Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе № 1

По курсу: Моделирование

На тему: Изучение функции распределения и функции плотности распределения случайной величины

Студент:

Андреев Александр Алексеевич

Группа: ИУ7-74Б

Преподователь:

Рудаков Игорь Владимирович

Москва, 2022 г.

Содержание

1 Задание

Реализовать программу для построения графиков функции и плотности для следующих распределений:

- равномерное распределение;
- распределение Эрланга (вариант 4).

2 Теоритическая часть

2.1 Равномерное распределение

Непрерывное равномерное распределение - распределение случайной вещественной величины, принимающей значения, принадлежащие некоторому промежутку конечной длины, характеризующееся тем, что плотность вероятности на этом промежутке почти всюду постоянна.

Плотность распределения представлена в формуле 1.

$$f_X(x) = \begin{cases} \frac{1}{b-a}, x \in [a, b] \\ 0, x \notin [a, b] \end{cases} \tag{1}$$

Функция распределения представлена в формуле 2.

$$F_X(x) = \begin{cases} 0, x < a \\ \frac{x-a}{b-a}, a \le x < b \\ 1, x \ge b \end{cases} \tag{2}$$

2.2 Распредление Эрланга

Распределение Эрланга – это гамма-распределение Γ(x | a, b) с параметрома, принимающим лишь целые значения. Здесь оно приводится лишь из-за того, что часто встречается в инженерных приложениях, особенно телефонии.

Плотность распределения n-го порядка представлена в формуле 3.

$$p(x) = \lambda \frac{(\lambda x)^{n-1}}{!(n-1)} e^{-\lambda x}, x >= 0 \tag{3}$$

Функция распределения представлена в формуле 4.

$$F(x) = \int_0^x \lambda \frac{(\lambda t)^{n-1}}{!(n-1)} e^{-\lambda t} dt \quad (4)$$

3 Результаты

3.1 Равномерное распределение

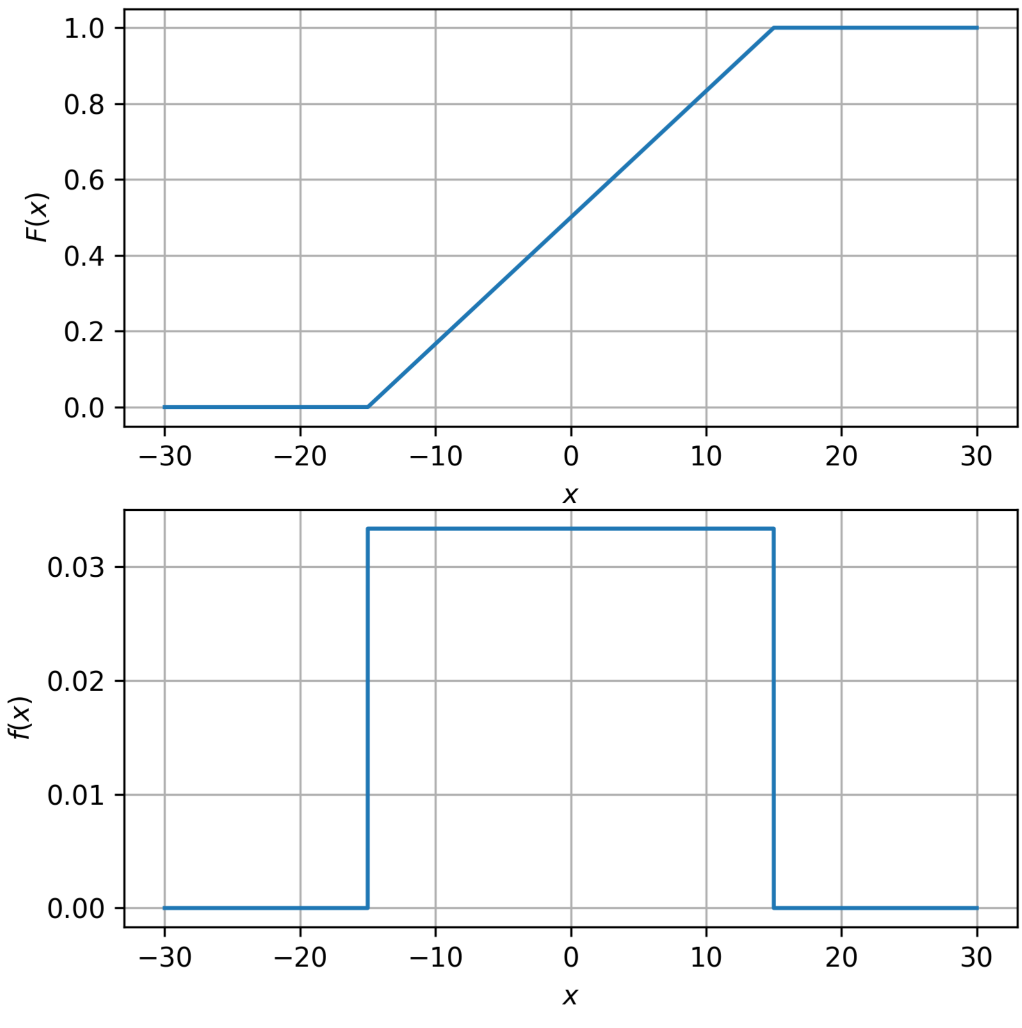

Рис. 1: Равномерное распределение при a = -15, b = 15

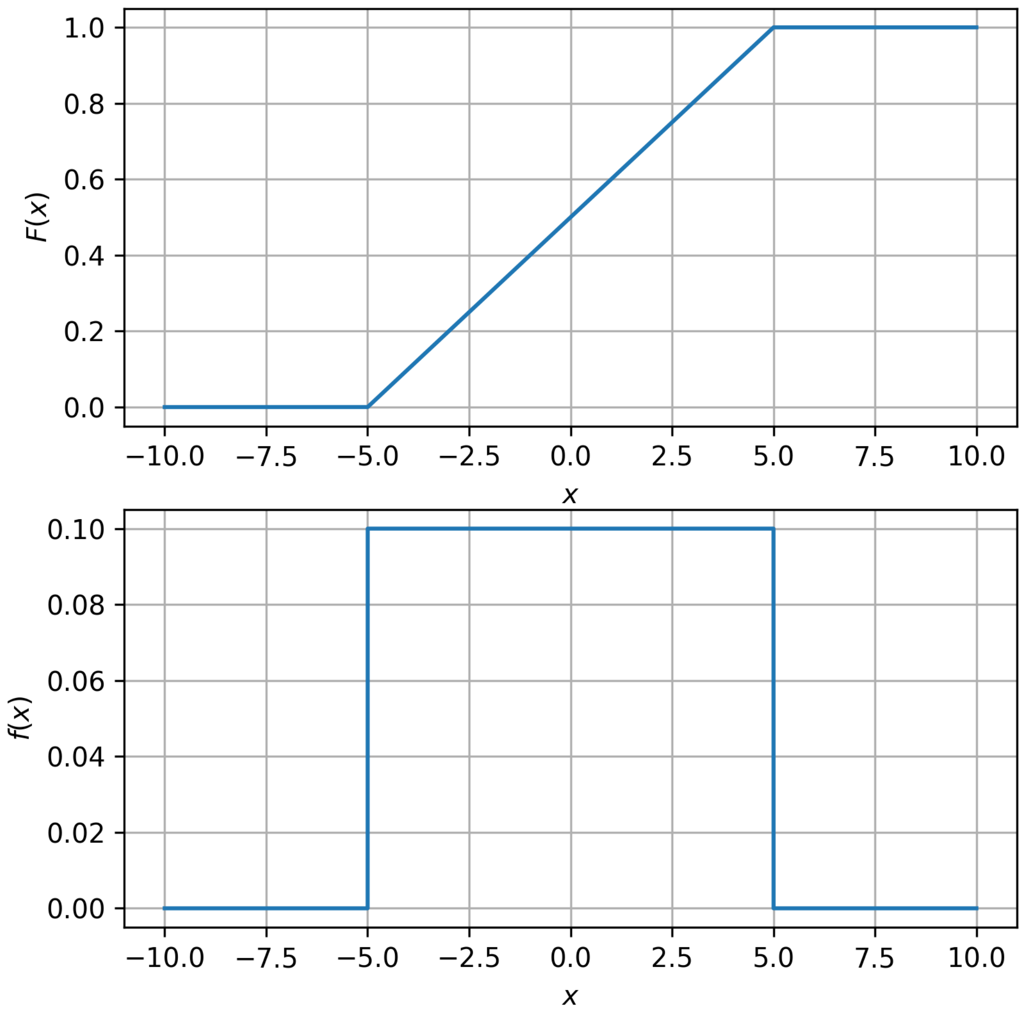

Рис. 2: Равномерное распределение при a = -5, b = 5

3.2 Распределение Эрланга

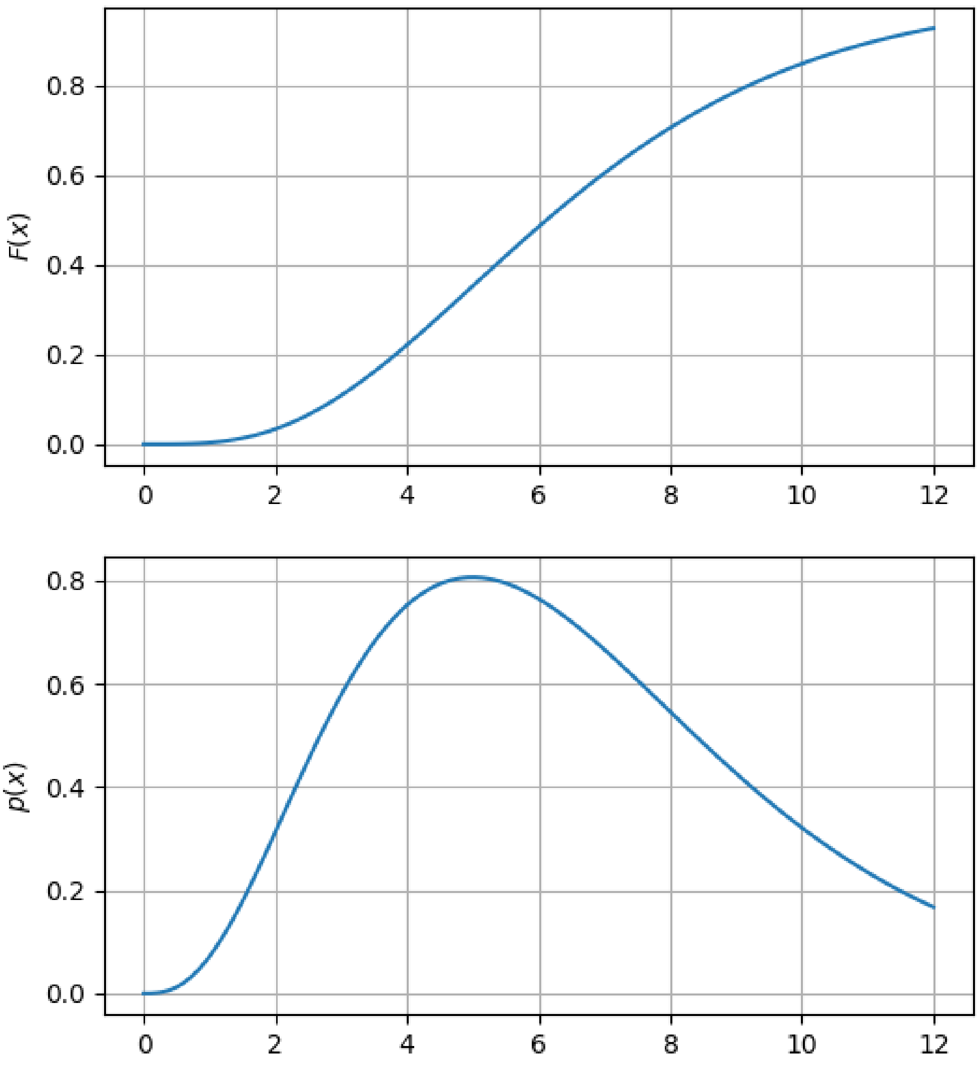

Рис. 3: Распределение Эрланга при k = 4, $\lambda = 0.6$

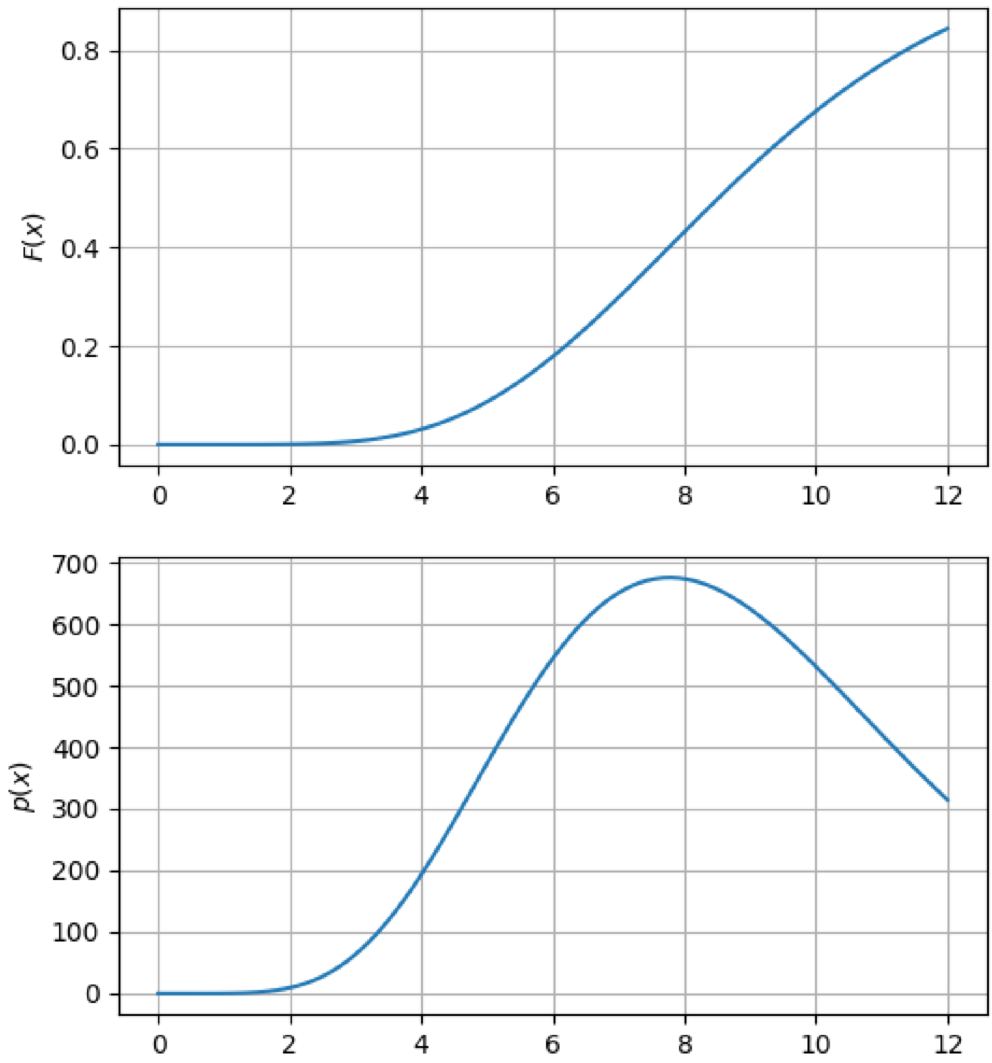

Рис. 4: Распределение Эрланга при k = 8, $\lambda = 0.9$

4 Листинг кода

```
from cmath import exp


class ErlangDistribution:
    def factorial(self, n):
```

```
        factorial_ans = 1
        for i in range(1, n + 1):
            factorial_ans = factorial_ans * i
        return factorial_ans

    def distributionFunction(self, x, k, lambda_v):
        summ = 0

        for i in range(0, k):
            summ += 1 / self.factorial(i) * exp(-lambda_v * x) * pow((lambda_v *

        return 1 - summ

    def densityFunction(self, x, k, lambda_v):
        return pow(lambda_v, k) * pow(x, k - 1) * exp(-lambda_v * x)
```

Листинг 1: сущность метода Эрланга

```
import numpy as np
import matplotlib.pyplot as plt

import erlangdistribution


def main():
    a = float(input("Enter start point a: "))
    b = float(input("Enter end point b: "))
    k = int(input("Enter k: "))
    lambda_v = float(input("Enter lambda: "))

    delta = b - a

    x_uniform = np.arange(a - delta / 2, b + delta / 2, 0.001)

    y_uniform_cdf = [uniform_distribution_cdf(a, b, value) for value in x_uniform
    y_uniform_pdf = [uniform_distribution_pdf(a, b, value) for value in x_uniform

    draw_plots(x_uniform, y_uniform_cdf, y_uniform_pdf, k, lambda_v)


```

```python
def draw_plots(x, y_cdf, y_pdf, k, lambda_v):
    fig, axs = plt.subplots(2, figsize=(6, 7))

    axs[0].plot(x, y_cdf)
    axs[1].plot(x, y_pdf)

    axs[0].set_xlabel('$x$')
    axs[0].set_ylabel('$F(x)$')

    axs[1].set_xlabel('$x$')
    axs[1].set_ylabel('$f(x)$')

    axs[0].grid(True)
    axs[1].grid(True)

    plt.show()

    erlang_f(k, lambda_v)


def uniform_distribution_cdf(a, b, x):
    return (x - a) / (b - a) if (a <= x < b) else 0 if x < a else 1


def uniform_distribution_pdf(a, b, x):
    return 1 / (b - a) if (a <= x <= b) else 0


def normal_distribution_cdf(x, mu, sigma):
    return norm.cdf(x, mu, sqrt(sigma))


def normal_distribution_pdf(x, mu, sigma):
    return norm.pdf(x, mu, sqrt(sigma))

def erlang_f(k, lambda_v):
    fig, axs = plt.subplots(2, figsize=(6, 7))

    dist_1 = []
```

```
    dist_2 = []

    a = 0
    b = 12
    step = (b - a) / 100
    #
    # k = 8
    # lambda_v = 0.9

    steps = []

    while a < b:
        steps.append(a)
        dist_1.append(erlangdistribution.ErlangDistribution().distributionFunctio
            a, k, lambda_v
        ))
        dist_2.append(erlangdistribution.ErlangDistribution().densityFunction(
            a, k, lambda_v
        ))
        a += step

    axs[0].plot(steps, dist_1)
    axs[1].plot(steps, dist_2)

    axs[0].set_ylabel('$F(x)$')
    axs[1].set_ylabel('$p(x)$')

    axs[0].grid(True)
    axs[1].grid(True)

    plt.show()


if __name__ == '__main__':
    main()
```

Листинг 2: программная реализация равномерного распределения и распределения Эрланга