛАБОРАТОРНОЙ РАБОТЕ №4
по дисциплине
«Методы программирования»

Работу выполнил студент группы СКБ202 _____ Белов Н.С.
Подпись, дата

Работу проверил _____ Драчев Г.А.
Подпись, дата

Москва 2023

Постановка задачи

1. Модифицировать (предложить собственные) два метода генерации псевдослучайных чисел.
2. Получить не менее 10 выборок каждым методом (диапазон чисел в каждой выборке не менее 10000) объемом не менее 50 элементов каждая.
3. Для каждой выборки посчитать среднее, отклонение и коэффициент вариации. Сделать вывод об однородности выборки.
4. Каждую выборку проверить на равномерность распределения и случайность, используя критерий Хи-квадрат.
5. Засечь время генерации чисел от тысячи до миллиона элементов обоими предложенными методами и любым стандартным методом используемого языка программирования. Построить графики сравнения скоростей в зависимости от объема выборки.
6. В отчете обязательно отразить: код алгоритма генерации и критерия проверки гипотезы, скриншот с результатами выполнения, анализ полученных результатов и выводы.

1. Алгоритм решения задачи

Данная лабораторная работа была реализована на языке программирования Python 3.9.1. Графики были построены с помощью Wolfram Mathematica. В файле lab4.py определены функции для анализа полученной выборки, а именно функция sample_average(sample: list) для подсчёта выборочного среднего, функция sample_variance(sample: list) для подсчёта выборочной дисперсии, chi_square(sample: list) для определения случайности и равномерности распределения с использованием критерия хи-квадрат и функция gen_params(gen) генерирует выборку, с использованием генератора псевдо случайных чисел, и вычисляет её параметры. В файле gen.py определены генераторы псевдослучайных чисел. Функция linear_congruent_method(size) генерирует псевдослучайные числа с помощью линейного конгруэнтного метода, функция middle_products(size) генерирует псевдослучайные числа с помощью метода срединных произведений и функция std_randint(size) генерирует псевдослучайные числа с помощью встроенного генератора.

2.Выполнение задания

2.1. Функции sample_average(sample: list), sample_variance(sample: list).

Функция sample_average(sample: list) возвращает выборочное среднее выборки sample, переданной в функцию. Функция sample_variance(sample: list) возвращает выборочную дисперсию выборки sample, переданной в функцию.

2.2. Функция chi_square(sample: list).

Функция chi_square(sample: list) реализует критерий хи-квадрат, с помощью которого определяется случайность и равномерность распределения выборки sample, переданной в функцию.

2.3. Функция gen_params(sample: list).

Функция gen_params(sample: list) осуществляет подсчёт параметров выборки sample, сгенерированной с помощью генератора псевдослучайных чисел и проверку критерия хи-квадрат. Выводит полученные результаты на экран.

2.4. Функции linear_congruent_method(size), middle_products(size), std_randint(size).

Функции linear_congruent_method(size), middle_products(size) и std_randint(size) осуществляют генерацию последовательности псевдослучайных чисел размером size, линейным конгруэнтным методом, методом срединных произведений и с помощью встроенного генератора, соответственно.

3. Результаты работы программы

```
Линейный конгруэнтный метод:
Размер выборки: 50
Среднее: 0.512625
Дисперсия: 0.078932
Отклонение: 0.280949
Коэффициент вариации: 0.548059
Значение статистики: 0.400968
Равномерность: Принимается: уровень значимости >= 0.99
Случайность: Отвергается

Размер выборки: 100
Среднее: 0.495965
Дисперсия: 0.08334
Отклонение: 0.288687
Коэффициент вариации: 0.582071
Значение статистики: 0.099261
Равномерность: Принимается: уровень значимости >= 0.99
Случайность: Отвергается

Размер выборки: 500
Среднее: 0.509845
Дисперсия: 0.082971
Отклонение: 0.288047
Коэффициент вариации: 0.564969
Значение статистики: 2.269979
Равномерность: Принимается : уровень значимости (0.95, 0.99]
Случайность: Принимается : уровень значимости (0.95, 0.99]
```

```
Размер выборки: 1000
Среднее: 0.500105
Дисперсия: 0.083586
Отклонение: 0.289112
Коэффициент вариации: 0.578102
Значение статистики: 0.077643
Равномерность: Принимается: уровень значимости >= 0.99
Случайность: Отвергается

Размер выборки: 5000
Среднее: 0.50104
Дисперсия: 0.083625
Отклонение: 0.289179
Коэффициент вариации: 0.577158
Значение статистики: 0.013018
Равномерность: Принимается: уровень значимости >= 0.99
Случайность: Отвергается

Размер выборки: 10000
Среднее: 0.501404
Дисперсия: 0.08365
Отклонение: 0.289223
Коэффициент вариации: 0.576825
Значение статистики: 0.590061
Равномерность: Принимается: уровень значимости >= 0.99
Случайность: Отвергается
```

```
Размер выборки: 50000
Среднее: 0.50092
Дисперсия: 0.083649
Отклонение: 0.289221
Коэффициент вариации: 0.577379
Значение статистики: 0.00448
Равномерность: Принимается: уровень значимости >= 0.99
Случайность: Отвергается

Размер выборки: 100000
Среднее: 0.5009
Дисперсия: 0.083649
Отклонение: 0.289221
Коэффициент вариации: 0.577402
Значение статистики: 0.003994
Равномерность: Принимается: уровень значимости >= 0.99
Случайность: Отвергается

Размер выборки: 500000
Среднее: 0.501003
Дисперсия: 0.083638
Отклонение: 0.289203
Коэффициент вариации: 0.577247
Значение статистики: 0.26427
Равномерность: Принимается: уровень значимости >= 0.99
Случайность: Отвергается
```

```
Размер выборки: 1000000
Среднее: 0.500949
Дисперсия: 0.083649
Отклонение: 0.289222
Коэффициент вариации: 0.577348
Значение статистики: 1.206197
Равномерность: Принимается: уровень значимости >= 0.99
Случайность: Отвергается

Размер выборки: 2000000
Среднее: 0.500918
Дисперсия: 0.08365
Отклонение: 0.289222
Коэффициент вариации: 0.577384
Значение статистики: 0.000236
Равномерность: Принимается: уровень значимости >= 0.99
Случайность: Отвергается

Метод серединных произведений:
Размер выборки: 50
Среднее: 0.46376
Дисперсия: 0.092939
Отклонение: 0.304858
Коэффициент вариации: 0.657363
Значение статистики: 12.155402
Равномерность: Отвергается
Случайность: Отвергается
```

```
Размер выборки: 100
Среднее: 0.494266
Дисперсия: 0.0857
Отклонение: 0.292745
Коэффициент вариации: 0.592282
Значение статистики: 2.062043
Равномерность: Принимается : уровень значимости (0.9, 0.95]
Случайность: Принимается : уровень значимости (0.9, 0.95]

Размер выборки: 500
Среднее: 0.504622
Дисперсия: 0.086551
Отклонение: 0.294195
Коэффициент вариации: 0.583002
Значение статистики: 2.450062
Равномерность: Принимается : уровень значимости (0.95, 0.99]
Случайность: Принимается : уровень значимости (0.95, 0.99]

Размер выборки: 1000
Среднее: 0.498527
Дисперсия: 0.085012
Отклонение: 0.291569
Коэффициент вариации: 0.584861
Значение статистики: 4.659012
Равномерность: Отвергается
Случайность: Отвергается
```

```
Размер выборки: 5000
Среднее: 0.50189
Дисперсия: 0.084462
Отклонение: 0.290623
Коэффициент вариации: 0.579058
Значение статистики: 6.686389
Равномерность: Отвергается
Случайность: Отвергается

Размер выборки: 10000
Среднее: 0.499749
Дисперсия: 0.083856
Отклонение: 0.289579
Коэффициент вариации: 0.579449
Значение статистики: 5.6531
Равномерность: Принимается : уровень значимости (0.95, 0.99]
Случайность: Принимается : уровень значимости (0.95, 0.99]

Размер выборки: 50000
Среднее: 0.500842
Дисперсия: 0.083675
Отклонение: 0.289267
Коэффициент вариации: 0.577561
Значение статистики: 0.14336
Равномерность: Принимается: уровень значимости >= 0.99
Случайность: Отвергается
```

```
Размер выборки: 100000
Среднее: 0.501048
Дисперсия: 0.083669
Отклонение: 0.289255
Коэффициент вариации: 0.5773
Значение статистики: 0.758446
Равномерность: Принимается: уровень значимости >= 0.99
Случайность: Отвергается

Размер выборки: 500000
Среднее: 0.500885
Дисперсия: 0.083648
Отклонение: 0.28922
Коэффициент вариации: 0.577419
Значение статистики: 2.124392
Равномерность: Принимается: уровень значимости >= 0.99
Случайность: Отвергается

Размер выборки: 1000000
Среднее: 0.500855
Дисперсия: 0.08365
Отклонение: 0.289223
Коэффициент вариации: 0.577458
Значение статистики: 5.326766
Равномерность: Принимается: уровень значимости >= 0.99
Случайность: Отвергается
```

```
Размер выборки: 2000000
Среднее: 0.500881
Дисперсия: 0.083649
Отклонение: 0.289222
Коэффициент вариации: 0.577426
Значение статистики: 0.351163
Равномерность: Принимается: уровень значимости >= 0.99
Случайность: Отвергается

Время генерации:
    Линейный конгруэнтный метод: [0.0, 0.0, 0.003331, 0.013896, 0.010246, 0.020815, 0.044309, 0.056436, 0.080178, 0.284648, 0.439323, 0.599688, 0.968591]
    Метод серединных произведений: [0.0, 0.0, 0.0, 0.001043, 0.0, 0.025868, 0.024292, 0.040251, 0.052723, 0.158129, 0.331277, 0.613759, 0.666377]
    Стандартный способ: [0.009854, 0.008126, 0.006696, 0.024704, 0.034693, 0.066855, 0.106021, 0.154112, 0.198812, 0.548144, 1.110113, 1.639893, 2.21494]
```

4. Построение графиков и выводы

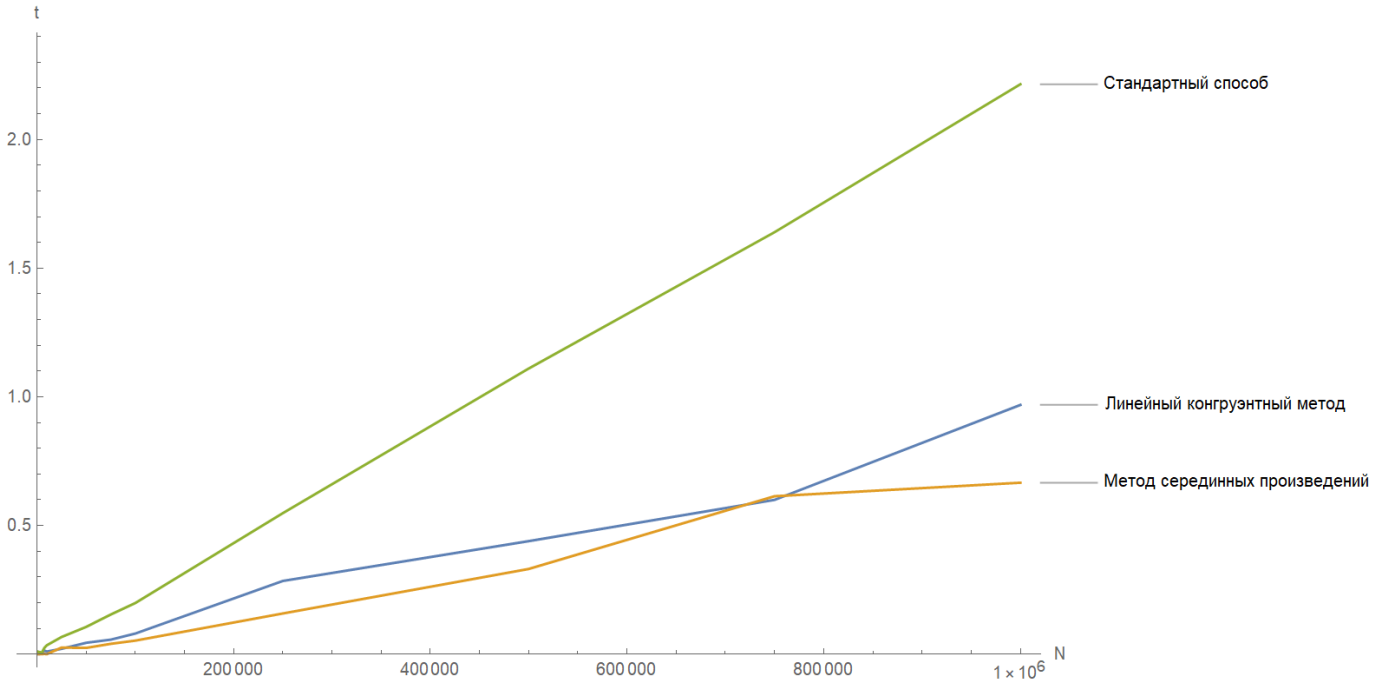

Рисунок 1 - результаты работы программы

Как видно из графиков, генерация методом серединных произведений и линейным конгруэнтным методом работает быстрее, чем с использованием встроенного генератора. Так же гипотеза о равномерности выборки, сгенерированной методом серединных произведений, принимается в большем числе случаев и имеет больший уровень значимости, чем при генерации методом срединных произведений.

Приложение А

Исходный код файла **lab.py**

```
import time
import numpy as np
from gen import linear_congruent_method, middle_products, std_randint
from math import floor, log2

significance_level = [0.99, 0.95, 0.90] # уровень значимости

chi_table = {            # таблица статистики хи-квадрат
    5: [0.55, 1.15, 1.61],
    6: [0.87, 1.64, 2.20],
    7: [1.24, 2.18, 2.83],
    8: [1.65, 2.73, 3.49],
    9: [2.09, 3.33, 4.17],
    10: [2.56, 3.94, 4.87],
    11: [3.05, 4.57, 5.58],
    12: [3.57, 5.23, 6.30],
    13: [4.11, 5.89, 7.04],
```

```
    15: [5.23, 7.26, 8.5],
    16: [5.81, 7.98, 9.31],
    17: [6.41, 8.67, 10.09],
    18: [7.02, 9.39, 10.87],
    19: [7.63, 10.1, 11.7],
    20: [8.26, 10.9, 12.4],
    21: [8.90, 11.56, 13.2],
    22: [9.54, 12.34, 14.04],

}


def sample_average(sample: list) -> float:
    """
    Вычисление выборочного среднего

    :param sample: выборка
    :type sample: list

    :return: выборочное среднее
    :rtype: float
    """
    return sum(sample) / len(sample)


def sample_variance(sample: list) -> float:
    """
    Вычисление выборочной дисперсии

    :param sample: выборка
    :type sample: list

    :return: выборочная дисперсия
    :rtype: float
    """
    m = sample_average(sample)
    summ = 0
    for i in sample:
        summ += (i - m) * (i - m)

    return summ / len(sample)


def chi_square(sample: list) -> tuple:
    """
    Используя критерий хи-квадрат, определяем случайность и равномерность распределения
```

```
    :param sample: выборка
    :type sample: list

    :return: значение статистики, а также строковые описания
    :rtype: tuple
    """
    a = 0
    theta = 16384
    N = len(sample) # объём выборки
    k = 1 + floor(log2(N)) # количество интервалов, вычисляется по формуле Старджеса
    intervals = np.arange(a, a + theta, (theta - 1) / k) # список интервалов

    prob_intervals = [] # список вероятностей попадания в интервал
    for i in range(len(intervals) - 1):
        left = np.ceil(intervals[i])
        right = np.floor(intervals[i+1])
        if intervals[i+1] == right and right != 16383:
            right -= 1
        prob_intervals.append((right - left + 1) / theta)
    intervals[-1] += 1

    intervals_count = [0] * k # список количества элементов выборки, попадающих в интервал
    for num in sample:
        for i in range(len(intervals) - 1):
            if intervals[i] <= num < intervals[g + 1]:
                intervals_count[i] += 1

    summ = 0
    for j in range(k):
        summ += intervals_count[j] ** 2 / (N * prob_intervals[j])

    v = summ - N
    sig_level_line = chi_table[k - 1]

    if v < sig_level_line[0]: # Если уровень значимости больше 0.99, то выборка равномерна и не случайна
        return (f"Принимается: уровень значимости >= {max(significance_level)}", "Отвергается", v)
    elif v > sig_level_line[2]: # Если уровень значимости меньше 0.90, то выборка не равномерна и не случайна
        return ("Отвергается", "Отвергается", v)

    st = ""
    for i in range(len(significance_level) - 1): # Если уровень значимости между 0.99 и 0.90, то выборка равномерна и случайна
        if sig_level_line[i] <= v <= sig_level_line[i+1]:
            st = f": уровень значимости ({significance_level[i+1]}, {significance_level[i]})"
    return ("Принимается " + st, "Принимается " + st, v)
```

```
def gen_params(sample: list):
    """
    Вычисляет параметры выборки и, использую критерий хи-квадрат, определяется
    случайность и равномерность распределения

    :param sample: выборка
    :type sample: list
    """

    norm_sample = [i / 16353 for i in sample] # нормированная выборка
    mean = sample_average(norm_sample) # выборочное среднее
    dispersion = sample_variance(norm_sample) # выборочное среднее
    standart_deviation = dispersion ** (1 / 2) # отклонение
    variation_coefficient = standart_deviation / mean # коэффициент вариации
    r1, r2, val = chi_square(sample)  # критерий хи-квадрат

    print(
        f"Размер выборки: {len(sample)}",
        f"Среднее: {round(mean, 6)}",
        f"Дисперсия: {round(dispersion, 6)}",
        f"Отклонение: {round(standart_deviation, 6)}",
        f"Коэффициент вариации: {round(variation_coefficient, 6)}",
        f"Значение статистики: {round(val, 6)}",
        f"Равномерность: {r1}",
        f"Случайность: {r2}\n",
        sep="\n"
    )


if __name__ == "__main__":

    print("Линейный конгруэнтный метод:")
    for i in [50, 100, 500, 1000, 5000, 10000, 50000, 100000, 500000, 1000000, 2000000]:
        gen_params(linear_congruent_method(i))

    print("Метод серединных произведений:")
    for i in [50, 100, 500, 1000, 5000, 10000, 50000, 100000, 500000, 1000000, 2000000]:
        gen_params(middle_products(i))

    linear_congruent_time_lst = []
    middle_products_time_lst = []
    std_time_lst = []
    for g in [1000, 2500, 5000, 7500, 10000, 25000, 50000, 75000, 100000, 250000, 500000, 750000,
1000000]:
        time_start = time.time()
```

```
    linear_congruent_method(g)
    linear_congruent_time_lst.append(round(time.time() - time_start, 6))

    time_start = time.time()
    middle_products(g)
    middle_products_time_lst.append(round(time.time() - time_start, 6))

    time_start = time.time()
    std_randint(g)
    std_time_lst.append(round(time.time() - time_start, 6))

print("Время генерации:")
print("\tЛинейный конгруэнтный метод:", linear_congruent_time_lst)
print("\tМетод серединных произведений:", middle_products_time_lst)
print("\tСтандартный способ:", std_time_lst)
```

Исходный код файла **gen.py**

```
from random import randint
import time


def linear_congruent_method(size):
    """
    Линейный конгруэнтный метод [0, 16383]

    :param size: количество генерируемых чисел
    :type size: int

    :return: список псевдослучайных чисел
    :rtype: list
    """
    res = []
    M = (1 << 63) - 1

    k = 1 << 63
    b = int(time.perf_counter_ns() // 100)
    if b == M:
        b -= 1
    r0 = 13

    for i in range(size):
        r0 = (k * r0 + b) % M
        res.append(r0 % 16384)

    return res
```

```
def middle_products(size):
    """
    Метод серединных произведений [0, 16383]

    :param size: количество генерируемых чисел
    :type size: int

    :return: список псевдослучайных чисел
    :rtype: list
    """
    r0 = int(time.time()) % 128 + 1
    r1 = int(time.time()) % 128 + 1
    b = 11
    rez = []

    for i in range(size):
        r = (r0 * r1 * b) & 16383
        rez.append(r)
        r0 = r1
        r1 = r
        r0 += 13
        r1 += 17
        b += 2
    return rez


def std_randint(size):
    """
    Встроенный генератор псевдослучайных последовательностей [0, 16383]

    :param size: количество генерируемых чисел
    :type size: int

    :return: список псевдослучайных чисел
    :rtype: list
    """
    res = []
    for i in range(size):
        res.append(randint(0, 16384))
    return res
```

Ссылка на репозиторий: https://github.com/nbs13372/lab4.git