Белорусский Госдарственный Университет Информатики и Радиоэлектроники

Кафедра ПОИТ

Лабораторная работа №3

«Построение и исследование аналитической модели дискретно -стохастической системы массового обслуживания»

Вариант 37

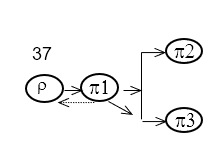
|  |  |
| --- | --- |
| Выполнил:  студент гр.751002 | Авко Д. А. |
| Проверил: | Мельник Н.И. |

Минск 2020

1. ЦЕЛЬ РАБОТЫ

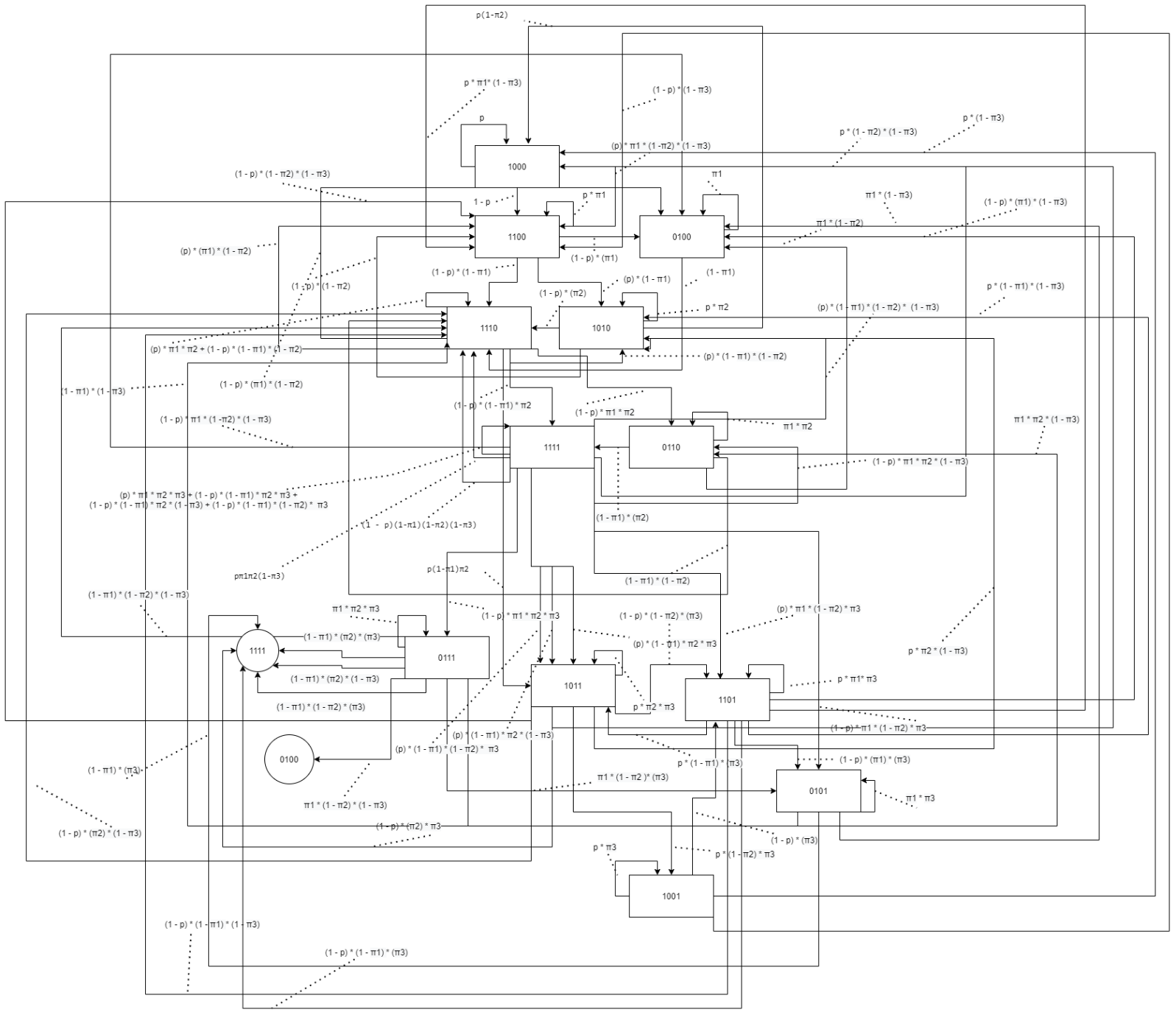
Построение аналитической модели дискретно-стохастической системы и расчет параметров данной системы с использованием построенной модели.

1. ИСХОДНЫЕ ДАННЫЕ К РАБОТЕ

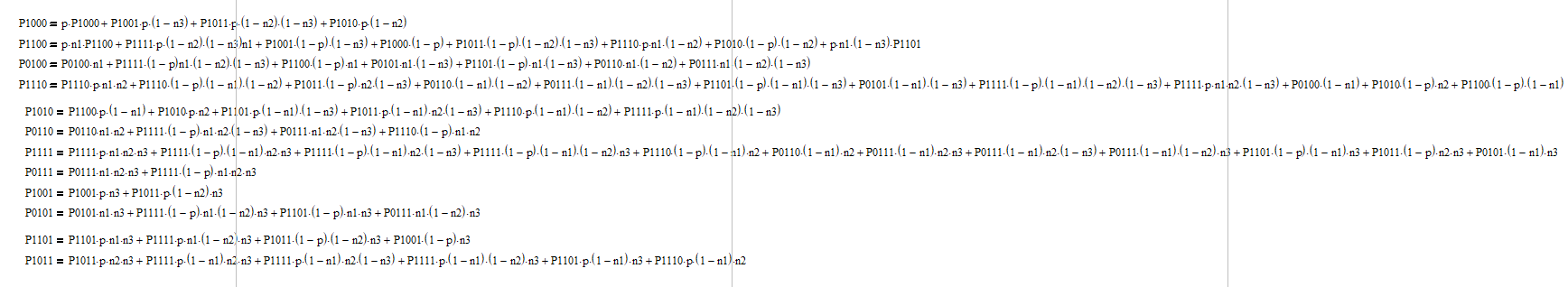


|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Вариант | ρ | π1 | π2 | π3 |
| 37 | 0.4 | 0,5 | 0,8 | 0.8 |

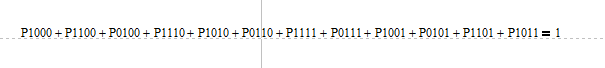
1. ХОД РАБОТЫ



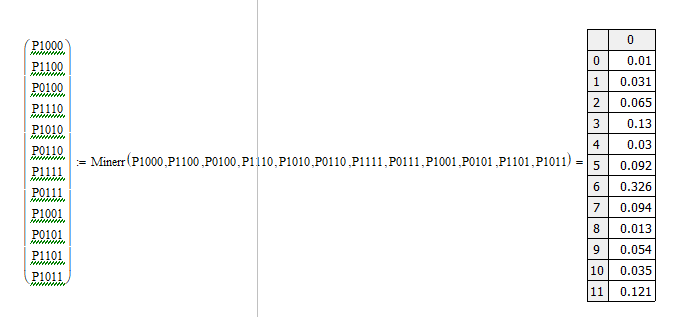
На основе графа переходов построим СЛАУ:

где p=0,4 n1=0.5, n2=0.8, n3=0.8

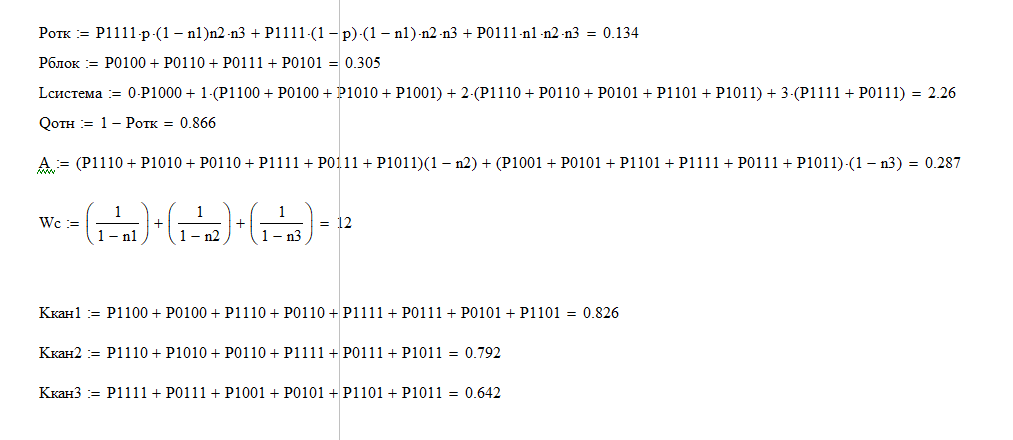
Воспользовавшись нормировочным уравнением:



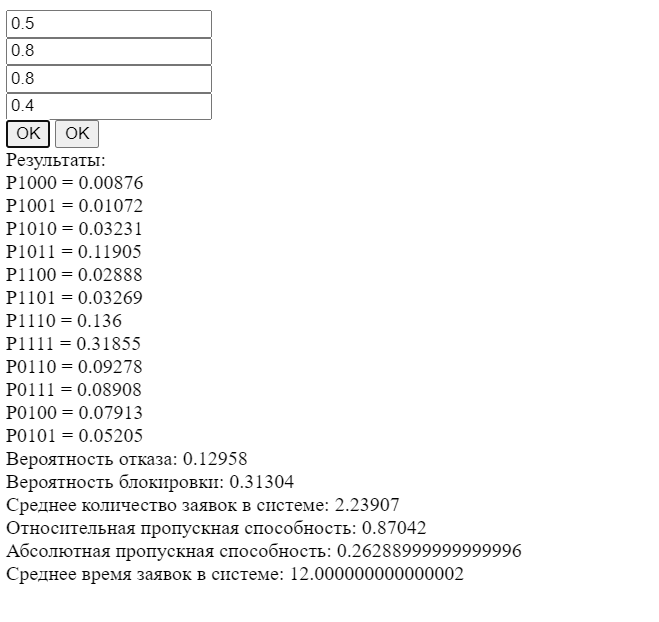
и заменив в исходной системе любое уравнение данным, найдем решение СЛАУ при заданных p, n1 и n2:



Рассчитаем показатели эффективности:



1. РЕЗУЛЬТАТ РАБОТЫ ПРОГРАММЫ



1. ВЫВОДЫ

В данной работе была исследована дискретно-стохастическая модель СМО. Для нее был построен граф состояний, на основе которого была составлена СЛАУ и рассчитаны вероятности нахождения системы в каждом из состояний. Произведен расчет данных показателей эффективности: Pотк, Pблок, Lc, Q, A, Wc, Kкан1, Kкан2. Проведено имитационное моделирование системы. В ходе сравнения полученных статистических показателей с расчетными показателями сильных отклонений не выявлено. Текущие отклонения связаны со статистической погрешностью.

1. КОД ПРОГРАММЫ

document.getElementById("ok").addEventListener("click", (ev => {

    let pi1 = Number(document.getElementById("pi1").value);

    let pi2 = Number(document.getElementById("pi2").value);

    let pi3 = Number(document.getElementById("pi3").value);

    let p = Number(document.getElementById("p").value);

    const numberOfSteps = 100000;

    let smo = new SMO(pi1, pi2, pi3, p);

    for (let i = 0; i < numberOfSteps; i++) {

        smo.processNextStep();

    }

    let result = "Результаты: <br/>";

    let meanSystem = 0;

    let A = 0;

    console.log(smo.stats);

    console.log(smo.stats.length);

    for (let key in smo.stats) {

        if (smo.stats[key] !== 1) {

            result += `P${key} = ${smo.stats[key] / numberOfSteps}<br/>`;

            meanSystem += smo.stats[key] / numberOfSteps \* (Number(key[1]) + Number(key[2]) + Number(key[3]));

            if ((key[3] == 1 || key[2] == 1) && (key[2] !== key[3])) {

                A += smo.stats[key] / numberOfSteps;

            }

        }

    }

    A \*= (1 - pi2);

    let pBlock = smo.blocked / numberOfSteps;

    let lambda = 0.5 \* (1 - pBlock);

    let pRej = smo.rejected / numberOfSteps;

    let Q = 1 - pRej;

    let meanTimeC1 = 1 / (1 - pi1);

    let meanTimeC2 = 1 / (1 - pi2);

    let meanTimeC3 = 1 / (1 - pi3);

    let meanTimeInSystem = meanTimeC1 + meanTimeC2 + meanTimeC3;

    result += `Вероятность отказа: ${pRej}<br/>`;

    result += `Вероятность блокировки: ${pBlock}<br/>`;

    result += `Среднее количество заявок в системе: ${meanSystem}<br/>`;

    result += `Относительная пропускная способность: ${Q}<br/>`;

    result += `Абсолютная пропускная способность: ${A}<br/>`;

    result += `Среднее время заявок в системе: ${meanTimeInSystem}<br/>`;

    document.getElementById("result").innerHTML = result;

}));

function\* lemenGenerator() {

    const m = 2 \*\* 32;

    const a = 1664525;

    const c = 1013904223;

    let R = 1;

    while (true) {

        R = (a \* R + c) % m;

        yield R / m;

    }

}

class Stat {

    source;

    channel1;

    channel2;

    channel3;

    constructor(source, channel1, channel2, channel3) {

        this.source = source;

        this.channel1 = channel1;

        this.channel2 = channel2;

        this.channel3 = channel3;

    }

    toString() {

        return `${this.source}${this.channel1}${this.channel2}${this.channel3}`;

    }

}

class SMO {

    constructor(pi1, pi2, pi3, p) {

        this.pi1 = pi1;

        this.pi2 = pi2;

        this.pi3 = pi3;

        this.p = p;

        this.channel1 = 0;

        this.channel2 = 0;

        this.channel3 = 0;

        this.source = 0;

        this.rejected = 0;

        this.blocked = 0;

        this.stats = {};

        this.saveStats();

        this.generator = lemenGenerator();

    }

    processNextStep() {

        if(this.channel3 && (1 - this.pi3 > this.generator.next().value)){

            this.channel3 = 0;

        }

        if(this.channel2 &&  (1 - this.pi2 > this.generator.next().value)){

            this.channel2 = 0;

        }

        if(this.channel1 && (1 - this.pi1 > this.generator.next().value)){

            if(this.channel2 && this.channel3){

                this.channel1 = 0;

                this.rejected++;

            }else if(!this.channel2){

                this.channel1 = 0;

                this.channel2 = 1;

            }else if (!this.channel3){

                this.channel1 = 0;

                this.channel3 = 1;

            }

        }

        if(this.source){

            if((1 - this.p > this.generator.next().value)){

                if(this.channel1){

                    this.source = 0;

                    this.blocked++;

                }else{

                    this.channel1 = 1;

                }

            }else{

            }

        }else if(!this.source){

            if(!this.channel1){

                this.source = 1;

                this.channel1 = 1;

            }else{

                this.blocked++;

            }

        }

        // if((1 - this.p > this.generator.next().value) && this.source && !this.channel1){

        //     this.source = 1;

        //     this.channel1 = 1;

        // }

        // if((1 - this.p > this.generator.next().value) && this.source && this.channel1){

        //     this.source = 0;

        //     this.blocked++;

        // }

        //01 - 01

        this.saveStats();

    }

    saveStats() {

        let stat = new Stat(this.source, this.channel1, this.channel2, this.channel3);

        if (!this.stats[stat]) {

            this.stats[stat] = 1;

        } else {

            this.stats[stat]++;

        }

    }

}