МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 52

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Доцент, к.т.н., доцент		Н.В. Марковская
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

ИССЛЕДОВАНИЕ ИНТЕНСИВНОСТИ ОТКАЗОВ ДЛЯ НЕВОССТАНАВЛИВАЕМЫХ СИСТЕМ

по курсу: НАДЕЖНОСТЬ ИНФОКОММУНИКАЦИОННЫХ СИСТЕМ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №	5911		А.А. Бенцлер
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2022

Цель работы

Необходимо реализовать программу имитационного моделирования процесса функционирования невосстанавливаемой системы для всех периодов жизни системы и построить зависимость надежности и интенсивности отказов от времени.

Теоретический материал:

Основные показатели надежности невосстанавливаемых объектов

В качестве основной исследуемой величины будем использовать наработку до первого отказа (другими словами, время безотказной работы), которую будем обозначать как τ. Величина τ является непрерывной случайной величиной, и в качестве ее показателей надежности, в принципе, можно использовать вероятностные характеристики, такие как функция распределения $F(t)$, плотность распределения $f(t)$ и математическое ожидание $M[\tau]$.

Основные показатели надежности невосстанавливаемых объектов:

- Функция надежности $R(t)$;
- Интенсивность отказов $\lambda(t)$;
- Среднее время безотказной работы $\overline{T}$.

функция надежности обладает следующими свойствами:

- Функция надежности - невозрастающая функция;
- Принимает только неотрицательные значения;
- Определена на интервале [0, ∞);
- $R(0) = 1$;
- Безразмерная функция.

Интенсивность отказов $\lambda(t)$, которая является локальной по времени характеристикой, определяет условную плотность вероятности отказа объекта в момент времени, следующий непосредственно за моментом t. Чем больше значение этого показателя, тем больше вероятность того, что объект, проработавший до момента t, вскоре откажет.

Функция интенсивности отказов подчиняется только естественным условиям:

- $\lambda(t) \geq 0$;
- $t \in [0, \infty)$.

Необходимо вывести графики функции интенсивности отказов $\lambda(t) = \frac{-R'(t)}{R(t)}$.

Для моделирования эти функции рассчитываются по формулам

$$R(t) = \frac{n_t}{n},$$

$$\lambda(t) = \frac{n_t - n_{t+\Delta t}}{n_t} * \frac{1}{\Delta},$$

где $\Delta t \ = \ 0{,}1 \ \cdot \ tstep, nt$ и $nt + \Delta t$ – это число систем, остающихся работоспособными в момент времени t и $t \ + \ \Delta$ соответственно.

Три периода жизни объекта

Весь период эксплуатации объекта, начиная от ввода и заканчивая достижением предельного состояния, можно условно разделить на три периода, как это показано на рисунке:

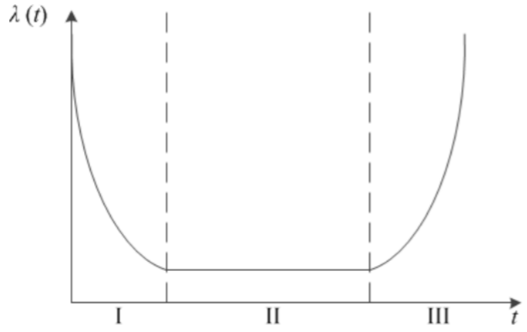

Рисунок 1 – Периоды жизни объекта

Период приработки

При моделировании первого периода - приработки - системы делятся на две группы с заданной вероятностью. После этого каждый элемент каждой группы принимается как отдельная система, для которой генерируется период жизни

$$T = \frac{-\ln(u)}{\lambda_k},$$

u – случайная величина, распределенная по равномерному закону

λ_k – интенсивность k-ой группы

Теоретическая формула функции надежности для первого периода выглядит следующим образом:

$$R(t) = \sum_{i=1}^{k} R_i(t) \cdot p_i;$$

$$R(t)_{k=2} = e^{-\lambda_1 t} p_1 + e^{-\lambda_2 t} p_2,$$

k – количество групп систем

$R_i(t)$ – функция надежности для i-той группы

p_i – вероятность того, что система принадлежит i-той группе

Функция интенсивности отказов в этом случае принимает вид

$$\lambda(t)_{k=2} = \frac{-R'(t)}{R(t)} = \frac{-\lambda_1 e^{-\lambda_1 t} p_1 - \lambda_2 e^{-\lambda_2 t} p_2}{e^{-\lambda_1 t} p_1 + e^{-\lambda_2 t} p_2}.$$

Период нормального функционирования

Период нормального функционирования предполагает моделирование системы из последовательно соединенных элементов из разных групп. Период жизни системы из двух элементов определяется по формуле

$$T = \min(T_1, T_2),$$

Теоретическая формула функции надежности для второго периода выглядит следующим образом:

$$R(t) = \prod_{i=1}^{k} R_i(t)\,;$$

$$R(t)_{k=2} = e^{-\lambda_1 t} \cdot e^{-\lambda_2 t} = e^{-(\lambda_1+\lambda_2)t},$$

k – количество групп систем

$R_i(t)$ – функция надежности для i-той группы

Функция интенсивности отказов в этом случае принимает вид

$$\lambda(t)_{k=2} = \frac{(\lambda_1 + \lambda_2) e^{-(\lambda_1+\lambda_2)t}}{e^{-(\lambda_1+\lambda_2)t}} = (\lambda_1 + \lambda_2).$$

Период старения

Период жизни системы при моделировании периода старения определяется, как

$$T = \max(T_1, T_2),$$

и представляет из себя систему из параллельно соединенных элементов.

Теоретическая формула функция надежности для третьего периода:

$$R(t) = 1 - \prod_{i=1}^{k} \left(1 - R_i(t)\right);$$

$$R(t)_{k=2} = 1 - \left(1 - R_1(t)\right)\left(1 - R_2(t)\right) = R_1(t) + R_2(t) - R_1(t)R_2(t) =$$

$$= e^{-\lambda_1 t} + e^{-\lambda_2 t} - e^{-(\lambda_1+\lambda_2)t}.$$

Теоретическая формула функции интенсивности отказов:

$$\lambda(t)_{k=2} = \frac{\lambda_1 e^{-\lambda_1 t} + \lambda_2 e^{-\lambda_2 t} - (\lambda_1 + \lambda_2)e^{-(\lambda_1+\lambda_2)t}}{e^{-\lambda_1 t} + e^{-\lambda_2 t} - e^{-(\lambda_1+\lambda_2)t}}.$$

Ход работы:

Графики, полученные в результате выполнения программы:

$k = 2, n = 50000, \lambda_1 = 0.8,\ \lambda_2 = 0.9.$

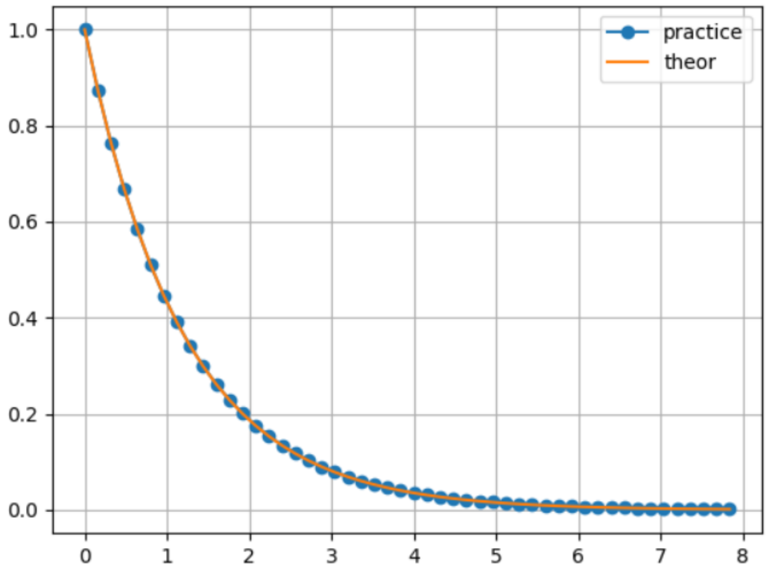

Рис.2 – Теоретическое и практическое значения функции надежности системы для периода приработки

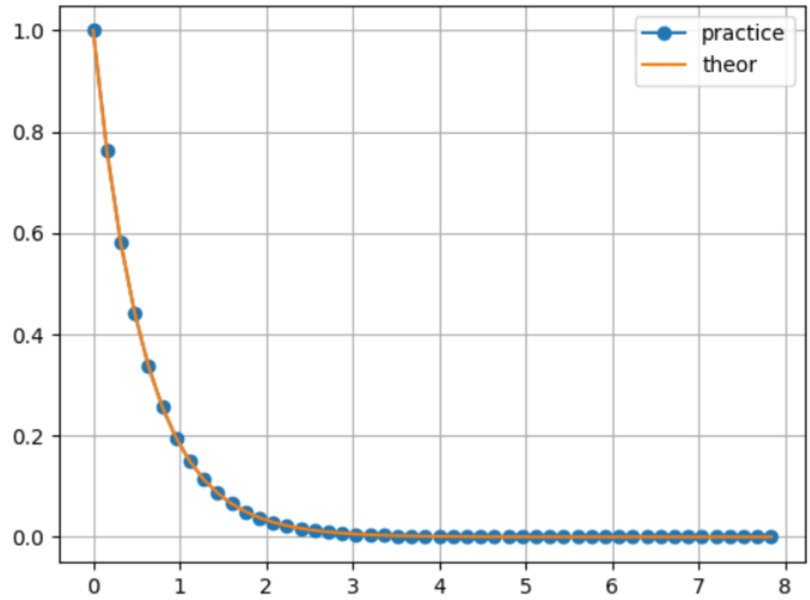

Рис.3 – Теоретическое и практическое значения функции надежности системы для периода нормального функционирования системы

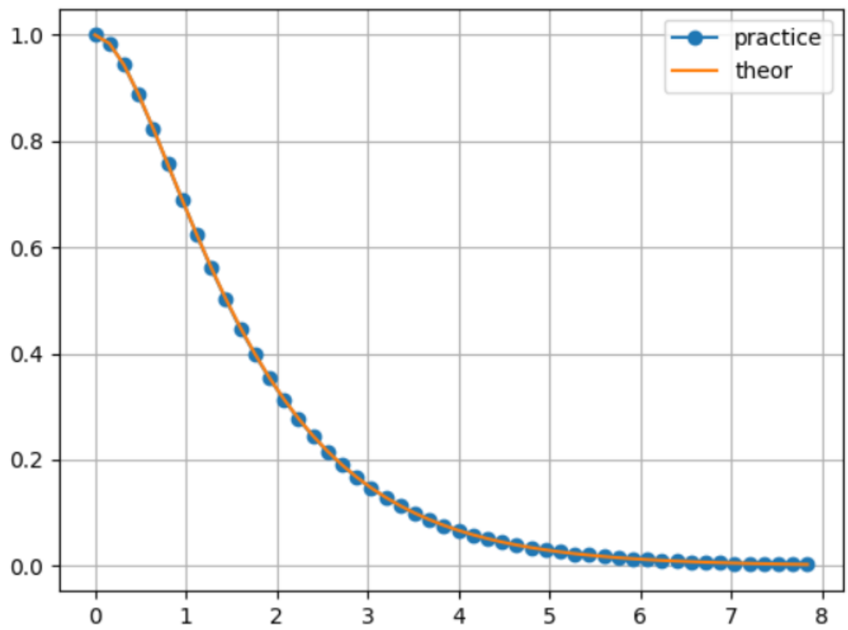

Рис.4 – Теоретическое и практическое значения функции надежности системы для периода старения системы

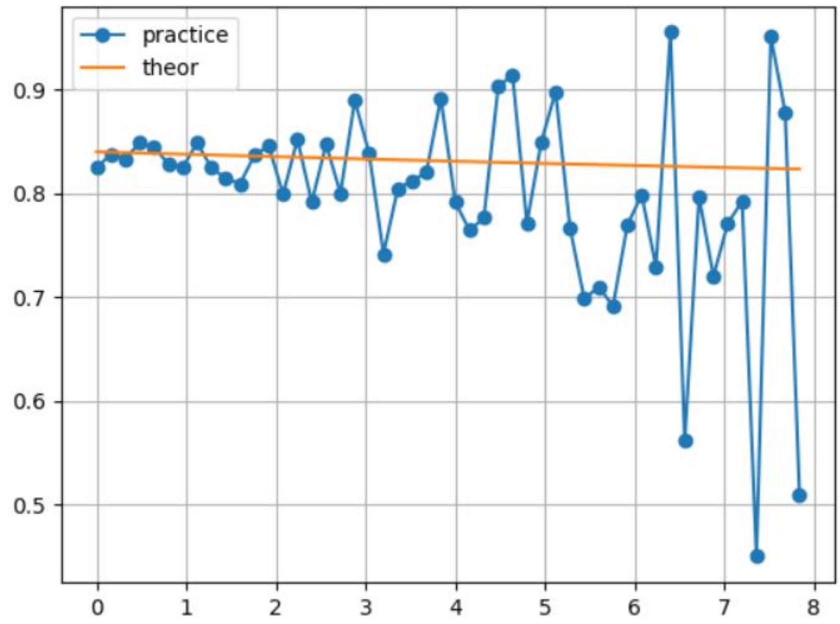

Рис.5 – Теоретическое и практическое значения функции интенсивности отказов для периода приработки

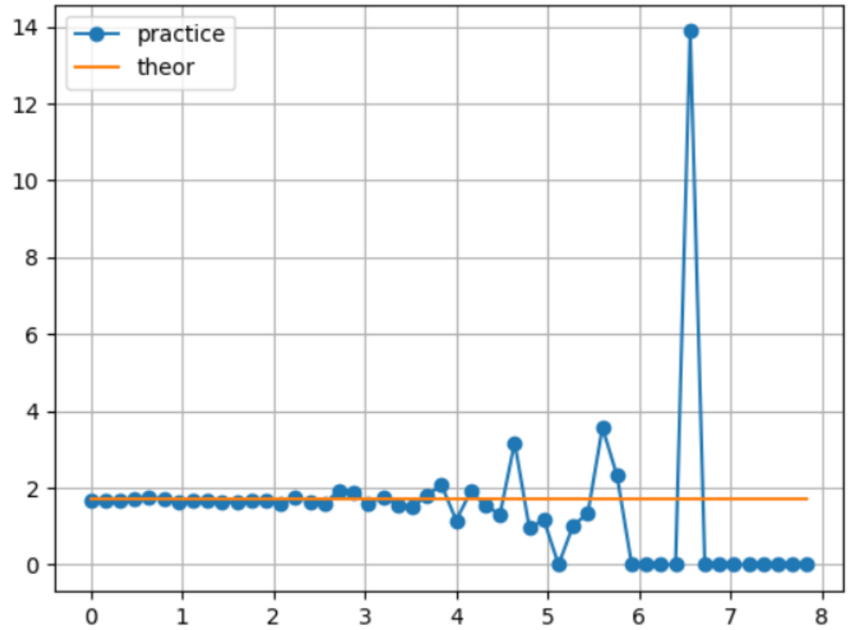

Рис.6 – Теоретическое и практическое значения функции интенсивности отказов для периода нормального функционирования системы

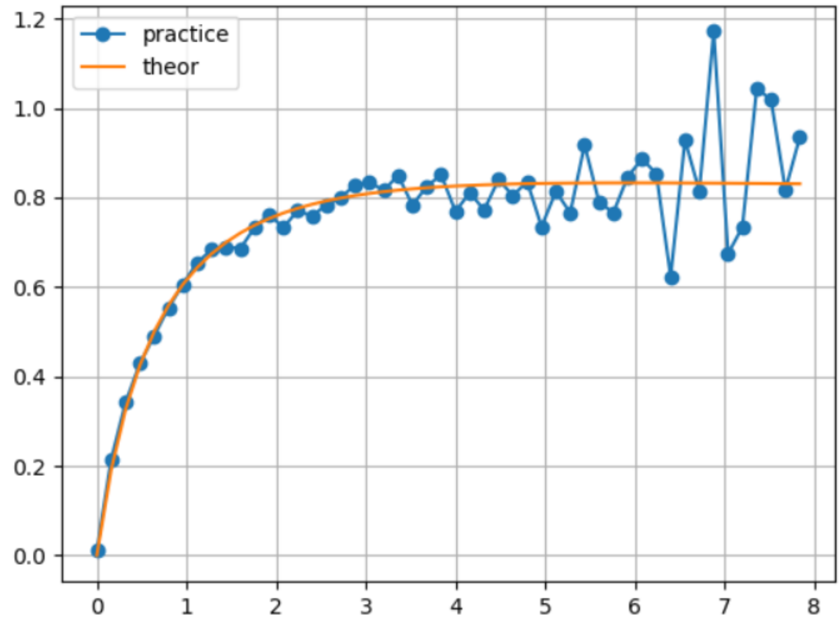

Рис.7 – Теоретическое и практическое значения функции интенсивности отказов для периода старения системы

Вывод

В ходе лабораторной работы было выполнено имитационное моделирование процесса функционирования невосстанавливаемой системы для всех периодов жизни системы и построены графики зависимости надёжности и интенсивности отказов от времени. В результате анализа полученных результатов можно сделать следующие выводы:

1. Функция надежности системы является убывающей для всех трех периодов жизни. Практические значения сошлись с теоретическими.
2. Функция интенсивности отказов для периода приработки является убывающей, для второго периода она является постоянной величиной, а в периоде старения интенсивность отказов - возрастающая функция.
3. Практические значения функции интенсивности отказов отклоняются от теоретических, однако, повторяют их траекторию с некоторой точностью, следовательно, моделирование выполнено верно.

Листинг

```
def run_in_period(time, lambda_fail, N, p, del_t):
   tau = []
   lambda_p = []
   tmp = N
   R = []
   R_theor = []
   R_diff = []
   lambda_p_theor = []
   for i in time:
      R_theor.append(float(math.e ** -(lambda_fail[0] * i) * p + math.e ** -
(lambda_fail[1] * i) * (1 - p)))
      R_diff.append(-lambda_fail[0] * p * math.e ** -(lambda_fail[0] * i) -
(1 - p) * lambda_fail[1] * math.e ** (-lambda_fail[1] * i))
   for i in range(len(R_theor)):
      lambda_p_theor.append(float(R_diff[i] / R_theor[i]) * -1)
   for i in range(N):
      value = random.random()
      if value < p:
         tau.append(-math.log(random.random()) / lambda_fail[0])
      else:
         tau.append(-math.log(random.random()) / lambda_fail[1])
   for i in range(len(time) ):
      count1 = 0
      count2 = 0
      for j in range(len(tau)):
         if tau[j] > time[i]:
            count1 += 1
         if tau[j] < time[i] + del_t and tau[j] > time[i]:
            count2 += 1
      R.append(count1 / N)
      lambda_p.append(((count1 - (count1 - count2)) / count1 * (1 / del_t)))
   return lambda_p, R, R_theor, lambda_p_theor

def normal_operation_period(time, lambda_fail, N, p, del_t, k):
   tau = []
   lambda_p = []
   R = []
   R_theor = []
   R_diff = []
   lambda_p_theor = []
   for i in time:
      R_theor.append(math.e ** -(lambda_fail[0] * i) * (math.e ** -
(lambda_fail[1] * i)))
      R_diff.append(-(lambda_fail[0] + lambda_fail[1]) * math.e ** (-
(lambda_fail[0] + lambda_fail[1]) * i))
   for i in range(len(R_theor)):
      lambda_p_theor.append(float(R_diff[i] / R_theor[i]) * -1)
   for i in range(N):
      tmp = []
      for j in range(k):
         tmp.append(-math.log(random.random()) / lambda_fail[j])
      tau.append(min(tmp))
   for i in range(len(time)):
      count1 = 0
      count2 = 0
      for j in range(len(tau)):
         if tau[j] > time[i]:
            count1 += 1
         if tau[j] < time[i] + del_t and tau[j] > time[i]:
            count2 += 1
      R.append(count1 / N)
```

```
        lambda_p.append(((count1 - (count1 - count2)) / count1 * (1 / del_t)))
    return lambda_p, R, R_theor, lambda_p_theor

def aging_period(time, lambda_fail, N, p, del_t, k):
    tau = []
    lambda_p = []
    R = []
    R_theor = []
    R_diff = []
    lambda_p_theor = []
    for i in time:
        R_theor.append(math.e ** -(lambda_fail[0] * i) + (math.e ** -
(lambda_fail[1] * i)) - ((math.e ** -(lambda_fail[0] * i)) * (math.e ** -
(lambda_fail[1] * i))))
        R_diff.append((lambda_fail[0] + lambda_fail[1]) * math.e ** -
((lambda_fail[0] + lambda_fail[1]) * i) - lambda_fail[1] * math.e ** -
(lambda_fail[1] * i) - lambda_fail[0] * math.e ** -(lambda_fail[0] * i))
    for i in range(len(R_theor)):
        lambda_p_theor.append(float(R_diff[i] / R_theor[i]) * -1)
    for i in range(N):
        tmp = []
        for j in range(k):
            tmp.append(-math.log(random.random()) / lambda_fail[j])
        tau.append(max(tmp))
    for i in range(len(time)):
        count1 = 0
        count2 = 0
        for j in range(len(tau)):
            if tau[j] > time[i]:
                count1 += 1
            if tau[j] < time[i] + del_t and tau[j] > time[i]:
                count2 += 1
        R.append(count1 / N)
        lambda_p.append(((count1 - (count1 - count2)) / count1 * (1 / del_t)))
    return lambda_p, R, R_theor, lambda_p_theor

def Plot(time, practice, theory, label1, label2):
    fig, ax = plt.subplots()
    ax.plot(time, practice,marker = 'o', label = label1)
    ax.plot(time, theory, label = label2)
    plt.legend()
    plt.grid(True)

if __name__ == "__main__":
    N = 500000
    k = 2
    p = 0.6
    lambda_fail = [0.8, 0.9]
    lambda_fail_theor = 0.5
    nfig = 1
    t = 8
    t_step = t / 50
    del_t = 0.1 * t_step
    time = np.arange(0, t, t_step)
    lambda_p = []
    R_theor = []
    R_pract = []
    lambda_p_theor = []
    lambda_p, R_pract, R_theor, lambda_p_theor = run_in_period(time,
lambda_fail, N, p, del_t)
    Plot(time, lambda_p, lambda_p_theor, "practice", "theor")
    Plot(time, R_pract, R_theor, "practice", "theor")
    lambda_p, R_pract, R_theor, lambda_p_theor = normal_operation_period(time,
lambda_fail, N, p, del_t, k)
```

```
    Plot(time, lambda_p, lambda_p_theor, "practice", "theor")
    Plot(time, R_pract, R_theor, "practice", "theor")
    lambda_p, R_pract, R_theor, lambda_p_theor = aging_period(time,
lambda_fail, N, p, del_t, k)
    Plot(time, lambda_p, lambda_p_theor, "practice", "theor")
    Plot(time, R_pract, R_theor, "practice", "theor")
    plt.show()
```