*Федеральное государственное бюджетное образовательное учреждение* *высшего образования*

|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | ***«Московский государственный технический университет  имени Н.Э. Баумана***  ***(национальный исследовательский университет)»***  ***(МГТУ им. Н.Э. Баумана)*** |

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

**Отчет**

**по лабораторной работе №1**

**Дисциплина:** Моделирование

**Название лабораторной работы:**

Моделирование Марковских процессов

Студент гр. ИУ6-52  **\_\_\_\_\_\_21.10.2018 Бурлаков А.С.**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_ Шайхутдинов А.А.**

(Подпись, дата) (И.О. Фамилия)

Москва, 2018

# ВВЕДЕНИЕ

Задание: создать программу для моделирования Марковских процессов следующими методами: аналитическим, для непрерывного и для дискретного времени.

# ОСНОВНАЯ ЧАСТЬ

Схемы алгоритмов:

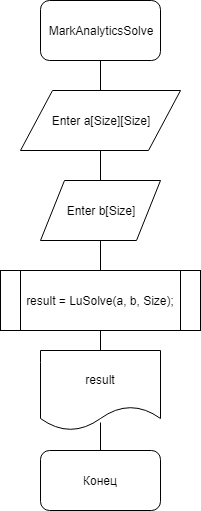


Рисунок 1 – Схема алгоритма для аналитического решения

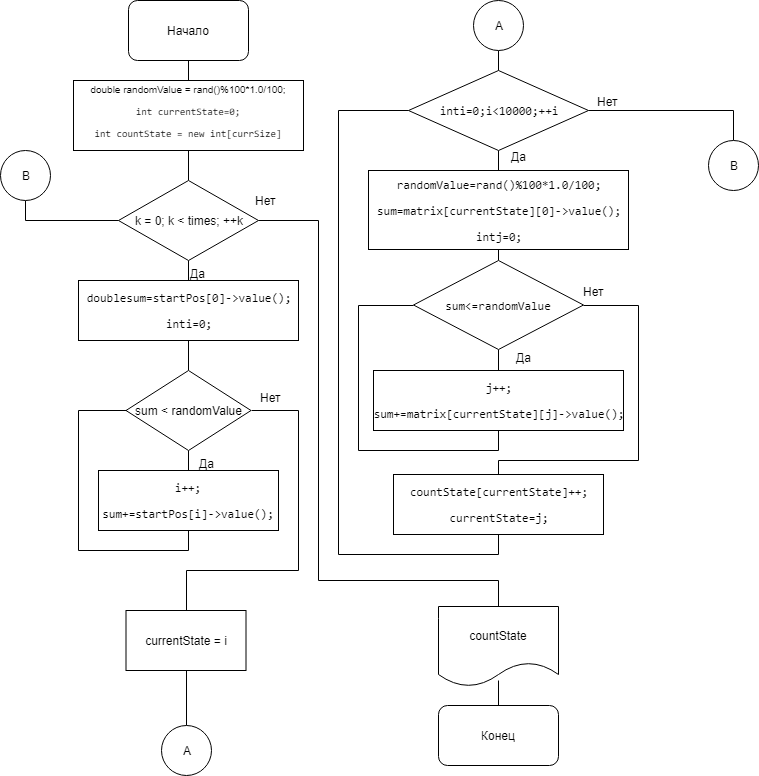


Рисунок 2 – Схема алгоритма для решения с дискретным временем

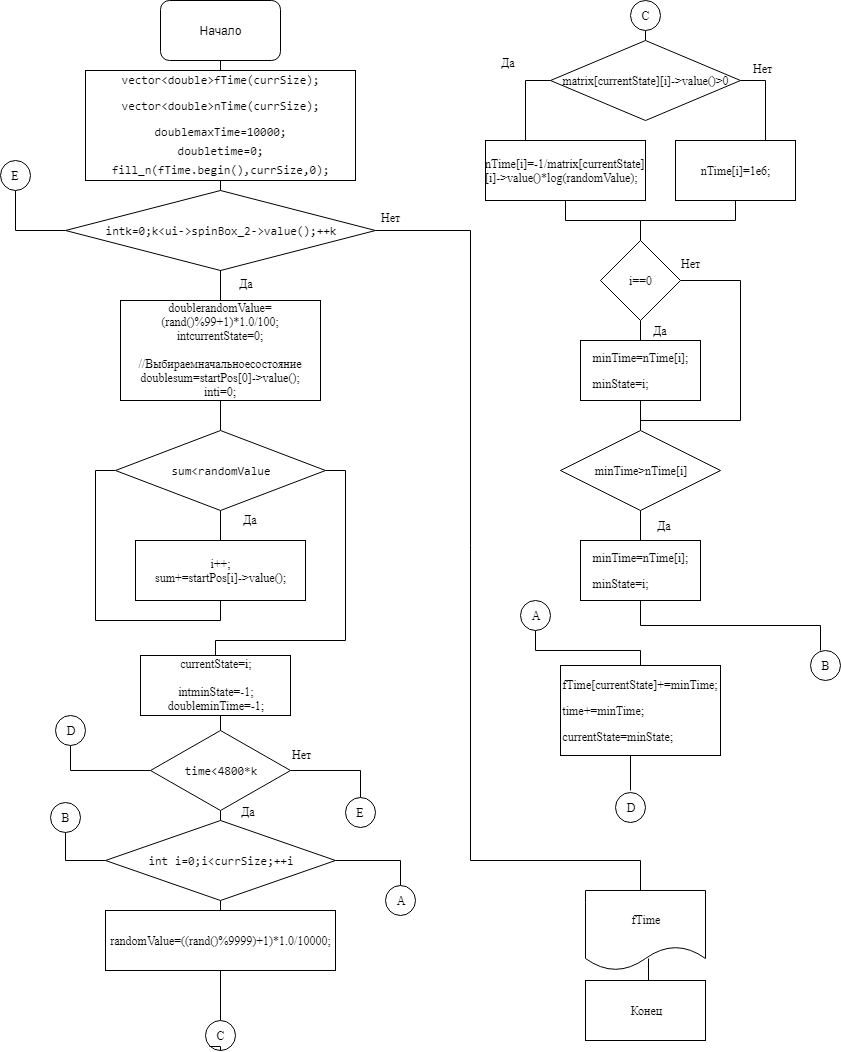


Рисунок 3 – Схема алгоритма для решения с непрерывным временем

**Код подпрограмм:**

**MarkAnalyticsSolve:**

int kSize = currSize + 1;

vector<vector<double>> a;

a.resize(kSize);

fill\_n(a.begin(), kSize, vector<double>(kSize));

for (int i = 0; i < kSize - 1; ++i){

for (int j = 0; j < kSize - 1; ++j){

a[j][i] = matrix[i][j]->value();

if (i == j){

a[i][j] -= 1;

}

}

}

for (int i = 0; i < kSize - 2; ++i){

a[kSize - 1][i] = 1;

}

vector<double> b;

b.resize(kSize);

fill\_n(b.begin(), kSize, 0);

b[kSize - 1] = 1;

vector<double> result1 = LuSolve(a, b, kSize);

for (int i = 0; i < kSize; ++i){

finale\_2[i]->setValue(result1[i]);

}

**MarkDiscretTime:**

**// Метод 1**

**double randomValue = rand() % 100 \* 1.0 / 100;**

**int currentState = 0;**

**// Назначение массива (Сколько раз был в состоянии)**

**int \*countState = new int[currSize];**

**for (int i = 0; i < currSize; ++i)**

**countState[i] = 0;**

**// Начало решения**

**for (int k = 0; k < ui->spinBox\_2->value(); ++k){**

**// Установка начального состояния**

**double sum = startPos[0]->value();**

**int i = 0;**

**while (sum < randomValue){**

**i++;**

**sum += startPos[i]->value();**

**}**

**currentState = i;**

**// Решение**

**for (int i = 0; i < 10000; ++i){**

**randomValue = rand() % 100 \* 1.0 / 100;**

**sum = matrix[currentState][0]->value();**

**int j = 0;**

**while (sum <= randomValue){**

**j++;**

**sum += matrix[currentState][j]->value();**

**}**

**countState[currentState]++;**

**currentState = j;**

**}**

**}**

**// Вывод финальных вероятностей**

**for (int i = 0; i < currSize; ++i){**

**finale\_1[i]->setValue(countState[i] \* 1.0 / (10000 \* ui->spinBox\_2->value()));**

**}**

**MarkEndlessTime:**

vector<double> fTime(currSize);

vector<double> nTime(currSize);

double maxTime = 10000;

double time = 0;

// Обнуление переменных

fill\_n(fTime.begin(), currSize, 0);

for (int k = 0; k < ui->spinBox\_2->value(); ++k){

double randomValue = rand() % 100 \* 1.0 / 100;

int currentState = 0;

// Выбираем начальное состояние

double sum = startPos[0]->value();

int i = 0;

while (sum < randomValue){

i++;

sum += startPos[i]->value();

}

currentState = i;

int minState = -1;

double minTime = -1;

// Имитируем работу в течении n-ого времени

while (time < 4800 \* k){

for (int i = 0; i < currSize; ++i){

randomValue = (rand() % 9999) \* 1.0 / 10000;

if (matrix[currentState][i]->value() > 0)

nTime[i] = - 1 / matrix[currentState][i]->value() \* log(randomValue);

else

nTime[i] = 100;

if (i == 0){

minTime = nTime[i];

minState = i;

}

if (minTime > nTime[i]){

minTime = nTime[i];

minState = i;

}

}

fTime[currentState] += minTime;

time += minTime;

currentState = minState;

}

}

cout << "start" << endl;

for (int i = 0; i < currSize; ++i){

finale\_3[i]->setValue(fTime[i] / time);

cout << fTime[i] << endl;

}

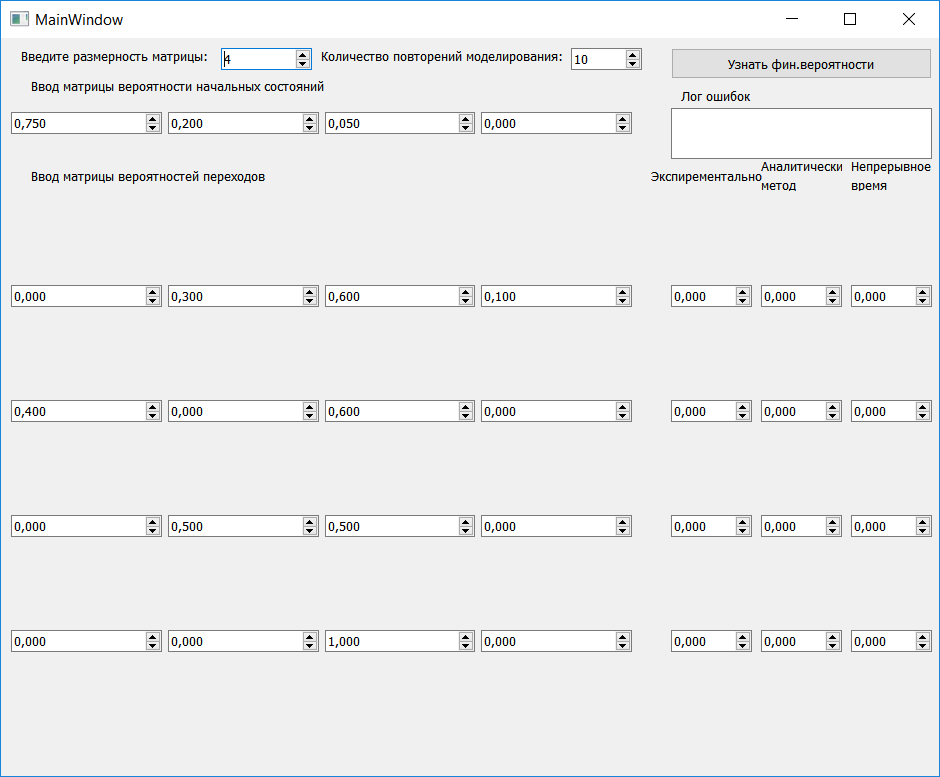


Рисунок 4 – Пользовательский интерфейс

Тестирование программы

Входные данные:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | [0] | [1] | [2] | [3] |
| a[0] | 0 | 0,3 | 0,6 | 0,1 |
| a[1] | 0,4 | 0 | 0,6 | 0 |
| a[2] | 0 | 0,5 | 0,5 | 0 |
| a[3] | 0 | 0 | 1 | 0 |
| p0 | 0,75 | 0,2 | 0,05 | 0 |

Выходные данные:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Правильно | Аналитический | Дискретное время | Непрерывное время |
| 1 | p0 = 0.125  p1 = 0.3125  p2 = 0.55  p3 = 0.0125 | p0 = 0.127  p1 = 0.316  p2 = 0.557  p3 = 0.013 | p0 = 0.125  p1 = 0.313  p2 = 0.55  p3 = 0.013 | p0 = 0.125  p1 = 0.311  p2 = 0.552  p3 = 0.013 |

# Заключение

1. В программной среде Qt creator написана программа, позволяющая смоделировать марковские процессы аналитически, с дискретным и с непрерывным временем;
2. Программа протестирована. Тестирование показало правильность работы программы.