БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Лабораторная работа №2

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

Минск 2018

Задание 1.

1. Постановка задачи

По 1000 реализаций S, вычисленных по формуле:

$$S = P(1 + n_1 i_1 + n_2 i_2 + \cdots + n_k n_k)$$

оценить числовые характеристики оценить числовые характеристики S: S_{min}, S_{max}, $E\{S\}$, $D\{S\}$, $P\{S_1 \le S \le S_2\}$ для следующих значений параметров: $k = 3, P = 15800, n = 1$ год, $S_1 = 16800, S_2 = 17000; n_1, n_2 -$ независимые СВ.

$\mathbf{n_1}$**, лет**	1/12	1/4
P	0.7	0.3

$\mathbf{n_2}$**, лет**	1/4	1/3	1/2
P	0.5	0.4	0.1

$$i_1 \in \begin{cases} R(3\%; 4.2\%), & \text{если } n_1 = 1/12 \\ R(5\%; 6.2\%), & \text{если } n_1 = 1/4 \end{cases}$$

$$i_2 \in \begin{cases} R(5\%; 6.2\%), & \text{если } n_2 = 1/4 \\ R(6\%; 7\%), & \text{если } n_2 = \frac{1}{3} \\ R(6.5\%; 7.8\%), & \text{если } n_2 = 1/12 \end{cases}$$

$$i_3 \in \begin{cases} R(5\%; 7.8\%), & \text{если } n_3 = 1/2 \\ R(6\%;\ 8\%), & \text{если } n_3 > 1/2 \end{cases}$$

2. Формулы и краткие пояснения к ним

Наращивание суммы

Формула наращения по ставке простых процентов, изменяющейся кусочно-линейно от интервала к интервалу, имеет вид:

$$S = P(1 + n_1 i_1 + n_2 i_2 + \cdots + n_k n_k)$$

где S – наращенная сумма, P – первоначальная сумма, $i_s, s = \overline{1,k}$- ставка простых процентов, определенная на интервале $(n_s, n), n = n_1 + n_2 + \cdots + n_k$.

Современное значение P можно получить из формулы:

$$P = S(1 + n_1 i_1 + n_2 i_2 + \cdots + n_k i_k)^{-1}$$

Моделирование дискретной СВ

Дискретной СВ называется ξ, имеющая дискретное распределение вероятностей, определяемое дискретным множеством значений $c_1, c_2, \dots, c_n$ $(n \le \infty)$ и заданными вероятностями значений:

$$p_i = P\{\xi = c_i\}, \qquad \sum_{i=1}^{n} p_i = 1$$

Функция распределения дискретной СВ имеет вид:

$$F(x) = P(X \le x) = \sum_{k \le x} P(x = k) \qquad (1)$$

Алгоритм моделирования ДСВ ξ, заданной распределением (1), состоит из вычисления вспомогательного вектора $q = (q_1, q_2, \dots, q_n) = (p_1, p_1 + p_2, \dots, p_1 + \cdots p_{n-1}, 1)$ и двух шагов, повторяющихся при каждом обращении к алгоритму:

1. Моделирование с помощью датчика БСВ реализации a.
2. Принятия решения о том, что реализацией ξ является x, определяемое по правилу: $x = c_i$, если $q_{i-1} \le a < q_i$, $q_0 = 0, i = \overline{1,n}$

По условию $q^1 = (0.7, 1)$, $q^2 = (0.5, 0.9, 1)$

3. Выходные данные

S_max	16956.4	16956.4	16971.2	16971.2	16971.2	16971.2
S_min	16599.5	16599.5	16599.5	16599.5	16599.5	16599.5
Mean	16807	16807.9	16806.6	16807.2	16806.8	16807.1
Variance	4464.52	4533.52	4537.39	4472.02	4503.63	4497.04
P{S1 ≤ S ≤ S2}	0.546	0.552	0.543	0.54825	0.5504	0.550833

4. Листинг

```
# utils.py

import random as r
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))


def basic_rv():
    return basic_random_values[r.randint(0, 9990)]


def continuous_uniform_rv(a, b):
    return a + (b - a) * basic_rv()


# main.py

from utils import *
from numpy import mean, var
from tabulate import tabulate


k = 3
P = 15800.0
S1 = 16800.0
S2 = 17000.0


result = []
maxs = ['S_max']
mins = ['S_min']
means = ['Mean']
vars = ['Variance']
ps = ['P{S1 <= S <= S2}']


with open('output.txt', 'w') as output:
    for _ in range(0, 6):
    for _ in range(0, 1000):
    a1 = basic_rv()# q1 = (0.7, 1.0)


if a1 < 0.7:
    n1 = 1 / 12
i1 = continuous_uniform_rv(0.03, 0.042)
else :
```

```
    n1 = 1 / 4
i1 = continuous_uniform_rv(0.05, 0.062)


a2 = basic_rv()# q2 = (0.5, 0.9, 1.0)


if a2 < 0.5:
    n2 = 1 / 4
i2 = continuous_uniform_rv(0.05, 0.062)
elif a2 < 0.9:
    n2 = 1 / 3
i2 = continuous_uniform_rv(0.06, 0.07)
else :
    n2 = 1 / 2
i2 = continuous_uniform_rv(0.065, 0.078)


n3 = 1 - (n1 + n2)


if n3 > 0.5:
    i3 = continuous_uniform_rv(0.06, 0.08)
else :
    i3 = continuous_uniform_rv(0.05, 0.078)


result.append(P * (1 + n1 * i1 + n2 * i2 + n3 * i3))


maxs.append(max(result))
mins.append(min(result))
means.append(mean(result))
vars.append(var (result))
ps.append(sum(S1 <= x <= S2
    for x in result) / len(result))


table = [maxs, mins, means, vars, ps]


output.write(tabulate(
    table,
    numalign = 'right'
))
```