БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №1

Выполнила: Анейчик Ольга
4 курс 2 группа
Преподаватель: Кирлица В.П.

Минск 2018

Задание 1.

1. Постановка задачи

С помощью моделирования M = 100 реализаций СВ $\xi = \Pi(\lambda)$ и СВ $\eta = N_1(\lambda, \lambda)$ исследовать возможность и точность аппроксимации распределения Пуассона нормальным. Рассмотреть случаи: λ = 6, 9, 12, 15, 18.

2. Формулы и краткие пояснения к ним

Для генерации БСВ используется *мультипликативно конгруэнтный метод*:

$$X_{k+1} = (a * X_k + c)\ mod\ m; \qquad m > 0, \qquad 0 \le a, c, X_0\ \le m$$

где a - множитель, c – приращение, m – модуль, X_0 – начальное значение

Получаемая последовательность зависит от выбора стартового числа X_0и при разных его значениях получаются различные последовательности случайных чисел. Для выбора коэффициентов имеются свойства позволяющие максимизировать длину периода (максимальная длина равна m), то есть момент, с которого генератор зациклится.

С целью оптимизации скорости вычислений значение параметра m выбирается согласно разрядности компьютера, например для 32-разрядной машины:

$$m = 2^{32} + 1 = 2\ 147\ 483\ 649;$$

Если Z – основание основание системы счисления, которое используется в машине, e – разрядность, то коэффициент a вычисляется следующим образом:

$$a = Z^k + 1, \qquad 2 \le k < e$$

В контекте нашей задачи возьмем $a = c = 32\ 769, k = 15$.

Моделирование Пуассоновской СВ

Для генерации Пуассоновской СВ с параметром распределения λ используется следующий алгоритм (псевдокод):

$n \coloneqq 0$
$\alpha = generate\ basic\ random\ value(\,)$
$while\ \alpha \ge\ e^{-\lambda}$:
$\quad \alpha *= generate\ basic\ random\ value(\,)$
$\quad n += 1$

На выходе значение n – значение Пуассоновской СВ.

Моделирование нормальной СВ

Из курса теории вероятностей известно, что случайные величины стандартного нормального распределения и нормального распределения обладают свойством:

$$\xi = \eta + \sigma\eta, \qquad \text{где}\ \xi\ \in N(m, \sigma^2), \eta \in N(0,1)$$

Таким образом, задача моделирования ξ сводится к моделированию стандартной гаусовской СВ η.

Алгоритм моделирования реализуем методом суммирования, основанном на центральной предельной теореме: если $\alpha_1, \alpha_2, \dots, \alpha_N$ - независимые БСВ, то при $N \to \infty$ случайная величина распределена асимптотически нормально:

$$S_N = \sum_{i=1}^{N} \alpha_i$$

$$\frac{S_N - \mu n}{\sigma\sqrt{n}} \xrightarrow[n\to\infty]{} N(0,1)$$

Введем случайную величину $z \in N(0,1)$:

$$z = \frac{S_N - N/2}{\sqrt{N/12}}$$

Тогда алгоритм моделирования $\xi \in N(m, \sigma^2)$:

$$\xi = m + \frac{\sigma}{\sqrt{N/12}} * (R_i - N/2)$$

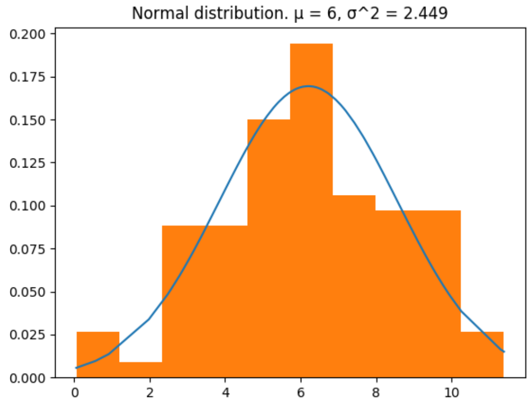
$$\xi = m + \sigma\left(\sum_{i=1}^{12} S_i - 6\right) \text{при } N = 12$$

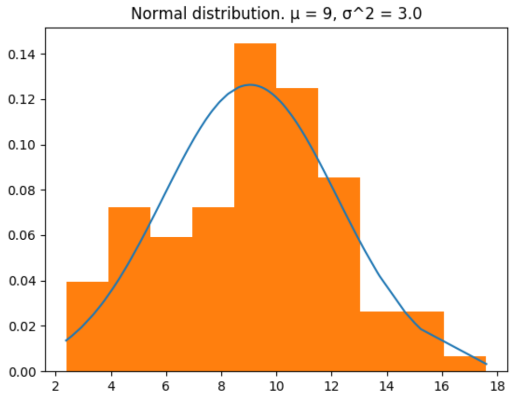
Запишем псведокод алгоритма генерации $\xi \in N(m, \sigma^2)$:

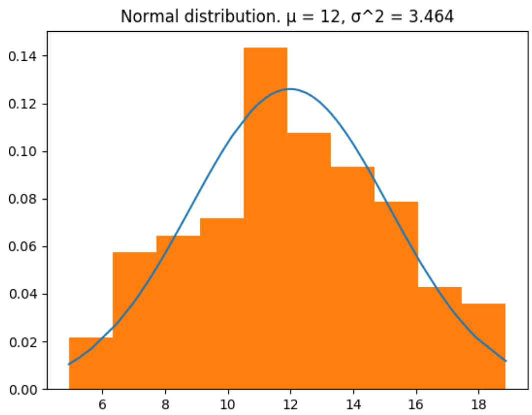
$\alpha_i = generate\ basic\ random\ value(\ ),\ \ i = \overline{1, 12}$
$\xi = \sum_{i=1}^{12} \alpha_i - 6$

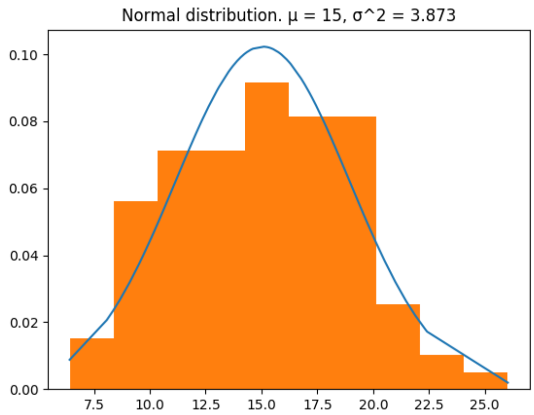
3. Выходные данные

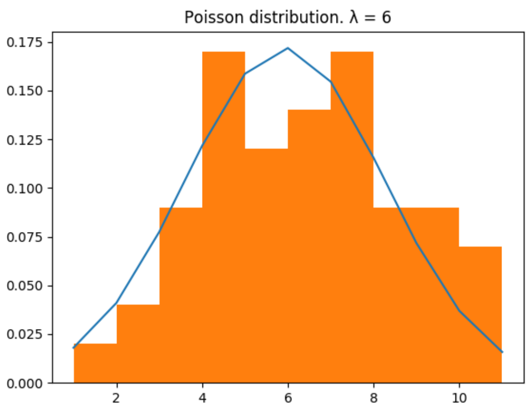
λ	Poisson(Mean)	Normal(Mean)	Poisson(Variance)	Normal(Variance)
6	6	6.3	4.04	6.76
9	8.42	9.12	8.6636	9.7969
12	12.28	11.9	10.6816	13.69
15	14.97	15.1	14.3491	15.8404
18	17.79	17.92	15.7659	19.5364

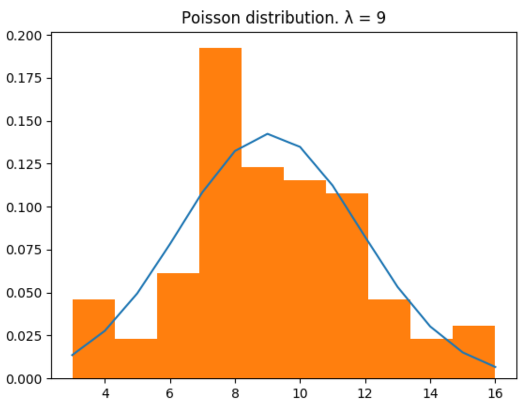


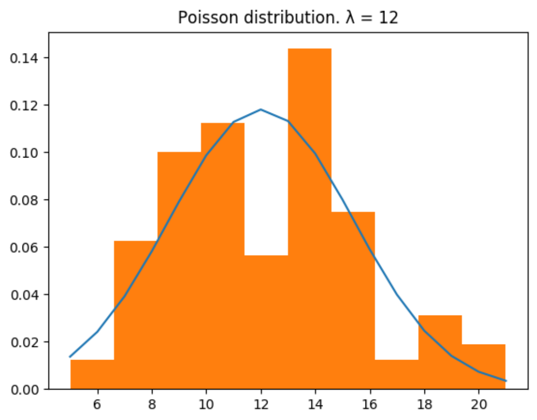


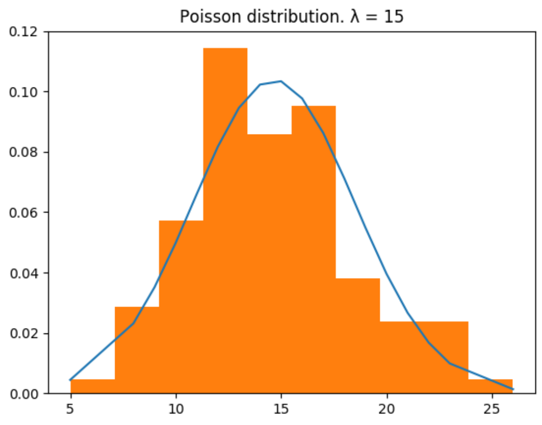












4. Листинг

```
# utils.py

A = C = 2 ** 15 + 1
M = 2 ** 32 + 1


def linear_congruential_generator(n, a=A, c=C, m=M):
    for i in range(n):
        a = (c * a) % m
        yield a / m


basic_random_values = list(linear_congruential_generator(9999))


# main.py

import np
import random
import tabulate
import matplotlib.pyplot as plt
from math import sqrt, exp
from numpy import mean, std
from scipy.stats import norm
from linear_congruential_generator import *


LAMBDAS = [6, 9, 12, 15, 18]
M = 100


def poisson(λ, size):
    for _ in range(size):
        n = 0
        alpha = basic_random_values[random.randint(0, 9000)]
```

```python
        while alpha >= exp(-λ):
            alpha *= basic_random_values[random.randint(0, 9000)]
            n += 1
        yield n


def standard_gauss_rv():
    return sum([
        basic_random_values[random.randint(1, 9000)] for _ in range(12)
    ]) - 6


def gauss(m, standard_deviation, size):
    for _ in range(size):
        yield m + standard_deviation * standard_gauss_rv()


poissons_means = []
normals_means = []
poissons_std = []
normals_std = []

for λ in LAMBDAS:
    poissons = sorted(list(poisson(λ, M)))
    normals = sorted(list(gauss(λ, sqrt(λ), M)))

    density_poisson = norm.pdf(poissons, mean(poissons), std(poissons))
    plt.plot(poissons, density_poisson)
    plt.hist(poissons, density=True)
    plt.title(f'Poisson distribution. λ = {λ}')
    plt.savefig(f'plots/poisson_{λ}')
    plt.show()
    poissons_means.append(mean(poissons))
    poissons_std.append(std(poissons)**2)

    density_normals = norm.pdf(normals, mean(normals), std(normals))
    plt.plot(normals, density_normals)
    plt.hist(normals, density=True)
    plt.title(f'Normal distribution. μ = {λ}, σ^2 = {round(sqrt(λ), 3)}')
    plt.savefig(f'plots/normal_{λ}')
    plt.show()
    normals_means.append(round(mean(normals), 2))
    normals_std.append(round(std(normals), 2)**2)

with open("output.txt", "w") as output:
    output.write(
        tabulate.tabulate(
            np.c_[LAMBDAS, poissons_means, normals_means, poissons_std, normals_std],
            headers=['λ', 'Poisson(Mean)', 'Normal(Mean)', 'Poisson(Variance)', 'Normal(Variance)']
        )
    )
```