Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»**

**Институт Информационных технологий, математики и механики**

**Кафедра: программной инженерии**

Отчет по учебной практике:

Тема:

«Циклический алгоритм управления конфликтными потоками с адаптивной длинной цикла»

**Выполнил:** студент группы 381603-3

Кумин

Алексей Александрович

Подпись

**Научный руководитель:**

Ассистент

Евгений Владимирович

Кудрявцев

Подпись

Нижний Новгород

2018 г

Содержание

[1 Постановка задачи 3](#_Toc533549436)

[2 Описание случайных величин 4](#_Toc533549437)

[3 Рекуррентные соотношения 5](#_Toc533549438)

[4 Описание программы 6](#_Toc533549439)

[5 Алгоритмы 7](#_Toc533549440)

[6 Литература 8](#_Toc533549441)

# Постановка задачи

Требуется реализовать модель перекрестка с двумя очередями.

В системе присутствует 4 состояния и 2 пуассоновских потока:

1. Г1 – обслуживание заявок(автомобилей) по первому по току с длительностью T1
2. Г2 – дообслуживание заявок по первому потоку с длительностью Т2(обслуживается 1 заявка)
3. Г3 – обслуживание заявок по второму потоку с длительностью Т3
4. Г4 – дообслуживание заявок по второму потоку, Т4

Поступающие потоки такие, что для любой группы непересекающихся отрезков времени, число появившихся на их протяжении требований – взаимно независимые случайные величины.

Исходные данные: время Т1, Т2, Т3, Т4, параметры очередей λ1, λ2, время обслуживания одной заявки по потокам t1, t2, исходное кол-во заявок в очереди x1, x2, количество шагов системы N.

Выходные данные: среднее число заявок в очередях за время действия системы midX1, midX2, среднее время ожидания заявки в очередях mid T x1, mid T x2, таблица состояний в каждый момент времени (T1, T2, T3, T4) и количество заявок в очередях в эти моменты времени.

# Описание случайных величин

В данной задачи рассматриваются 2 пуассоновских потока, это означает, что вероятность поступления в промежуток t k требований:

Свойства:

1. Стационарность – вероятность появления k требований на промежутке (T, T + t) времени не зависит от T, а зависит только от t и k
2. Взаимная независимость появления требований - вероятность появления k требований на промежутке t не зависит от предыдущих требований.
3. Ординарность – невозможность появления нескольких требований в конкретный момент времени.

# Рекуррентные соотношения

Обозначим вероятность того, что на n+1 промежутке времени в очереди окажется k заявок:

где:

–длина очереди на n +1 промежутке времени

– кол-во заявок, пришедших на n промежутке времени

– кол-во заявок, обслуженных в течение n промежутка

Эти случайные величины – независимы, по определению, тогда:

Пусть – сколько может обслужиться заявок, тогда:

Отсюда следуют следующие рекуррентные соотношения:

# Описание программы

Программа моделирует перекресток с двумя пуассоновскими потоками. Интерфейс представлен на скриншотах.

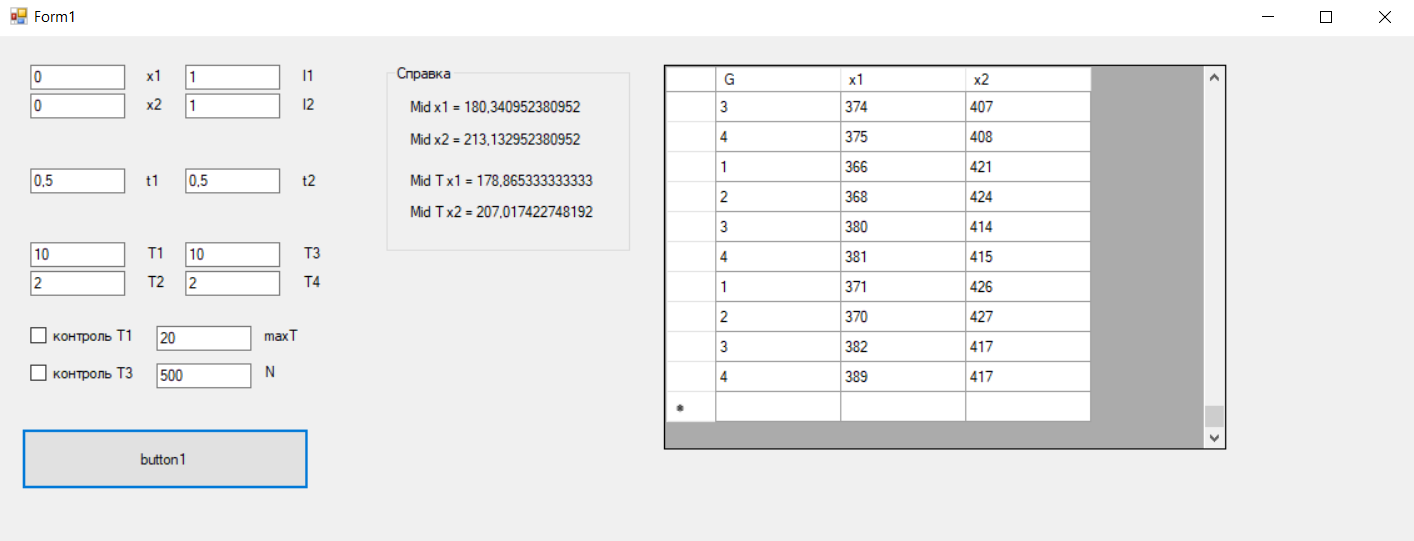


Рис.1

1. x1, x2 – начальное кол-во заявок в очередях
2. l1, l2 –интенсивность потоков.
3. t1, t2 – время обслуживания 1 заявки каждого из потоков
4. T1, T2, T3, T4 – время нахождения системы в каждом из состояний
5. Контроль T1 – регулирование времени T1, для того чтобы первый поток не увеличивался бесконечно.
6. Контроль T2 – регулирование времени T2, для того чтобы второй поток не увеличивался бесконечно.
7. Mid x1, Mid x2 – средняя длинна потоков
8. Mid Т x1, Mid Т x2 – среднее ожидание обслуживания

На рисунке 1 видно, что очереди неограниченно растут, при данном наборе параметров. Данную ситуацию можно исправить, уменьшив время обслуживания одной заявки, либо уменьшив интенсивность потока, либо увеличив T1 и T3 для каждого из потоков соответственно (при этом другой поток будет увеличиваться с еще большей скоростью)

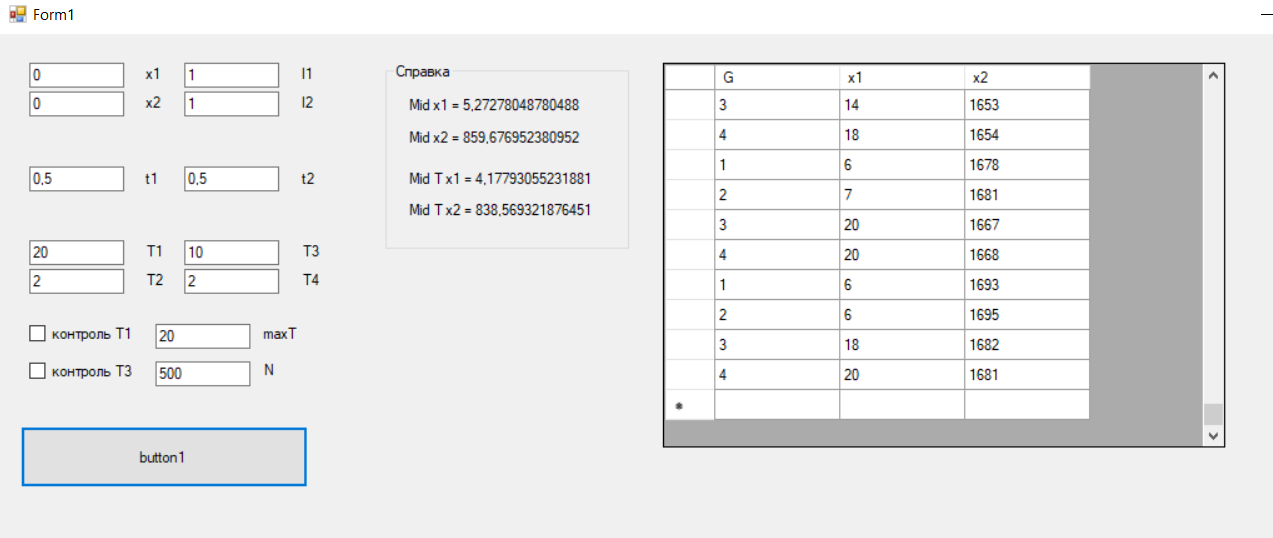


Рис.2

На рисунке 2 видно, что если увеличить параметр T1, то по первому потоку не будет наблюдаться увеличения очереди, а по второму наблюдается увеличение с еще большей скоростью.

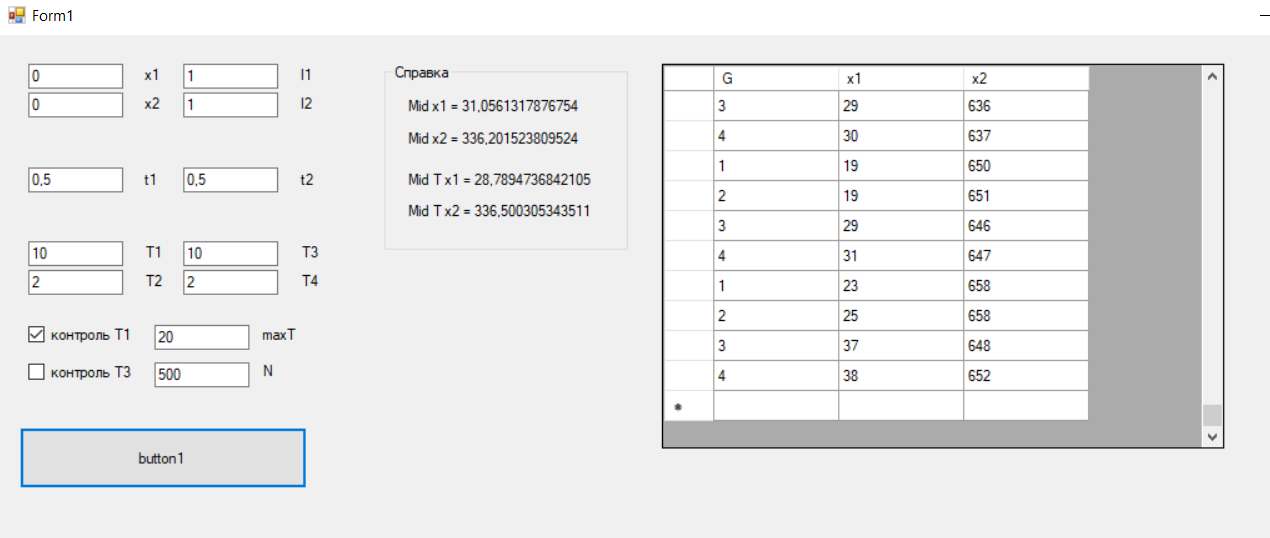


Рис.3

Наблюдается аналогичная ситуация.

Вывод: можно подобрать такой набор параметров, при котором очереди не будут увеличиваться и в системе будет наблюдаться стационар. При регулировании состояний T1, T3 стационара можно добиться только в одном из состояний (при удовлетворительных t1, t2, l1, l2).

# Алгоритмы

## Моделирование случайных величин

### 1 способ

double relX(double a)

{

return (-log(1 - ((double)(rand() + 0.5) / (double)RAND\_MAX)) / a);

}

int relETA(double a, double t)

{

double sum = relX(a);

int i = 0;

while (t > sum)

{

sum += relX(a);

i++;

}

return i;

}

### 2 способ

long int fact(int N)

{

if (N == 1 || N == 0)

return 1;

return N \* fact(N - 1);

}

int relETA1(double a, double t)

{

double eta = exp(a \* t) \*(double)rand() / (double)RAND\_MAX;

double sum = 0;

int i = 0;

while (eta > sum)

{

sum += pow(a \* t, i) / (double)fact(i);

i++;

}

return i - 1;

}

### Общее моделирование

int Eta(double a, double t)

{

if ((a \* t) < 20)

return relETA1(a, t);

else

return relETA(a, t);

}

## Класс для моделирования потока StreamModel

### StreamModel.h

#pragma once

#include <vector>

#include <iostream>

#include <random>

#include <time.h>

#include <math.h>

using namespace std;

class StreamModel

{

public:

int key;//состояние

int x;//сколько было в очереди

double t; //время обработки заявки

double l;//интенстивность потока

vector<int> Auto;

vector<double> T;

vector<double> mesX;//массив кол-ва заявок в разные промежутки времени

StreamModel(int KEY = 0, int X = 0, double T = 1, double L = 1);

void move(double T);

void service(double T);

void stagnation(double T);

double Med\_x();

double Med\_Tx();

~StreamModel();

};

### Простой потока

void StreamModel::stagnation(double T1)

{

int ETA = Eta(l, T1);

for (int i = 0; i < T.size(); i++)

if (Auto[i] == 1)

T[i] += T1;

for (int i = 0; i < (ETA); i++)

{

Auto.push\_back(1);

T.push\_back(0);

}

//....

x += ETA;

}

### Дообслуживание одной заявки

void StreamModel::service(double T1)

{

if (T1 >= t)

{

int ETA = Eta(l, T1);

if ((x + ETA - 1) >= 0)

{

//.....

int k = 1;

for (int i = 0; i < Auto.size(); i++)

{

if (Auto[i] == 1)

{

Auto[i] = 0;

k = 0;

break;

}

}

for (int i = 0; i < T.size(); i++)

{

if (Auto[i] == 1)

T[i] += T1;

}

for (int i = 0; i < (ETA); i++)

{

Auto.push\_back(1);

T.push\_back(0);

}

if (k)

{

for (int i = 0; i < Auto.size(); i++)

{

if (Auto[i] == 1)

{

Auto[i] = 0;

break;

}

}

}

//.....

x += ETA - 1;

}

else

{

//......

for (int i = 0; i < T.size(); i++)

if (Auto[i] == 1)

T[i] += T1;

for (int i = 0; i < (ETA); i++)

{

Auto.push\_back(1);

T.push\_back(0);

}

//......

x += ETA;

}

}

mesX.push\_back(x);

}

### Обслуживание потока

void StreamModel::move(double T1)

{

double sumT = 0;

while (sumT < T1)

{

sumT += t;

service(t);

}

}

### Вычисление средний длинны очереди и среднего ожидания обслуживания

double Med(vector<double> mes)

{

double Sum = 0;

for (int i = 0; i < mes.size(); i++)

Sum += mes[i];

return (Sum / (double)mes.size());

}

double StreamModel::Med\_x()

{

return Med(mesX);

}

double StreamModel::Med\_Tx()

{

return Med(T);

}

# Литература

1. Зорин А.В, Зорин В.А, Федоткин М.А. «Теория управляемых систем массового обслуживания: Учебное пособие.» Нижний Новгород: Издательство Нижегородского госуниверситета, 2007 г. – 47 с.
2. Гнеденко Б.В., Коваленко И.Н. «Введение в теорию массового обслуживания» М.: Наука, 1966. — 432 с.