БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

**Лабораторная работа №1**

Выполнил: Данилов Валентин Юрьвевич

4 курс 4 группа

Преподаватель: Кирлица В.П.

Минск 2019

**Задание 14.**

1. **Постановка задачи**

С помощью моделирования CB ξ~E(λ), получить реализации CB η, имеющей распределение Эрланга ν-го порядка γ(1/λ, ν) Положить λ =1, ν = 1,2,3.

1. **Формулы и краткие пояснения к ним**

**Генерация БСВ**

Для генерации БСВ используется *мультипликативно конгруэнтный метод*:

где *a* - множитель, *c* – приращение, *m* – модуль, *X0* – начальное значение

Получаемая последовательность зависит от выбора стартового числа и 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:

Если *Z* – основание системы счисления, которое используется в машине, *е* – разрядность, то коэффициент *а* вычисляется следующим образом:

В контекcте нашей задачи возьмем .

**Моделирование экспоненциального распределения**

Алгоритм моделирования СВ ξ ~ E(λ) основа на методе обратной функции. Обратная функция для , находится при решении уравнения относительно x:

Далее в соответствии с методом обратной функции алгоритм моделирования СВ ξ состоит из двух шагов:

1. Моделирование реализации БСВ a.

2. Вычисление в соответствии с (26) реализации x СВ ξ:

**Моделирование распределения Эрланга**

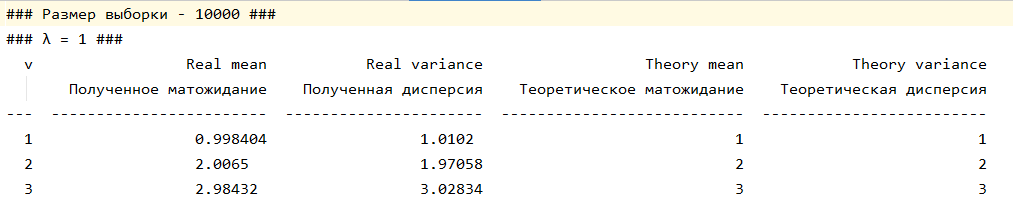
Распределение Эрланга является частным случаем гамма-распределения – для произвольного целого ν > 1 гамма-распределение называется распределением Эрланга порядка ν с параметром λ = 1/b .

Для целых значений ν ≥1 алгоритм моделирования гамма-распределения описывается следующей формулой:

Таким образом алгоритм можно описать следующим образом:

1. Генерируем *v* БСВ.
2. По указанной выше формуле вычисляем реализацию распределения Эрланга.

**3. Выходные данные**

****

**4. Исходный код программы**

**distribution.py**

**from** math **import** sqrt, erf, log, exp, factorial  
**from** random **import** random, randint  
  
A = C = 2 \*\* 15 + 1  
M = 2 \*\* 32 + 1  
  
  
**def** multiplyList(myList):  
 *# Multiply elements one by one* result = 1  
 **for** x **in** myList:  
 result = result \* x  
 **return** result  
  
  
*# BRV generator***def** linear\_congruential\_generator(n=9999, a=A, c=C, m=M):  
 **for** i **in** range(n):  
 a = (c \* a) % m  
 **yield** a / m  
  
  
basic\_random\_value = list(linear\_congruential\_generator())  
  
  
**def** next\_brv(n=9999):  
 bsv = basic\_random\_value[randint(0, n)]  
 **while** bsv >= 1:  
 bsv = basic\_random\_value[randint(0, n)]  
 **return** bsv  
  
  
**def** erlang(\_lambda, v):  
 **if** v == 1:  
 **return** exponential(1 / \_lambda)  
  
 a = [next\_brv() **for** \_ **in** range(v)]  
  
 **return** -log(multiplyList(a)) \* (1 / \_lambda)  
  
  
**def** erlang\_mean(\_lambda, v):  
 **return** v / \_lambda  
  
  
**def** erlang\_variance(\_lambda, v):  
 **return** v / (\_lambda \*\* 2)  
  
  
**def** exponential(\_lambda):  
 a = next\_brv()  
 **return** (-1 / \_lambda) \* log(a)

**lab1.py**

**from** distribution **import** \*  
**from** utils **import** \*  
**import** numpy **as** np  
  
*# parameters*v = [1, 2, 3]  
lambdas = [1]  
n = 15  
N = 10000  
  
erlang\_theory\_means = []  
erlang\_theory\_variances = []  
erlang\_real\_means = []  
erlang\_real\_variances = []  
  
**for** \_v **in** v:  
 **for** \_lambda **in** lambdas:  
 \_selection = np.array([erlang(\_lambda, \_v) **for** \_ **in** range(N)])  
 \_mean = np.mean(\_selection)  
 \_variance = np.var(\_selection)  
 erlang\_real\_means.append(\_mean)  
 erlang\_real\_variances.append(\_variance)  
 erlang\_theory\_means.append(erlang\_mean(\_lambda, \_v))  
 erlang\_theory\_variances.append(erlang\_variance(\_lambda, \_v))  
  
headers = [**"v"**, **"Real mean\nПолученное матожидание"**, **"Real variance\nПолученная дисперсия"**,  
 **"Theory mean\nТеоретическое матожидание"**, **"Theory variance\nТеоретическая дисперсия"**]  
rows = np.c\_[v, erlang\_real\_means, erlang\_real\_variances, erlang\_theory\_means, erlang\_theory\_variances]  
report\_erlang(**"output1.txt"**, headers, rows, N, lambdas[0])

**utils.py**

**from** tabulate **import** tabulate  
**import** io  
  
HEADING\_TEMPLATE = **"### Размер выборки - {} ###\n### λ = {} ###\n"  
  
  
def** report\_erlang(filename, headers, values, selection\_size, \_lambda):  
 **with** io.open(filename, **"w"**, encoding=**"utf-8"**) **as** file:  
 file.write(HEADING\_TEMPLATE.format(selection\_size, \_lambda))  
 file.write(  
 tabulate(  
 values, headers  
 )  
 )