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

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

По дисциплине «Моделирование»

Выполнил:
Студент гр. ИВ-622
Ольман Р.Ю.

Проверила:
Ассистент Кафедры ВС
Петухова Я. В.

Новосибирск 2020

Оглавление

Постановка задачи

Генерация независимых, одинаково распределенных случайных величин:

- Непрерывное распределение с помощью метода отбраковки;
- Дискретное распределение с возвратом;
- Дискретное распределение без возврата.

Теоретические сведения

Метод отбраковки

Законы распределения вероятности могут быть заданы различными функциями. Поэтому надо уметь превращать равномерный ГСЧ в такой генератор случайных чисел, который задан произвольным законом распределения. Один из методов для моделирования непрерывной случайной величины является метод отбраковки. Суть метода заключается в том, что функция плотности распределения вероятностей случайных величин $f_\eta(x)$ вписывается в прямоугольник $(a,b)\ \times(0,c)$, такой что a и b соответствует границам диапазона изменения случайных величин η, а c – максимальному значению функции плотности её распределения. Тогда очередная реализация случайных величин определяется по следующему алгоритму:

- Выбрать функцию плотности распределения случайных величин
- Получить два независимых случайных числа ξ_1 и ξ_2
- И если $f_\eta\,(a+(b-a)*\ \xi_1)>c*\ \xi_2$ то выдать $(a+(b-a)*\ \xi_1)$ в качестве результата , иначе повторить шаг 2.

Дискретное распределение

Дискретные распределения используются для описания событий с не дифференцируемыми характеристиками, определёнными в изолированных точках. Проще говоря, для событий, исход которых может быть отнесён к некоторой дискретной категории: успех или неудача, целое число (например, игра в рулетку, в кости), орёл или решка и т.д.

Описывается дискретное распределение вероятностью наступления каждого из возможных исходов события. Как и для любого распределения (в том числе непрерывного) для дискретных событий определены понятия математическое ожидания и дисперсии. Однако, следует понимать, что математическое ожидание для дискретного случайного события — величина в общем случае нереализуемая как исход одиночного случайного события, а скорее, как величина, к которой будет стремиться среднее арифметическое исходов событий при увеличении их количества. В моделировании дискретных случайных событий важную роль играет комбинаторика, так как вероятность исхода события можно определить, как отношение количества комбинаций, дающих требуемый исход к общему количеству комбинаций.

Дискретное распределение с возвратом

Случайная величина X называется дискретной, если она может принимать дискретное множество значений ($x1, x2, x3, \ldots, xn$). Дискретная случайная величина описывается с помощью таблицы (распределение случайной величины X) или в виде аналитической формулы.

$$X = \begin{pmatrix} x_1 \; x_2 \; x_3 \dots x_n \\ p_1 \; p_2 \; p_3 \cdots p_n \end{pmatrix}$$,где

$(x_1, x_2, x_3, \ldots, x_n)$ – возможные значения величины X

$(p_1, p_2, p_3, \ldots, p_n)$ – соответствующие им вероятности:

$$P(X = x_i) = p_i$$

Числа могут быть любыми, а вероятности должны удовлетворять двум условиям:

$$p_i > 0 \text{ и } \sum_{i=1}^{n} p_i = 1$$

Функция распределения принимает такой вид:

$$F(x) = \sum_{x_k < x} P(X = x_k)$$

Наиболее общий вид построения генератора дискретных случайных величин основывается на следующем алгоритме. Пусть имеется таблица пара (xi, pi). Тогда сумму можно представить интервалом [0,1), разбитым на полуинтервалы $[\nu_{i-1}, \nu_i), \nu_0 = 0, \nu_n = 1$ длины p_i. Случайная величина X с неизбежностью принадлежит одному из этих полуинтервалов, тем самым определяя индекс дискретного значения. Номера полуинтервала определяется как:

$$min\{i|X < \nu_i\} = min\, \{i\, |X < \sum_{j=1}^{i} p_i\}$$

Дискретное распределение без возврата

Есть n случайных величин с одинаковой вероятностью (при следующих выборках вероятность распределяется поровну между величинами), мы выбираем $3 * \frac{n}{4}$ следующих величин без повторений, проделываем это большое количество раз и считаем частоты этих значений. Универсальный алгоритм для выборки без повторения. Входные данные: n – число индексов; m – длина выборки. Выходные данные: $L[1..m]$ – результирующий вектор индексов.

Ход работы

Для дальнейшей работы нужно вычислить функцию плотности:

$$f(x) = \begin{cases} 0, \quad x \le 0 \\ a * (x+7), 0 < x \le 3 \\ 0, \quad x > 3 \end{cases}$$

По условию нормировки:

$$\int_{-\infty}^{\infty} f(x)dx = 1$$

Тогда:

$$\int_{-\infty}^{0} 0dx + \int_{0}^{2} a * (x+7)\,dx + \int_{2}^{\infty} 0dx = a * \ (x+7) \ |\begin{matrix} 2 \\ 0 \end{matrix} = 1$$

$$f(x) = \begin{cases} 0, \quad x \le 0 \\ 0.25 * \ (x+7), 0 < x \le 3 \\ 0, \quad x > 3 \end{cases}$$

Исходя из найденной плотности распределения мы можем построить график:

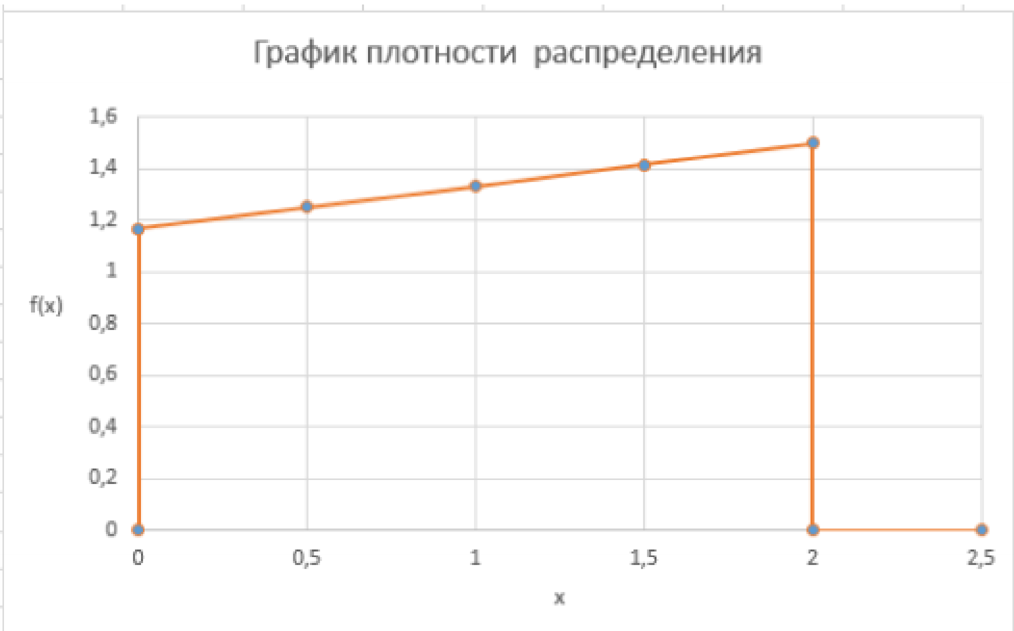

Рисунок 1 – График плотности

Рисунок 2 – Результат работы метода отбраковки

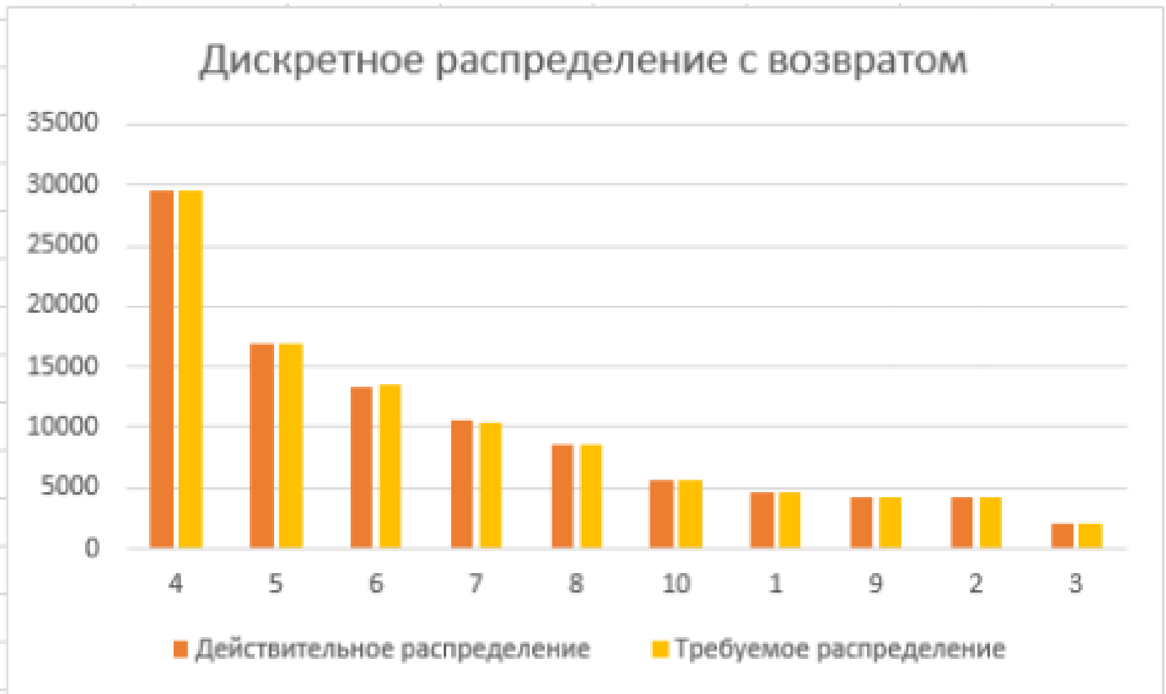

Рисунок 3 – Дискретное распределение с возвратом

Рисунок 4 – Дискретное распределение без возврата

Заключение

В данной лабораторной работе были использованы такие методы моделирования случайных величин как:

- Метод отбраковки
- Дискретное распределение без возврата
- Дискретное распределение с возвратом

Роль ГСЧ выполнял стандартный модуль random() из библиотеки Python. Исходя из результатов, отображенных на рисунках, видно, что исходную функцию плотности метод смог смоделировать достаточно точно, но недостатки все-таки есть:

- Тратится очень много времени на выполнение работы алгоритма. Все это из-за того, что время так же отводится и на генерацию точек, которые не попали на кривую распределения плотности.
- Так же, в силу того, что для распределения с длинными «хвостами» у данного алгоритма слабая эффективность и в этом случае повышается частота повторных испытаний он применим только для аналитических функций.

В итоге на основе наших результатов можно сделать следующий вывод:

ГСЧ языка программирования Python показывает очень близкое распределение к равномерному так как действительное распределение сильно приближено к требуемому распределению.

Листинг

lab_1.py

```
import random
import typing as tp

N = 1000000
K = 100
a_ = 0.25


def f(x: float) -> float:
    if x > 0 and x <= 2:
        return a_ * ((x + 3) / 2)
    else:
        return 0


def method_rejections() -> float:
    array = []
    a = 0
    b = 2
    c = 1
    while True:
        x1 = random.uniform(0, 1)
        x2 = random.uniform(0, 1)
        tmp = f(a + (b - a) * x1)
        if tmp > c * x2:
            return a + (b - a) * x1


def distribution_density() -> tp.List[int]:
    hit = [0] * K
    x = []
    y_1 = []
    m = 2 / K
    f = open('density.txt', 'w')
    for i in range(N):
        rand = method_rejections()
        x.append(rand)
        y = a_ * (x[i] + 5)
        y_1.append(y)
        inter = rand / m
        hit[int(inter)] += 1
        f.write(str(x[i]) + '\t' + str(y) + '\n')
    f.close()
    return hit


def repeat() -> tp.List[int]:
    residue_chance = 1.0
    chance_hit = [0] * K
    hit = [0] * K

    for i in range(K - 1):
        chance_hit[i] = random.uniform(0, 1) % residue_chance
        residue_chance -= chance_hit[i]
```

```
    chance_hit[-1] = residue_chance

    for i in range(N):
        chance = random.uniform(0, 1)
        sum = 0.0
        for j in range(K):
            sum += chance_hit[j]
            if (chance < sum):
                hit[j] += 1
                break

    for i in range(K):
        chance_hit[i] = chance_hit[i] * 100000
    f = open('repeat.txt', 'w')
    i = 0
    while i != K:
        f.write(str(hit[i]) + '\t' + str(chance_hit[i]) + '\n')
        i += 1
    f.close()
    return hit, chance_hit


def no_repeat() -> tp.List[int]:
    k = 3 / 4 * K
    hit = [0] * K
    part = N / k + 1

    for i in range(int(part)):
        nums = [nums for nums in range(K)]
        if i == part - 1:
            k = N % k
        for j in range(int(k)):
            p = 1.0 / (K - j)
            it = int(random.uniform(0, 1) * 1.0 / p)
            hit[nums[it]] += 1
            nums.pop(it)
    f = open('norepeat.txt', 'w')
    i = 0
    while i != K:
        f.write(str(hit[i]) + '\t' + str(N / K) + '\n')
        i += 1
    f.close()
    return hit


distribution_density()
repeat()
no_repeat()
```