|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *voenmeh* | 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 | | |

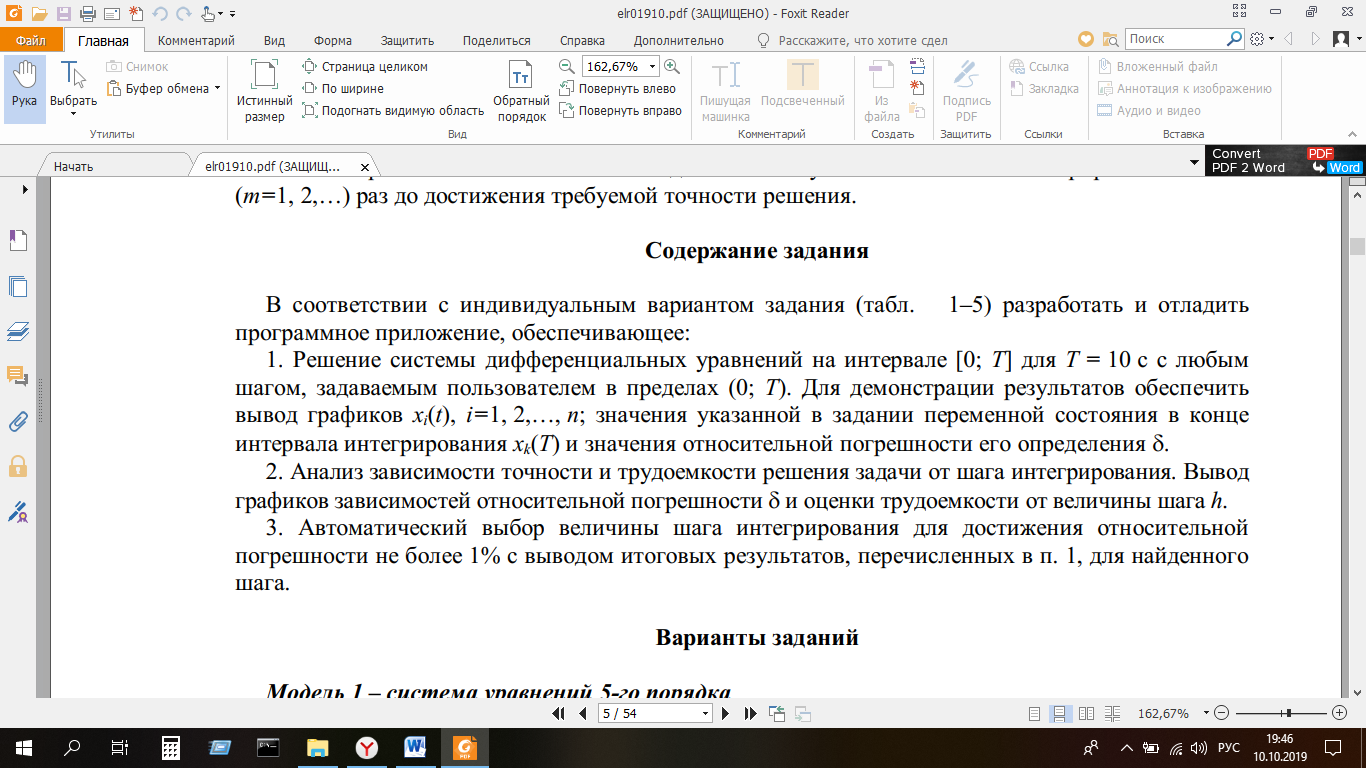
|  |
| --- |
| Лабораторная работа №1 |
| на тему «Программная реализация имитационной |
| модели нелинейной динамической системы» |
| Вариант №3 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил студент группы | | | |  | И967 |
| Васильев Н.А | | | | | |
| Фамилия И.О. | | | | | |
| **ПРЕПОДАВАТЕЛЬ** | | | | | |
| Захаров А.Ю. | |  |  | | |
| Фамилия И.О. Подпись | | | | | |
| «\_\_\_\_\_» |  | | |  | 2019 г. |

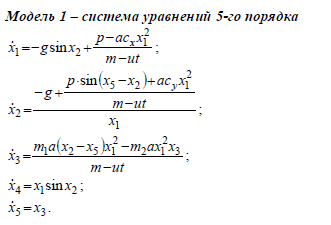
САНКТ-ПЕТЕРБУРГ

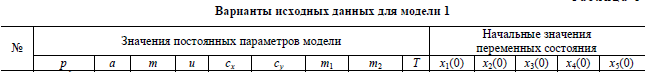
2019 г.

Задание:



Вариант задания:







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(numberOfSteps);

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:

