Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Кафедра ИС

Отчёт

По лабораторной работе № 2-5

КРИТЕРИИ ЭФФЕКТИВНОСТИ. КОЛИЧЕСТВЕННЫЕ ОЦЕНКИ

СТРУКТУРНО-ТОПОЛОГИЧЕСКИЕ ХАРАКТЕРИ- СТИКИ СИСТЕМ

АНАЛИЗ УПРАВЛЯЮЩИХ ЗАМКНУТЫХ СИСТЕМ НЕПРЕРЫВНОГО ТИПА

ПОНЯТИЕ СЛОЖНОСТИ СИСТЕМНЫХ ЗАДАЧ. ПОСТРОЕНРИЕ СИСТЕМНЫХ МОДЕЛЕЙ ПРОБЛЕМНЫХ СИТУАЦИЙ

Выполнил: ст. гр. ИС/б-31-о

Куркчи А. Э.

Проверил:

Кузнецов С.А.

Севастополь

2016

1.Цель работы

Построение обобщенного (интегрального) критерия эффективности системотехнических комплексов (СТК).

Определения количественных оценок графа для сравнения различных вариантов построения систем..

Линеаризование уравнения замкнутой системы заданной в нелинейном виде. Анализа устойчивости линейных систем.

Изучение понятия сложности системной задачи, спектры сложности, трансвычислительная сложность. Оценка алгоритмической разрешимости системной задачи.

2.Постановка задачи

Задание 1

а) Записать интегральный критерий эффективности СТК для m=3 и n=8, если оценки получены методом ранжировки (Таблица 4).

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Критерий | 1 Эксперт | | | | 2 Эксперт | | | | 3 Эксперт | | |
| места | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 |
| 11 | 1,2,3,4 | 7 | 5,6 | 8 | 1,4,5 | 8 | 2,6 | 3,7 | 1,4,5 | 6,7 | 2,3,8 |

Таблица 4 – Варианты заданий

б) Решить задачу получения экспертных оценок методом  
последовательных приближений. Число частных критериев n=8, m=l. Придумать первичный ряд оценок самостоятельно (наивысшая оценка - 1, наименьшая- 0) и уточнить их с помощью системы решений, заданной вариантом (Таблица 5).

Таблица 5 – Варианты заданий

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № вар. | Отношения | | | | | |
|  | 1 | 2 | 3 | 4 | 5 | 6 |
| 11 | < | > | < | > | < | > |

Задание 2

 Вариант 1

Задание 3

1. Дано уравнение разомкнутой системы. Необходимо его линеаризовать, записать передаточную функцию W(p), замкнуть систему, получив Ф(p) и D(p). Определить устойчивость по Гурвицу. В начальный момент времени все производные равны нулю.  X - выходная координата, Y - входная координата.

Варианты:

*16. 2xy+0,3x2+0,5x+0,1x + 0,2y; = 0; х°=у°=2;*

2. Дана передаточная функция разомкнутой системы:

*W(p)=*

и закон изменения входного воздействия: *g(t) = m0t2 + m1 t + т2.*

Найти закон изменения во времени ошибки x(t).

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № вар | b 0 | b1 | d0 | d1 | d2 | mо | m1 | m2 |
| 16 | 35 | 0 | 2,5 | 0,75 | 1 | 61 | 7 | 0,7 |

Задание 4

1. Записать временные функции сложности для n!, (n+1)2; nn, 22n.
2. Определить, какие из следующих соотношений истины,а какие нет:

а) 3n5+10n3+n2+25 имеет сложность O(n)5;

б) 2n+3n имеет сложность O(2n);

в) 2n+3n имеет сложность O(3n);

г) n! имеет сложность O(nn);

д) nn имеет сложность O(n!);

е) 103n имеет сложность O(n2n).

3.Ход работы

Задания 1-3 выполняются с помощью программы с соответствующим номером, а результат их выполнения представлен в пункте 5.

Функция f(n) в ряде случаев может иметь достаточно сложную аналитическую форму. Поскольку для временной теоретической сложности большее значение имеет не столько вид функции, сколько порядок ее роста, то во многих математических дисциплинах, в том числе и в теории алгоритмов, функцию f(n) определяют как O(g(n)) и говорят, что она порядка g(n) для больших n, если

Где f(n) и g(n) – экспериментальная и теоретическая функции сложности. При этом если f(n) = O(g(n)), то предел отношения равен константе и тогда функция f(n) имеет порядок O(g(n)).

Тогда для части 1 задания 4 справедливы следующее временные функции сложности:

O(n!) = O(n!) либо

O((n+1)^2) = O(n^2)

O(n^n) = O(n^n)

O(2^(2n)) = O(2^(2n))

Часть 2

а) 3n5+10n3+n2+25 имеет сложность O(n)5;

Это отношение ложно. Так как

Правильный ответ: O()

б) 2n+3n имеет сложность O(2n);

Это отношение ложно. Так как

Правильный ответ: O()

в) 2n+3n имеет сложность O(3n);

Это отношение истинно. Так как

г) n! имеет сложность O(nn);

Это отношение ложно. Так как

Правильный ответ: O(n!) либо

д) nn имеет сложность O(n!);

Это отношение ложно. Так как

Правильный ответ: O(n^n)

е) 103n имеет сложность O(n2n).

Это отношение ложно. Так как

Правильный ответ: O(10^3n)

4.Тексты программ

Программа 1

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.util.\*;

import java.util.stream.Collectors;

import java.util.stream.IntStream;

public class Lab2Main {

public static void main(String[] args) {

System.out.println("============================== taskA ==================================");

new Lab2().taskA();

System.out.println("\n============================== taskB ==================================");

new Lab2().taskB();

}

static class Lab2 {

public void taskA() {

final String input1 = "taskA.txt";

List<Integer> list;

int n, m;

String line;

Expert[] experts;

ScaledAssessmentB[] b;

CoefficientB mainB;

try (Scanner input = new Scanner(new FileInputStream(input1))) {

n = input.nextInt();

m = input.nextInt();

input.nextLine();

experts = new Expert[m];

b = new ScaledAssessmentB[m];

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

experts[i] = new Expert();

do {

line = input.nextLine();

list = Arrays.stream(line.split(" "))

.map(Integer::parseInt)

.filter(a -> a != 0)

.collect(Collectors.toList());

if (!list.contains(-1))

experts[i].getCriterions().add(list);

} while (!list.contains(-1));

Rang r = new Rang(experts[i], n);

AssessmentC c = new AssessmentC(r);

b[i] = new ScaledAssessmentB(c);

System.out.println("expert №" + (i + 1));

r.formatPrintf(5, 3);

System.out.println();

c.formatPrintf(5, 3);

System.out.println();

b[i].formatPrintf(5, 3);

System.out.println("\n");

}

mainB = new CoefficientB(b);

mainB.formatPrintf(5, 3);

System.out.println("\n");

output(mainB);

} catch (FileNotFoundException e) {

e.printStackTrace();

}

}

private void output(CoefficientB mainB) {

System.out.println(IntStream.range(0, mainB.getValues().size())

.mapToObj(i -> Double.toString(Math.round((mainB.getValues().get(i) \* 1000)) / 1000d) + "q" + (i + 1))

.reduce("E = ", (s1, s2) -> s1 + (!s1.equals("E = ") ? " + " : "") + s2));

}

public void taskB() {

final String taskB = "taskB.txt";

Map<Character, Integer> comparison = new HashMap<>();

comparison.put('<', -1);

comparison.put('=', 0);

comparison.put('>', 1);

String restriction;

AssessmentC beg;

try (Scanner input = new Scanner(new FileInputStream(taskB))) {

int n = input.nextInt();

double[] estimate = new double[n];

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

estimate[i] = input.nextDouble();

}

beg = new AssessmentC(Arrays.stream(estimate).boxed().collect(Collectors.toList()));

input.nextLine();

restriction = input.nextLine();

restriction = restriction.replaceAll(" ", "");

double sum = estimate[n - 1] + estimate[n - 2];

for (int i = n - 3; i >= 0; i--) {

int comp = comparison.get(restriction.charAt(i));

if (Double.compare(estimate[i], sum) != comp) {

if (comp == 1) {

estimate[i] = sum + estimate[n - 1] / 10;

} else if (comp == -1) {

estimate[i] = sum - estimate[n - 1] / 10;

} else {

estimate[i] = sum;

}

rebalance(estimate, i, n);

}

sum += estimate[i];

}

AssessmentC c = new AssessmentC(Arrays.stream(estimate).boxed().collect(Collectors.toList()));

ScaledAssessmentB cb = new ScaledAssessmentB(c);

CoefficientB b = new CoefficientB(cb);

beg.formatPrintf(5, 3);

System.out.println();

System.out.println(restriction);

c.formatPrintf(5, 3);

System.out.println();

b.formatPrintf(5, 3);

System.out.println("\n");

output(b);

} catch (FileNotFoundException e) {

e.printStackTrace();

}

}

private void rebalance(double[] estimate, int i, int n) {

while (i != 0 && estimate[i - 1] < estimate[i]) {

estimate[i - 1] = estimate[i] + estimate[n - 1] / 10;

i--;

}

}

}

static class Expert {

protected List<List<Integer>> criterions = new ArrayList<>();

public List<List<Integer>> getCriterions() {

return criterions;

}

}

static class Rang {

protected List<Double> rangs;

public Rang(List<Double> rangs) {

this.rangs = rangs;

}

public Rang(Expert e, int n) {

rangs = new ArrayList<>(n);

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

rangs.add(0d);

}

double[] newNum;

double sum;

int num, k;

num = IntStream.range(0, e.getCriterions().size())

.map(i -> e.getCriterions().get(i).size())

.sum();

newNum = new double[num];

k = 1;

for (int i = 0; i < e.getCriterions().size(); i++) {

sum = 0d;

for (int j = 0; j < e.getCriterions().get(i).size(); j++) {

sum += k++;

}

sum /= e.getCriterions().get(i).size();

for (int j = 0; j < e.getCriterions().get(i).size(); j++) {

newNum[e.getCriterions().get(i).get(j) - 1] = sum;

}

}

for (int i = 0; i < e.getCriterions().size(); i++) {

for (int j = 0; j < e.getCriterions().get(i).size(); j++) {

rangs.set(e.getCriterions().get(i).get(j) - 1, newNum[e.getCriterions().get(i).get(j) - 1]);

}

}

}

public List<Double> getRangs() {

return rangs;

}

@Override

public String toString() {

return rangs.stream()

.map(d -> Double.toString(Math.round(d \* 10000) / 10000d))

.reduce("rij: ", (s1, s2) -> s1 + "\t" + s2);

}

public void formatPrintf(int all, int sign) {

String format = "rij: ";

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

format += "%" + all + "." + sign + "f ";

}

System.out.printf(format, rangs.toArray());

}

}

static class AssessmentC {

protected List<Double> values;

public AssessmentC(List<Double> values) {

this.values = values;

}

public AssessmentC(Rang rang) {

int n = rang.getRangs().size();

values = rang.getRangs().stream()

.map(r -> 1 - (r - 1d) / n)

.collect(Collectors.toList());

}

public List<Double> getValues() {

return values;

}

public double getSum() {

return values.stream().reduce(0d, (a, b) -> a + b);

}

@Override

public String toString() {

return values.stream()

.map(d -> Double.toString(Math.round(d \* 10000) / 10000d))

.reduce("cij: ", (s1, s2) -> s1 + "\t" + s2);

}

public void formatPrintf(int all, int sign) {

String format = "cij: ";

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

format += "%" + all + "." + sign + "f ";

}

System.out.printf(format, values.toArray());

}

}

static class ScaledAssessmentB {

protected List<Double> values;

public ScaledAssessmentB(List<Double> values) {

this.values = values;

}

public ScaledAssessmentB(AssessmentC assessmentC) {

double sum = assessmentC.getSum();

values = assessmentC.getValues().stream()

.map(c -> c / sum)

.collect(Collectors.toList());

}

public List<Double> getValues() {

return values;

}

@Override

public String toString() {

return values.stream()

.map(d -> Double.toString(Math.round(d \* 10000) / 10000d))

.reduce("bij: ", (s1, s2) -> s1 + "\t" + s2);

}

public void formatPrintf(int all, int sign) {

String format = "bij: ";

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

format += "%" + all + "." + sign + "f ";

}

System.out.printf(format, values.toArray());

}

}

static class CoefficientB {

protected List<Double> values;

public CoefficientB(List<Double> values) {

this.values = values;

}

public CoefficientB(ScaledAssessmentB... scaledAssessmentB) {

int m = scaledAssessmentB.length;

List<Double> reduced = new ArrayList<>();

for (int i = 0; i < scaledAssessmentB[0].values.size(); i++) {

reduced.add(0d);

}

for (int i = 0; i < scaledAssessmentB[0].getValues().size(); i++) {

for (int j = 0; j < m; j++) {

reduced.set(i, reduced.get(i) + scaledAssessmentB[j].getValues().get(i));

}

}

values = reduced.stream()

.map(d -> d / m)

.collect(Collectors.toList());

}

public List<Double> getValues() {

return values;

}

public void formatPrintf(int all, int sign) {

String format = "bi: ";

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

format += "%" + all + "." + sign + "f ";

}

System.out.printf(format, values.toArray());

}

}

}

Программа 2

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.util.Scanner;

import java.util.stream.IntStream;

public class Lab3 {

protected boolean[][] a;

protected int size;

protected boolean oriented;

protected int[][] d;

public Lab3(boolean[][] a, int size) {

this.a = a;

this.size = size;

}

public Lab3(Scanner input) {

size = input.nextInt();

oriented = input.nextBoolean();

a = new boolean[size][size];

d = new int[size][size];

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

for (int j = 0; j < size; j++) {

a[i][j] = (input.nextInt() == 1);

}

}

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

for (int j = 0; j < size; j++) {

d[i][j] = input.nextInt();

}

}

}

public static void main(String[] args) {

final String inputName = "lab3.txt";

try (Scanner input = new Scanner(new FileInputStream(inputName))) {

Lab3 lab = new Lab3(input);

lab.analysis();

} catch (FileNotFoundException e) {

e.printStackTrace();

}

}

public void analysis() {

// int R = R();

System.out.println("-- уЧСЪБООПУФШ УФТХЛТХТЩ:");

System.out.println("УЙУФЕНБ " + (R() >= size - 1 ? "УЧСЪБОБС" : "ОЕУЧСЪБОБС"));

System.out.println();

int alpha = alpha();

System.out.println("-- уФТХЛФХТОБС ЙЪВЩФПЮОПУФШ:");

System.out.println(

(alpha == 0 ? "НЙОЙНБМШОБС"

: (alpha > 0 ? "НБЛУЙНБМШОБС"

: "УЙУФЕНБ ОЕУЧСЪОБС")));

System.out.println();

double delta = delta();

System.out.println("-- уФЕРЕОШ ГЕОФТБМЙЪБГЙЙ:");

System.out.println("УЙУФЕНБ " +

(delta == 1 ? "ГЕОФТБМЙЪПЧБООБС"

: (delta == 0 ? "РПМОПУФША ДЕГЕОФТБМЙЪПЧБООБС"

: "УНЕЫБООБС")));

System.out.println();

}

public int R() {

int sum = 0;

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

for (int j = 0; j < size; j++) {

sum += a[i][j] ? 1 : 0;

}

}

return (oriented ? sum : sum / 2);

}

public int alpha() {

return alpha(R());

}

protected int alpha(int R) {

return R / (size - 1) - 1;

}

public int Q() {

int sum = 0;

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

for (int j = 0; j < size; j++) {

if (i != j)

sum += d[i][j];

}

}

return sum;

}

public double delta() {

double delta = 0;

if (oriented) {

int k;

int[] V = allV();

k = IntStream.range(0, size)

.reduce((a, b) -> (V(b) > V(a) ? b : a))

.orElse(0);

delta = (1d / ((size - 1) \* (V(k) - 1))) \* sum(V, k);

} else throw new UnsupportedOperationException("unsupported for unoriented graph");

return delta;

}

protected int[] allV() {

int[] v = new int[size];

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

v[i] = V(i);

}

return v;

}

protected int V(int i) {

int v = 0;

for (int j = 0; j < size; j++) {

v += (a[i][j] ? 1 : 0) + (a[j][i] ? 1 : 0);

}

v -= a[i][i] ? 1 : 0;

return v;

}

private int sum(int[] v, int k) {

int sum = 0;

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

sum += v[k] - v[i];

}

return sum;

}

}

Программа 3

clc;

clear;

syms x f x1 f1 x2 f2 x3 f3 p p1 p2 p3;

%x0 = 1;

%f0 = 2;

%F = 2\*x\*f1 + 4\*x^2\*f + 2\*x3 + 0.1\*f\*x1;

x0 = 2;

f0 = 2;

F = 2\*x\*f+0.3\*x^2\*f2+0.5\*x\*f1+0.1\*f3\*x+0.2\*x1\*f;

X = [x3 x2 x1 x];

Y = [f3 f2 f1 f];

P = [p^3 p^2 p^1 p^0];

S = [X Y];

Value = [0 0 0 x0 0 0 0 f0];

FF(1:length(X)) = F;

derivativesX = arrayfun(@(x, y) diff(x, y), FF, X, 'UniformOutput',false);

derivativesY = arrayfun(@(x, y) diff(x, y), FF, Y, 'UniformOutput',false);

linX = subs(derivativesX, S, Value);

linY = subs(derivativesY, S, Value);

xS = collect(sum(linX.\*X), X);

yS = collect(sum(linY.\*Y), Y);

WS = yS / xS;

PHS = simplify(WS/(1+WS));

Dp = collect(sum(linX.\*P),P);

PHp = Dp / collect(sum(linY.\*P),P);

n = length(linX) - 1;

for i=1:length(linX)

if linX(i) == 0

n = n - 1;

else break;

end

end

A = [zeros(1, n-1) linX(end:-1:1) zeros(1, n-2)];

G = zeros(n, n);

Mnrs = zeros(n, 1);

for i = 1:n

G(:, i) = A((n - i)\*2 + 1:3\*n - 2\*i);

end

for i = 1:n

Mnrs(i) = det(G(1:i,1:i));

end

Ust = 1;

%if any([linX(:); Mnrs(:)] <= 0)

if any(Mnrs(:) <= 0)

Ust = 0;

end

fprintf('-- Уравнение системы:\n');

fprintf('F(x, f, x1, x2, x3, f1, f2, f3) = \n');

pretty(vpa(F))

fprintf('-- Коэффициенты разложения в ряд Тейлора по х3, х2... :\n');

disp(linX);

fprintf('-- Коэффициенты разложения в ряд Тейлора по f3, f2... :\n');

disp(linY);

fprintf('-- Линеаризованная форма исходного нелинейного уравнения:\n');

pretty(vpa(xS))

fprintf(' = ');

pretty(vpa(-yS))

fprintf('-- Передаточная функция системы:\n');

pretty(vpa(WS));

fprintf('-- Передаточная функция замкнутой системы по задающему воздействию:\n');

pretty(vpa(PHp));

fprintf('-- Характеристический полином системы:\n');

pretty(vpa(Dp));

fprintf('-- n = ');

disp(n);

fprintf('-- Матрица Гурвица %dx%d:\n',n,n);

disp(vpa(G,3));

fprintf('-- Определители матрицы Гурвица:\n');

disp(Mnrs);

fprintf('-- Устойчивость системы:\n');

if Ust == 1

disp('система устойчивая');

else

disp('система неустойчивая');

end

fprintf('\n=================== Задание 2 ===============\n');

syms t;

T = [t^2 t 1];

B = [35 0];

D = [2.5 0.75 1];

M = [61 7 0.7];

n = 1:(length(T));

Wp = (B(1)\*p+B(2))/(p\*(D(1)\*p^2+D(2)\*p+D(3)));

gt = collect(sum(M .\* T), T);

%n = 1:2;

%Wp = 100/(0.7\*p^2+0.1\*p+0.4);

%gt = 10\*t+1;

Phxp = simplify(1 / (1 + Wp));

GT(n) = gt;

PHxp(n) = Phxp;

derivativesG = arrayfun(@(x, y) diff(x, t, y), GT, n-1, 'UniformOutput', false);

derivativesPhxp = arrayfun(@(x, y) diff(x, p, y), PHxp, n-1, 'UniformOutput', false);

C = subs(derivativesPhxp, p, 0);

xt = collect(sum((C./factorial(0:length(C)-1)).\*derivativesG));

fprintf('-- Передаточная функция разомкнутой системы:\n');

fprintf('W(p) = \n');

pretty(vpa(Wp, 4));

fprintf('-- Закон зменения входного воздействия:\n');

fprintf('g(t) = \n');

pretty(vpa(gt, 4));

fprintf('-- Передаточная функция замкнутой системы по ошибке:\n');

fprintf('Фх(p) = \n');

pretty(vpa(Phxp, 4));

fprintf('-- Производные от g(t):\n');

pretty(vpa(derivativesG', 4));

fprintf('-- Коэффициенты ошибок:\n');

disp(vpa(flipud(C), 4));

fprintf('-- Закно изменения во времени ошибки:\n');

fprintf('x(t) = \n');

pretty(vpa(xt, 4));

5.Выполнение

На рисунке 1 показан результат работы и промежуточные вычисления первого задания.

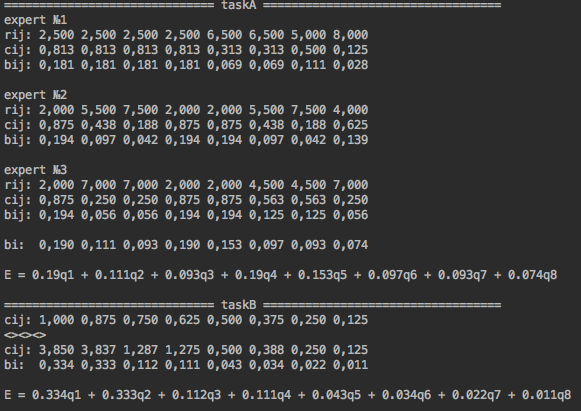


Рисунок 1 – Нахождение СТК и промежуточные вычисления

На рисунке 2 показан результат анализа системы представленной в виде графа.

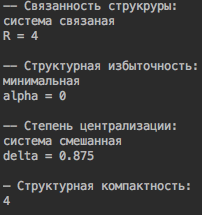


Рисунок 2 – Оценка системы

На рисунке 3 показана результат работы и промежуточные вычисления программы №3.

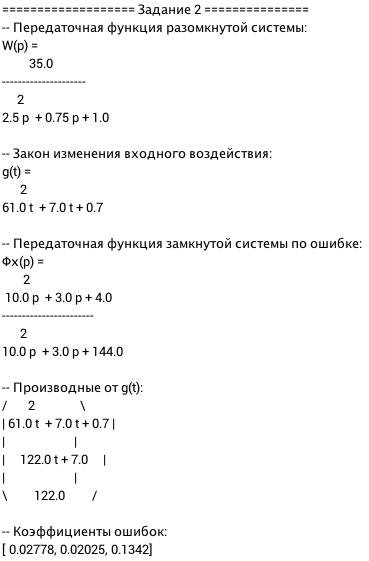
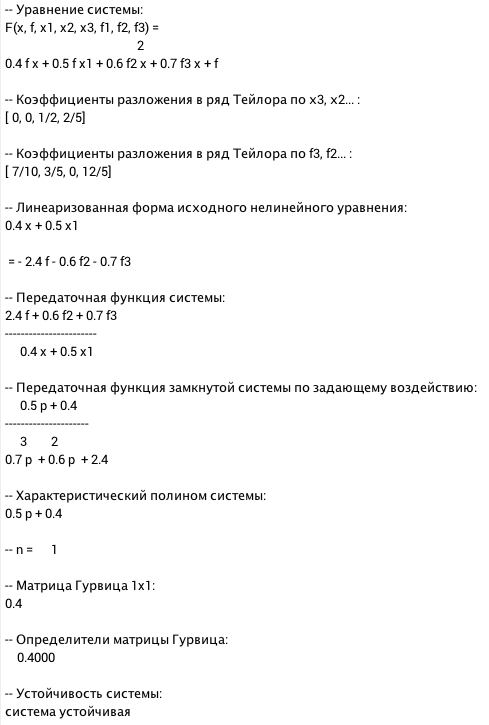


Рисунок 3 – Результат программы 3

Выводы

В ходе лабораторной работы я построил обобщенный критерий эффективности системотехнических комплексов (СТК). Определил количественные оценки графа для сравнения различных вариантов построения систем. Линеаризовал уравнения замкнутой системы заданной в нелинейном виде и проанализировал устойчивость линейных систем. Изучил понятия сложности системной задачи, спектры сложности, трансвычислительная сложность и оценил алгоритмическую разрешимость системной задачи.