МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Балтийский государственный технический университет «ВОЕНМЕХ» им. Д.Ф. Устинова»
(БГТУ «ВОЕНМЕХ» им. Д.Ф. Устинова»)

Факультет И — Информационные и управляющие системы
(шифр / Наименование)
Кафедра И9 — Систем управления и компьютерных технологий
(шифр / наименование)
Дисциплина Моделирование систем

Лабораторная работа №1
на тему «Программная реализация имитационной модели нелинейной динамической системы»
Вариант №3

Выполнил студент группы И967
Васильев Н.А
Фамилия И.О.

ПРЕПОДАВАТЕЛЬ
Захаров А.Ю.
Фамилия И.О. Подпись

«_____» _____ 2019 г.

САНКТ-ПЕТЕРБУРГ
2019 г.

Задание:

В соответствии с индивидуальным вариантом задания (табл. 1–5) разработать и отладить программное приложение, обеспечивающее:

1. Решение системы дифференциальных уравнений на интервале [0; T] для T = 10 с с любым шагом, задаваемым пользователем в пределах (0; T). Для демонстрации результатов обеспечить вывод графиков $x_i(t)$, i=1, 2,…, n; значения указанной в задании переменной состояния в конце интервала интегрирования $x_k(T)$ и значения относительной погрешности его определения δ.

2. Анализ зависимости точности и трудоемкости решения задачи от шага интегрирования. Вывод графиков зависимостей относительной погрешности δ и оценки трудоемкости от величины шага h.

3. Автоматический выбор величины шага интегрирования для достижения относительной погрешности не более 1% с выводом итоговых результатов, перечисленных в п. 1, для найденного шага.

Вариант задания:

Модель 1 – система уравнений 5-го порядка

$$\dot{x}_1 = -g\sin x_2 + \frac{p - ac_x x_1^2}{m - ut};$$

$$\dot{x}_2 = \frac{-g + \dfrac{p\cdot\sin(x_5 - x_2) + ac_y x_1^2}{m - ut}}{x_1};$$

$$\dot{x}_3 = \frac{m_1 a(x_2 - x_5)x_1^2 - m_2 a x_1^2 x_3}{m - ut};$$

$$\dot{x}_4 = x_1 \sin x_2;$$

$$\dot{x}_5 = x_3.$$

Варианты исходных данных для модели 1

№	Значения постоянных параметров модели									Начальные значения переменных состояния				
	p	a	m	u	c_x	c_y	m_1	m_2	T	$x_1(0)$	$x_2(0)$	$x_3(0)$	$x_4(0)$	$x_5(0)$
3	10^5	0,5	2000	20	0,03	0,002	0,05	0,01	12	1800	0,8	0	0	0,8

Текст программы:

```
import React, { useState } from
"react";
import "../App.css";

import { Line } from "react-chartjs-2";

/* Модель 1 вариант 3 */

const Lab1 = () => {
  const [defaultH, setH] =
useState(0.1);
  const [initialXValues,
setInitialXValues] = useState({
    x1: 1800,
    x2: 0.8,
    x3: 0,
    x4: 0,
    x5: 0.8
  });
  const [graphsData, setGraphData] =
useState({
    time: [],
    xArrays: [],
    hArray: [],
    deltaArray: [],
    numberOfStepsArray: []
  });

  const [constants, setConstants] =
useState({
    p: 100000,
    a: 0.5,
    m: 2000,
    u: 20,
    cx: 0.03,
    cy: 0.002,
    m1: 0.05,
    m2: 0.01,
    T: 12,
    g: 9.81
  });

  const { p, a, m, u, cx, cy, m1, m2, T, g
} = constants;

  const Euler = step => {
    const time = [0];
    let { x1, x2, x3, x4, x5 } = {
...initialXValues };

    const xArrays = {
      x1: [x1],
      x2: [x2],
      x3: [x3],
      x4: [x4],
      x5: [x5]
    };

    let numberOfSteps = 0;
    for (let i = 0; i <= T; i += step) {
      numberOfSteps += 1;
    }

    for (let i = 1; i < numberOfSteps;
i++) {
      const t = i * step;

      x1 += step * (-g * Math.sin(x2) +
(p - a * cx * x1 ** 2) / (m - u * t));
      x2 +=
        step *
        ((-g + (p * Math.sin(x5 - x2) + a *
cy * x1 ** 2) / (m - u * t)) / x1);
      x3 +=
        step *
        ((m1 * a * (x2 - x5) * x1 ** 2 -
m2 * a * x1 ** 2 * x3) / (m - u * t));
      x4 += step * (x1 * Math.sin(x2));
      x5 += step * x3;

      xArrays.x1.push(x1);
      xArrays.x2.push(x2);
      xArrays.x3.push(x3);
      xArrays.x4.push(x4);
      xArrays.x5.push(x5);
```

```
      time.push(t);
    }
    const lastValues = { x1, x2, x3, x4,
x5 };

    return { xArrays, time,
numberOfSteps, lastValues };
  };

  const calculate = (optimal = false) =>
{
    let delta = 100;
    const deltaArray = [100];

    let h = defaultH;
    const hArray = [defaultH];
    let temp = 0;
    for (let i = 0; i < T + h; i += h) {
      temp += 1;
    }

    let time,
      xArrays,
      numberOfStepsArray = [temp];

    if (optimal) {
      while (delta > 0.001) {
        const { lastValues } = Euler(h);

        const {
          xArrays: xArraysWithLowerH,
          lastValues:
lastValuesWithLowerH,
          time: newTime,
          numberOfSteps
        } = Euler(h / 2);

numberOfStepsArray.push(numberOfS
teps);
        time = newTime;

        xArrays = xArraysWithLowerH;
        delta = Math.abs(
          (lastValues.x4 -
lastValuesWithLowerH.x4) /
lastValuesWithLowerH.x4
        );

        h = h / 2;
        hArray.push(h);
        deltaArray.push(delta);
      }
    } else {
      const withoutOptimization =
Euler(h);
      xArrays =
withoutOptimization.xArrays;
      time = withoutOptimization.time;
      numberOfStepsArray =
[withoutOptimization.numberOfSteps];
    }

    return { xArrays, time,
numberOfStepsArray, hArray,
deltaArray, h };
  };

  const {
    time,
    xArrays,
    hArray,
    deltaArray = [],
    numberOfStepsArray
  } = graphsData;

  const data1 = {
    xLabel: "Время",
    yLabel: "x1",
    data: time.map((key, index) => ({ x:
key, y: xArrays.x1[index] }))
  };
  const data2 = {
    xLabel: "Время",
    yLabel: "x2",
    data: time.map((key, index) => ({ x:
key, y: xArrays.x2[index] }))
  };
```

```
  const data3 = {
    xLabel: "Время",
    yLabel: "x3",
    data: time.map((key, index) => ({ x:
key, y: xArrays.x3[index] }))
  };
  const data4 = {
    xLabel: "Время",
    yLabel: "x4",
    data: time.map((key, index) => ({ x:
key, y: xArrays.x4[index] }))
  };
  const data5 = {
    xLabel: "Время",
    yLabel: "x5",
    data: time.map((key, index) => ({ x:
key, y: xArrays.x5[index] }))
  };

  const dataDelta = {
    xLabel: "Шаг",
    yLabel: "delta",
    data: hArray.map((key, index) => ({
x: key, y: deltaArray[index] }))
  };
  const dataSteps = {
    xLabel: "Шаг",
    yLabel: "Количество шагов",
    data: hArray.map((key, index) => ({
x: key, y: numberOfStepsArray[index]
}))
  };

  const data = [data1, data2, data3,
data4, data5, dataDelta, dataSteps];

  console.log(constants, graphsData,
defaultH);
  return (
    <>
      <header>
        <div
className="parametersWrapper">
          <h2>Значения
коэффициентов</h2>
          <ul className="list">

{Object.entries(constants).map(([key,
value], index) => {
              return (
                <li key={index}>
                  <label>
                    <span
className="inputLabel">{key}</span
>
                    <input
                      defaultValue={value}
                      onChange={e => {
                        setConstants({
                          ...constants,
                          [key]: +e.target.value
                        });
                      }}
                    />
                  </label>
                </li>
              );
            })}
          </ul>
        </div>
        <div
className="parametersWrapper">
          <h2>Начальные значения</h2>
          <ul className="list">

{Object.entries(initialXValues).map(([
key, value], index) => {
              return (
                <li key={index}>
                  <label>
                    <span
className="inputLabel">
                      x<sub>{key}</sub>
                    </span>
                    <input
                      defaultValue={value}
                      onChange={e => {
```

```
                    setInitialXValues({
                      ...initialXValues,
                      [key]: +e.target.value
                    });
                  }}
                />
              </label>
            </li>
          );
        })}
      </ul>
    </div>
  </header>
  <div>
    <input
      type="button"
      onClick={() =>
setGraphData(calculate(false))}
      value="Посчитать с заданным
шагом"
    />
    <input
      defaultValue={defaultH}
      onChange={e => {
        setH(+e.target.value);
      }}
    />
  </div>
  <input
    type="button"
    onClick={() =>
setGraphData(calculate(true))}
    value="Посчитать с точностью
1%"
  />

  <div className="gridContainer">
    {data.map((item, key) => (
      <div key={key}>
        <Line
          options={{
            scales: {
              xAxes: [
                {
                  type: "linear",
                  scaleLabel: {
                    display: true,
                    labelString: item.xLabel
                  }
                }
              ],
              yAxes: [
                {
                  scaleLabel: {
                    display: true,
                    labelString: item.yLabel
                  }
                }
              ]
            },
            legend: {
              display: false
            }
          }}
          data={{
            datasets: [
              {
                data: item.data,
                fill: false,
                borderColor: "blue"
              }
            ]
          }}
        />
      </div>
    ))}
  </div>
 </>
 );
};

export default Lab1;
```

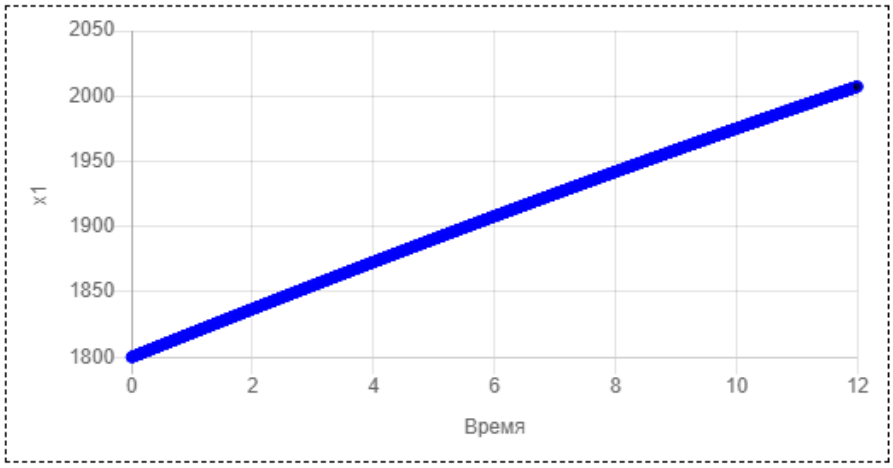
Результат работы программы:

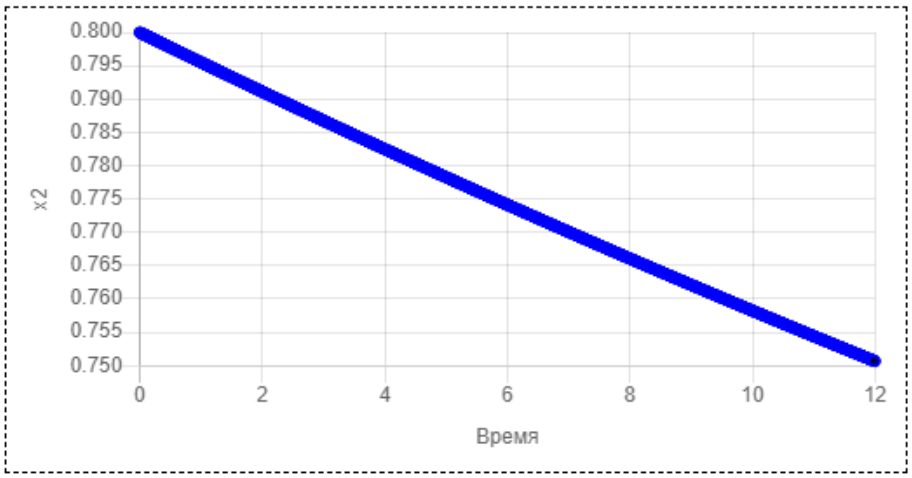
Значения коэффициентов **Начальные значения**

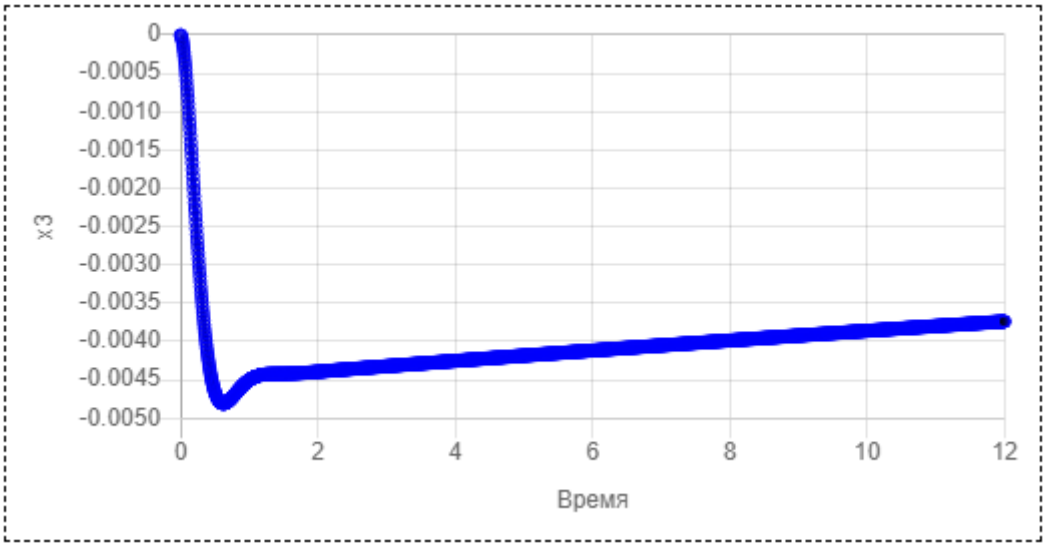
p	100000	x_{x1}	1800
a	0.5	x_{x2}	0.8
m	2000	x_{x3}	0
u	20	x_{x4}	0
cx	0.03	x_{x5}	0.8
cy	0.002		
m1	0.05		
m2	0.01		
T	12		
g	9.81		

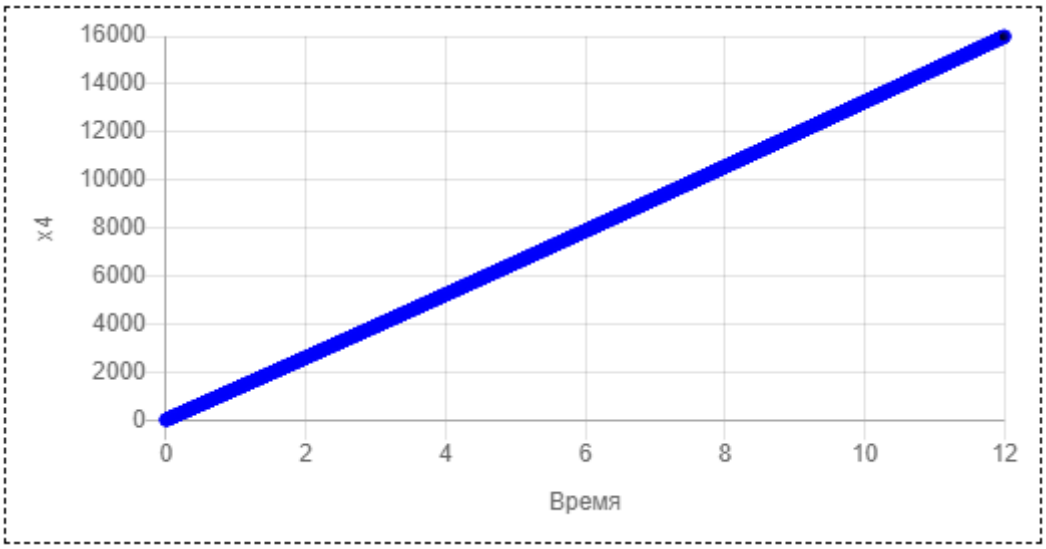
Посчитать с заданным шагом 0.1

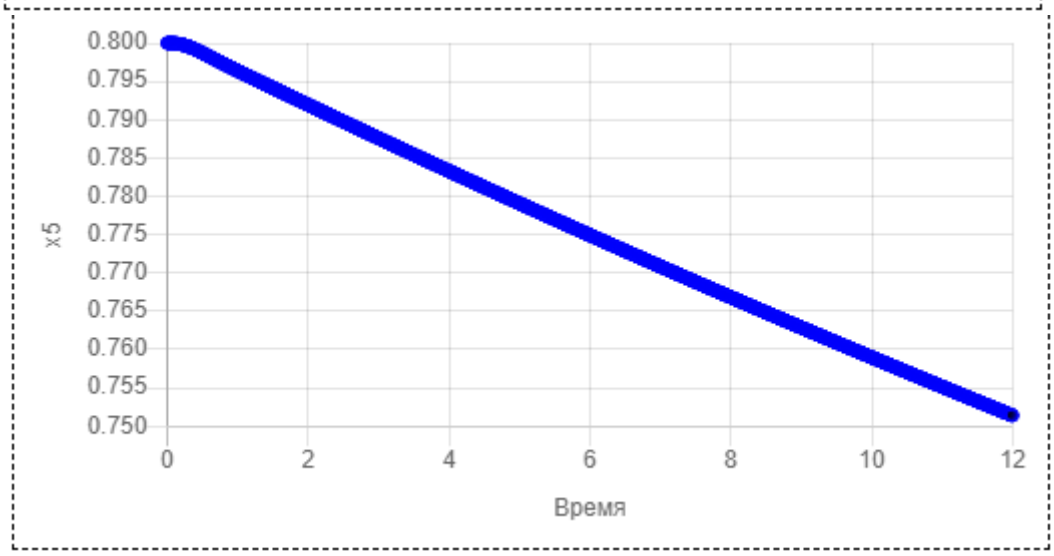
Посчитать с точностью 1%

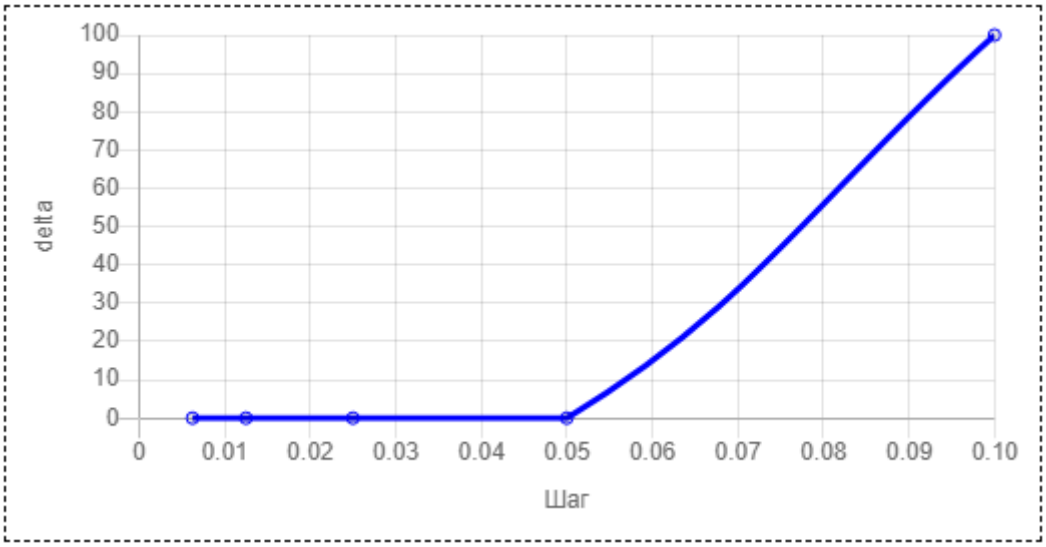


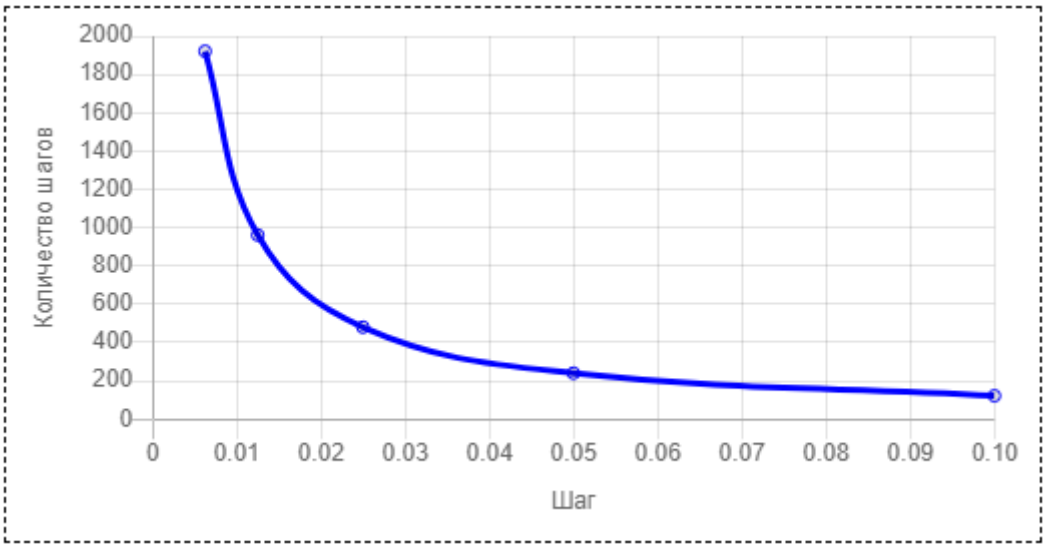



0
-0.0005
-0.0010
-0.0015
-0.0020
-0.0025
-0.0030
-0.0035
-0.0040
-0.0045
-0.0050
x3
0
2
4
6
8
10
12
Время


16000
14000
12000
10000
8000
6000
4000
2000
0
x4
0
2
4
6
8
10
12
Время


0.800
0.795
0.790
0.785
0.780
0.775
0.770
0.765
0.760
0.755
0.750
x5
0
2
4
6
8
10
12
Время


delta
100
90
80
70
60
50
40
30
20
10
0
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.10
Шаг


Количество шагов
2000
1800
1600
1400
1200
1000
800
600
400
200
0
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.10
Шаг