

ГУАП

КАФЕДРА № 25

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ  
ПРЕПОДАВАТЕЛЬ

Доцент, канд. техн. наук	17.02.2023	Н.В. Марковская
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

Исследование интенсивности отказов для невозстанавливаемых систем

по курсу: Надежность инфокоммуникационных систем

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР.	3031	М.А Попов
		подпись, дата
		инициалы, фамилия

г. Санкт-Петербург  
2023 г.

Цель работы: исследовать 3 периода жизни невосстанавливаемой системы.

Вариант 17:

$$\square 1 = 0.8$$

$$\square 2 = 1.5$$

$$p = 0.5$$

$$n = 500000$$

## Описание программы

Программа выполняет вычисление функции надёжности и интенсивность отказов при моделировании трёх периодов.

Результаты работы программы:

Для первого периода:

Модель: Есть 2 группы систем, первая с интенсивностью отказов  $\lambda_1$ , вторая с  $\lambda_2$ , всего  $n$  систем, время работы каждой отдельной системы определяется как случайная величина с экспоненциальным распределением

, где количество систем работающих в данный момент

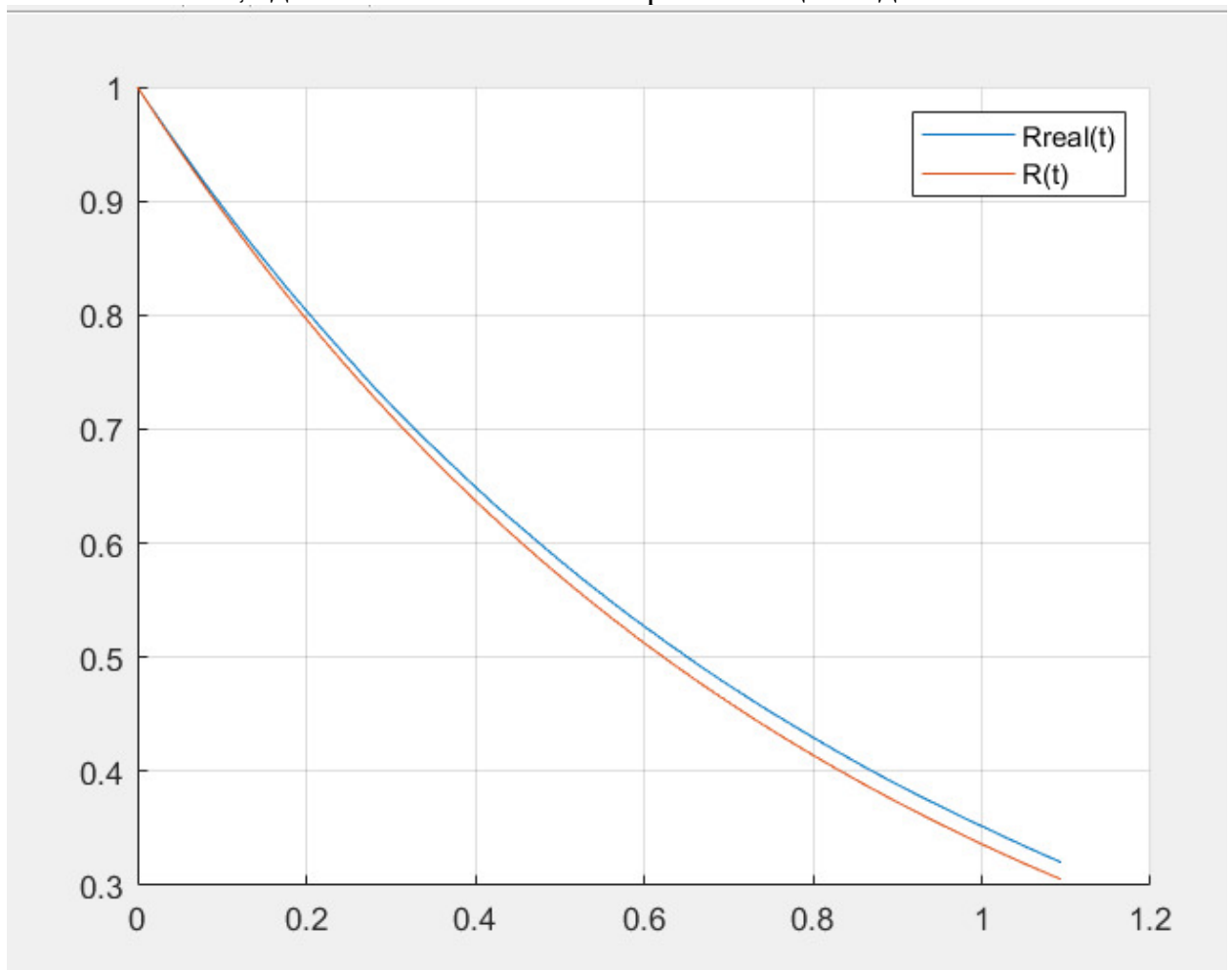


Рисунок 1 Результаты моделирования  $R(t)$

, где рассчитывается как одна пятая шага по интервалу

Теоретическое значение представляет собой линейный переход от  $\lambda_2$  к  $\lambda_1$

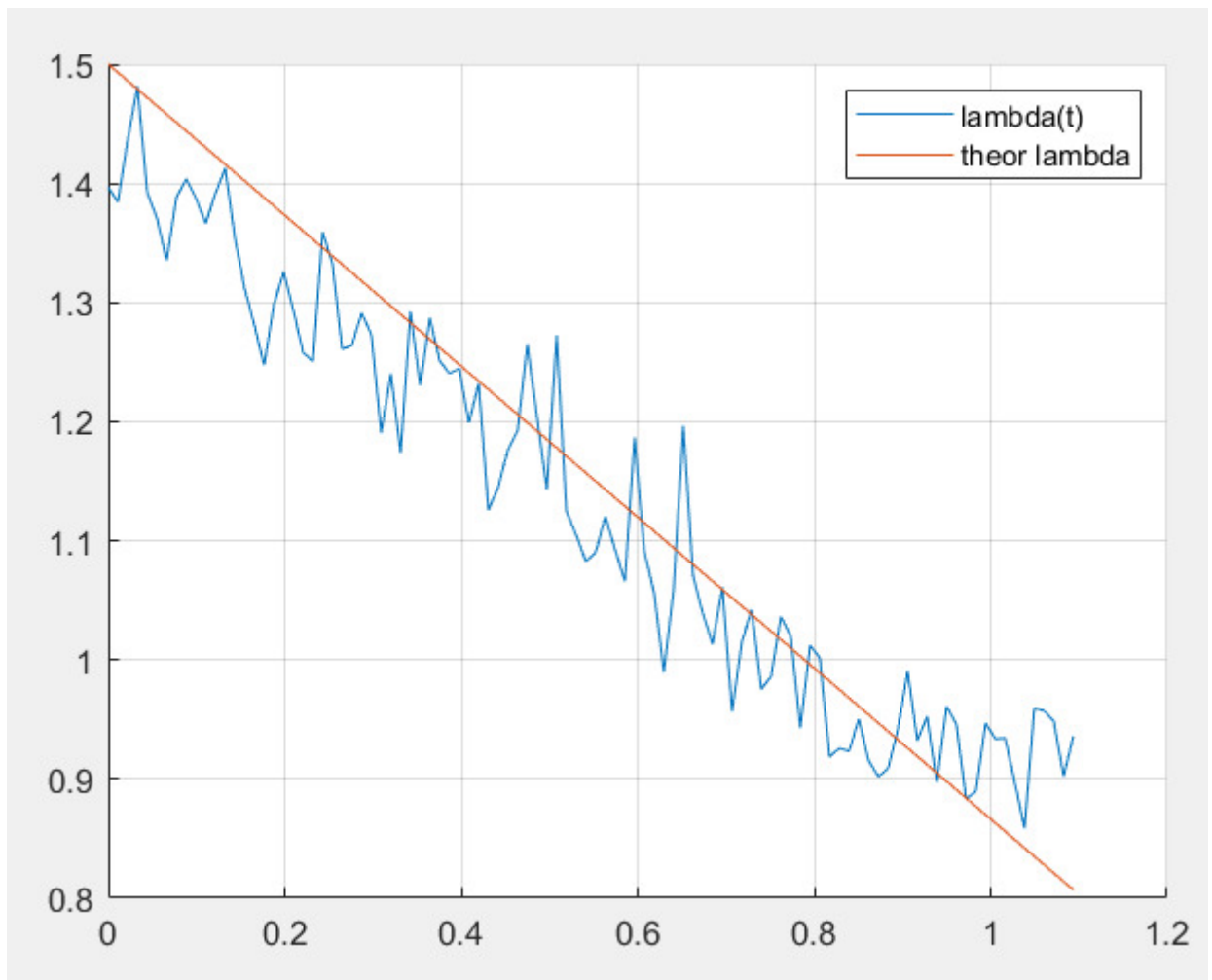


Рисунок 2 Результаты моделирования функции отказов

Для второго периода:

Модель: Есть  $n$  систем, состоящих из 2 элементов, подключённых последовательно, первая группа элементов с интенсивностью отказов  $\lambda_1$ , вторая с  $\lambda_2$ , время работы каждой отдельной системы определяется как минимум из двух случайных величин с экспоненциальным распределением

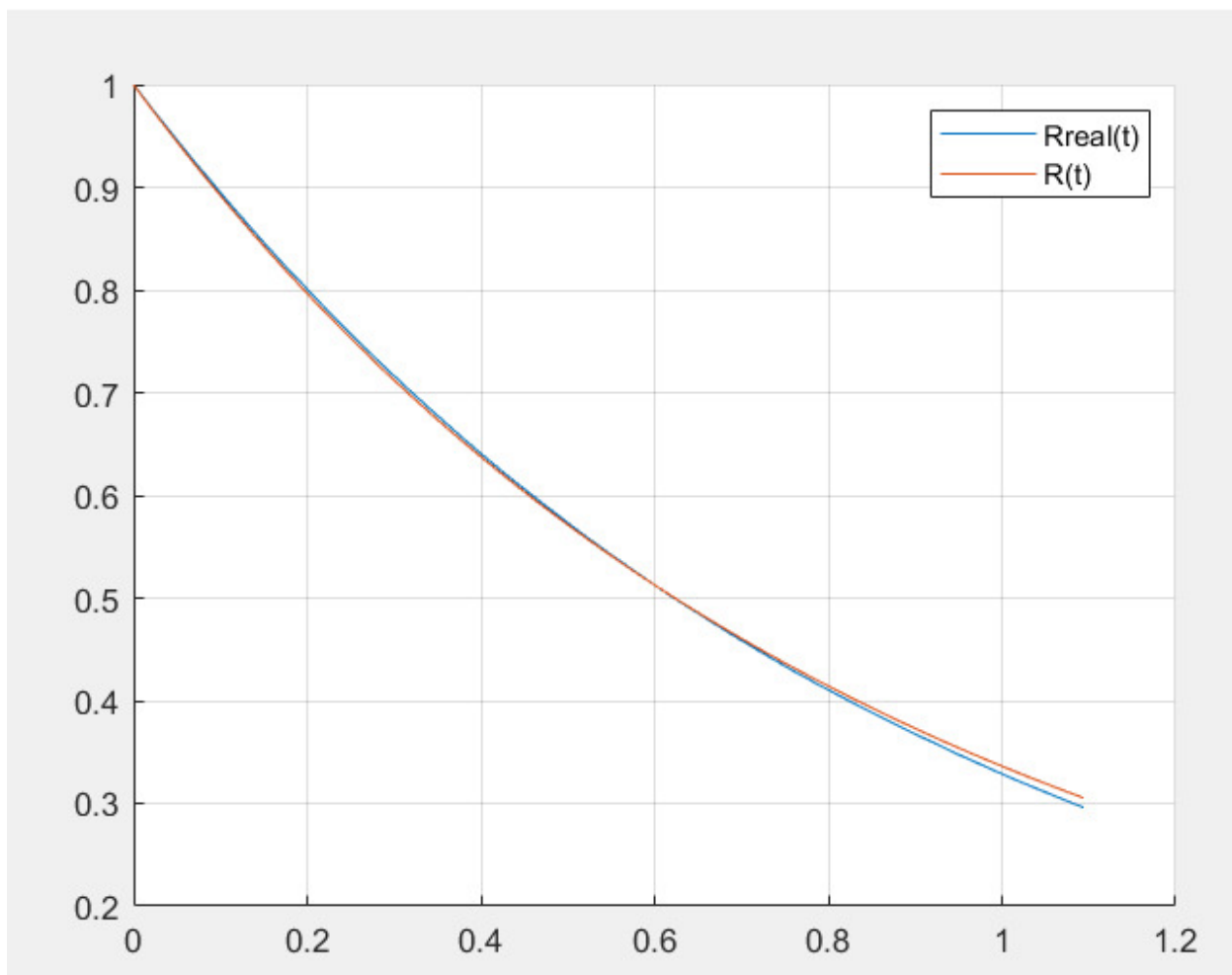


Рисунок 3 Результаты моделирования моделирования  $R(t)$

, где рассчитывается как одна пятая шага по интервалу

Теоретическое значение представляет собой сумму  $\square_2$  и  $\square_1$

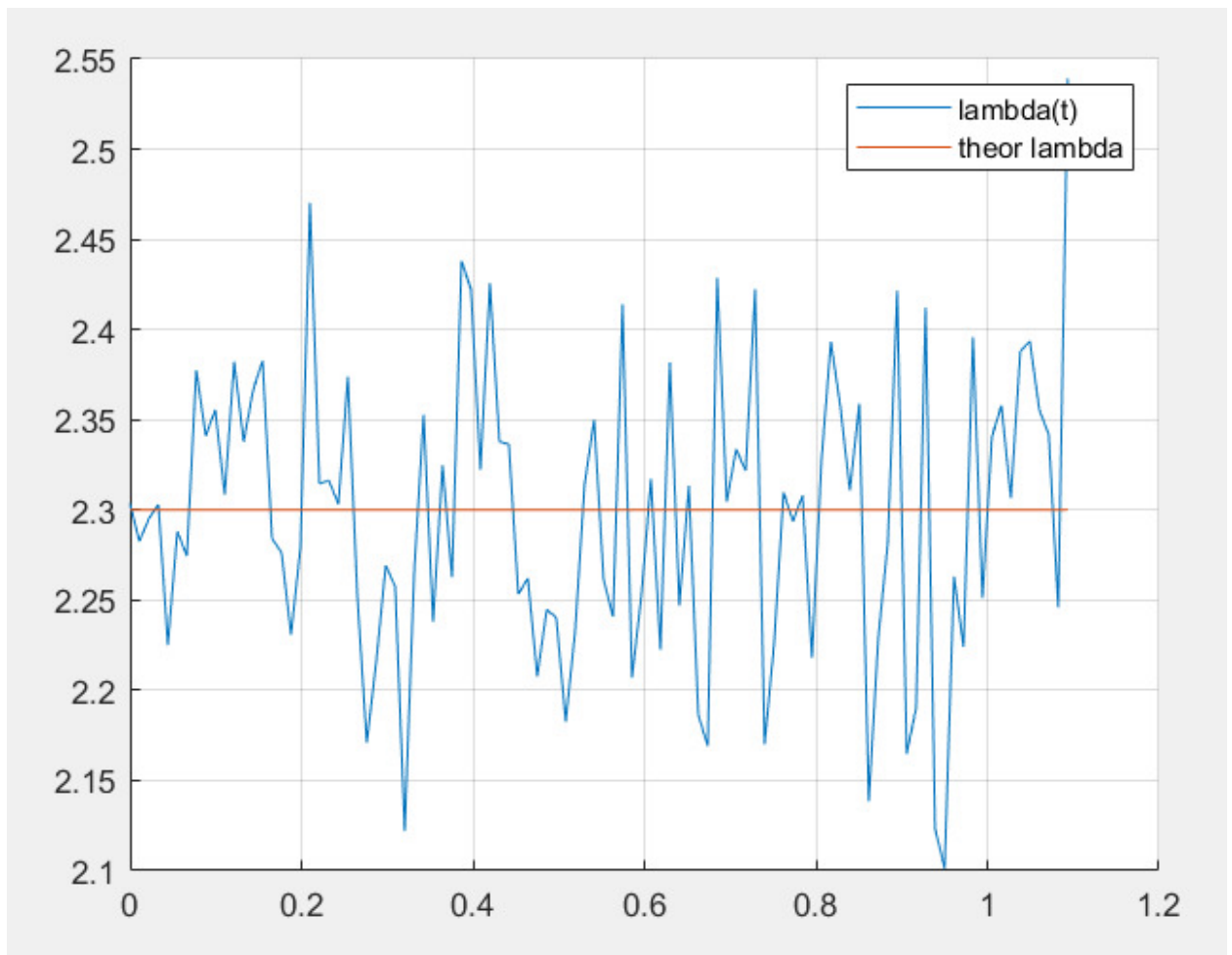


Рисунок 4 Результаты моделирования функции отказов

Для третьего периода:

Модель: Есть  $n$  систем, состоящих из 2 элементов, подключённых параллельно, первая группа элементов с интенсивностью отказов  $\lambda_1$ , вторая с  $\lambda_2$ , время работы каждой отдельной системы определяется как максимум из двух случайных величин с экспоненциальным распределением

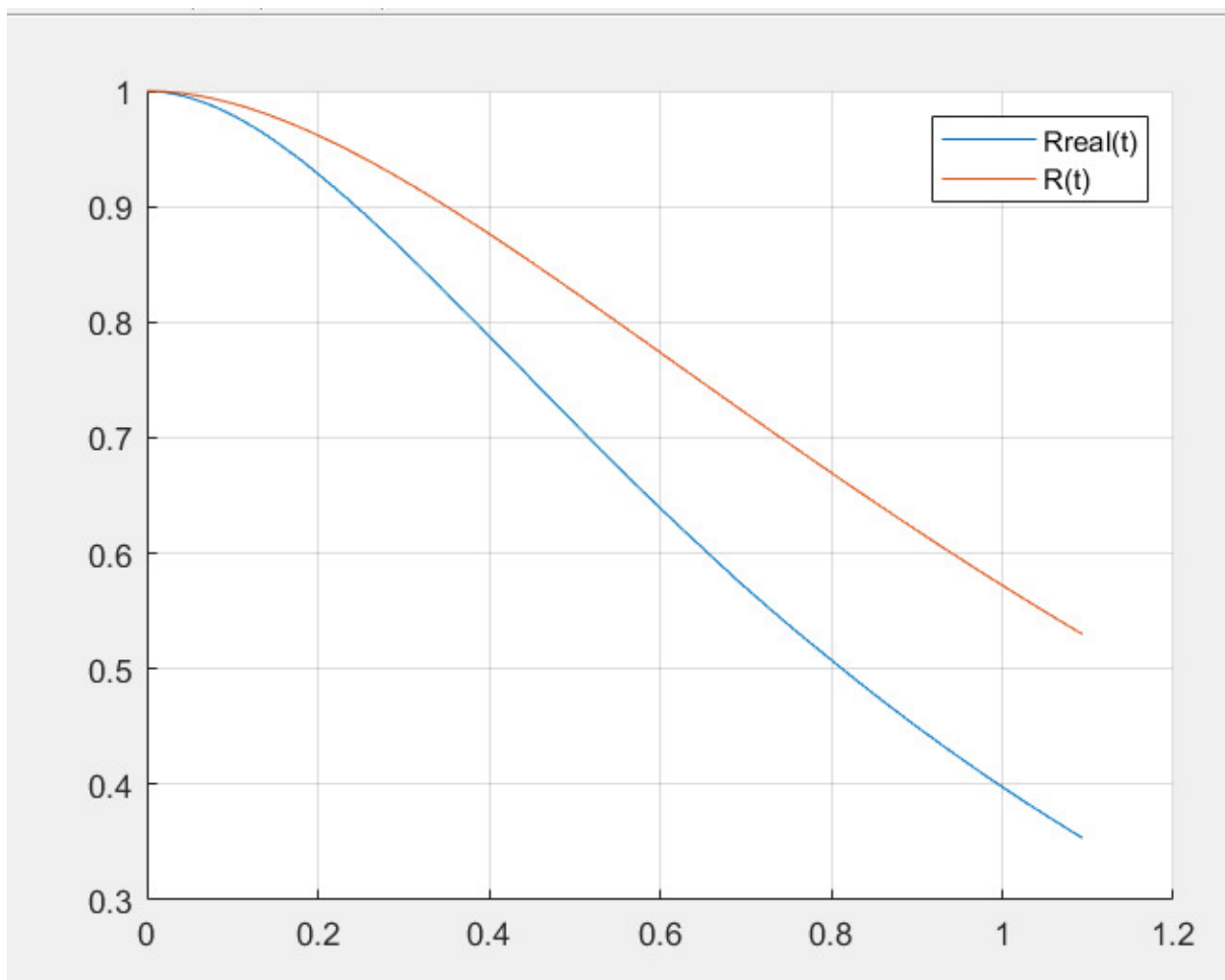


Рисунок 5 Результаты моделирования моделирования  $R(t)$

, где рассчитывается как одна пятая шага по интервалу

Теоретическое значение

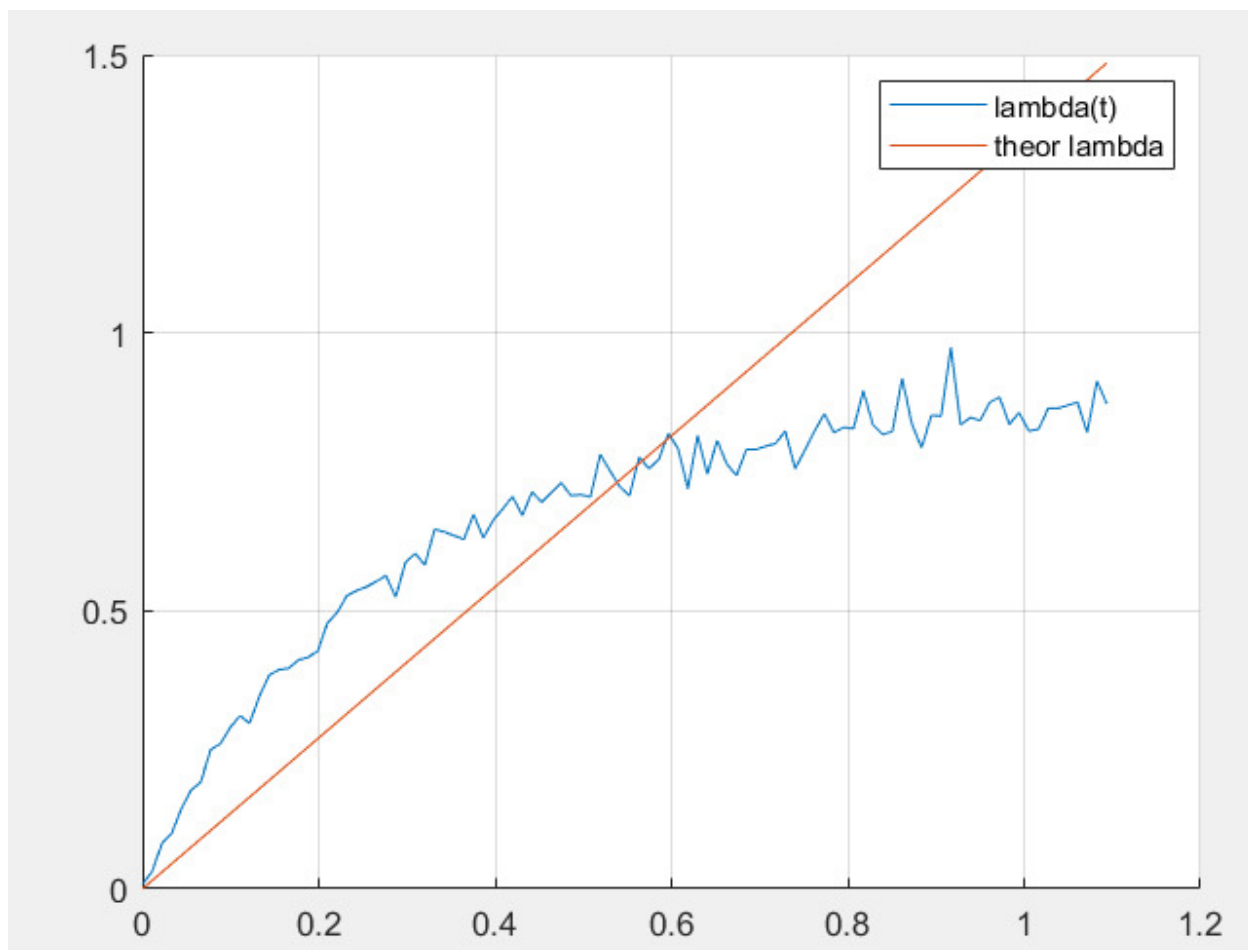


Рисунок 6 Результаты моделирования функции отказов



## Выводы

Результаты модулирования первого, второго и третьего периода схожи с теоретическими с поправкой на отклонения.

## Текст программы

```
#include <iostream>
#include <random>

void period1();

void period2();

void period3();

int main() {
    period3();
    return 0;
}

void period1() {
    std::random_device rd;
    std::mt19937_64 generator = std::mt19937_64(rd());
    double p = 0.5;
    std::bernoulli_distribution random = std::bernoulli_distribution(p);
    double lambda1 = 0.8;
    double lambda2 = 1.5;
    std::exponential_distribution taul =
std::exponential_distribution(lambda1);
    std::exponential_distribution tau2 =
std::exponential_distribution(lambda2);
    std::vector<double> group1;
    std::vector<double> group2;
    int n = 500000;
    for (int i = 0; i < n; ++i) {
        if (random(generator)) {
            group1.push_back(taul(generator) / lambda1);
        } else {
            group2.push_back(tau2(generator) / lambda2);
        }
    }
    double ttt = 0;
    for (double t: group1) {
        ttt += t;
    }
    for (double t: group2) {
        ttt += t;
    }
    ttt /= n;
    ttt += 0.1;
    double step = ttt / 100;
    double delta_step = step / 5;
    std::vector<double> rt;
    std::vector<double> lambdat;
    for (int i = 0; i < 100; i++) {
        int nt = 0;
        int ndt = 0;
        for (double t: group1) {
            if (t >= step * i) {
                ++nt;
            }
            if (t >= step * i + delta_step) {
                ++ndt;
            }
        }
        for (double t: group2) {
            if (t >= step * i) {
                ++nt;
            }
        }
    }
}
```

```

        }
        if (t >= step * i + delta_step) {
            ++nt;
        }
    }
    rt.push_back((double) nt / n);
    lambdat.push_back(((double) (nt - ndt) / nt) * (5 / step));
}
std::cout << step << std::endl;
std::cout << ttt << std::endl;
for (double v: rt) {
    std::cout << v << " , ";
}
std::cout << std::endl;
for (double v: lambdat) {
    std::cout << v << " , ";
}
std::cout << std::endl;
}

void period2() {
    std::random_device rd;
    std::mt19937_64 generator = std::mt19937_64(rd());
    double lambdal = 0.8;
    double lambda2 = 1.5;
    std::exponential_distribution tau1 =
std::exponential_distribution(lambdal);
    std::exponential_distribution tau2 =
std::exponential_distribution(lambda2);
    std::vector<double> group;
    int n = 500000;
    for (int i = 0; i < n; ++i) {
        group.push_back(std::min(tau1(generator) / lambdal, tau2(generator) /
lambda2));
    }
    double ttt = 0;
    for (double t: group) {
        ttt += t;
    }
    ttt /= n;
    ttt += 0.1;
    double step = ttt / 100;
    double delta_step = step / 5;
    std::vector<double> rt;
    std::vector<double> lambdat;
    for (int i = 0; i < 100; i++) {
        int nt = 0;
        int ndt = 0;
        for (double t: group) {
            if (t >= step * i) {
                ++nt;
            }
            if (t >= step * i + delta_step) {
                ++ndt;
            }
        }
        rt.push_back((double) nt / n);
        lambdat.push_back(((double) (nt - ndt) / nt) * (5 / step));
    }
    for (double v: rt) {
        std::cout << v << " , ";
    }
    std::cout << std::endl;
    for (double v: lambdat) {

```

```

        std::cout << v << " , ";
    }
    std::cout << std::endl;
}

void period3() {
    std::random_device rd;
    std::mt19937_64 generator = std::mt19937_64(rd());
    double lambda1 = 0.8;
    double lambda2 = 1.5;
    std::exponential_distribution taul =
std::exponential_distribution(lambda1);
    std::exponential_distribution tau2 =
std::exponential_distribution(lambda2);
    std::vector<double> group;
    int n = 500000;
    for (int i = 0; i < n; ++i) {
        group.push_back(std::max(taul(generator) / lambda1, tau2(generator) /
lambda2));
    }
    double ttt = 0;
    for (double t: group) {
        ttt += t;
    }
    ttt /= n;
    ttt += 0.1;
    double step = ttt / 100;
    double delta_step = step / 5;
    std::vector<double> rt;
    std::vector<double> lambdat;
    for (int i = 0; i < 100; i++) {
        int nt = 0;
        int ndt = 0;
        for (double t: group) {
            if (t >= step * i) {
                ++nt;
            }
            if (t >= step * i + delta_step) {
                ++ndt;
            }
        }
        rt.push_back((double) nt / n);
        lambdat.push_back(((double) (nt - ndt) / nt) * (5 / step));
    }
    for (double v: rt) {
        std::cout << v << " , ";
    }
    std::cout << std::endl;
    for (double v: lambdat) {
        std::cout << v << " , ";
    }
    std::cout << std::endl;
}

```