МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А.И. ГЕРЦЕНА»

Направление подготовки
09.03.01 – Информатика и вычислительная техника

Профиль «Технологии разработки программного обеспечения»

Лабораторная работа № 3

Работу выполнили:
Балаев Жамал,
Васильева Марина,
Иванов Никита,
Шардт Максим
Рожков Максим
очная форма обучения
курс: 2; группа: ИВТ-1.1

Научный руководитель:
Профессор Власова Елена Зотиковна

Санкт-Петербург
2022

Численные методы решения дифференциальных уравнений

Выполнил Балаев Ж.Б. ИВТ 1.1

Постановка задачи
Разработать программы решения дифференциальных уравнений с использованием численных методов Эйлера и Рунге-Кутта.

Часть 1

Был разработан программный модуль для дифференциальных уравнений первого порядка методами Эйлера и Рунге-Кутта:

Пользователь может выбрать пределы интегрирования и начальные значения для x и y, а также указать количество разбиений и выбрать метод вычислений.

Анализ результатов вычислений:

	Значение Y	
Значение X	**Метод Эйлера**	**Метод Рунге-Кутта**
0	1	1
0.1	1.1	1.09965878
0.2	1.199	1.197217232
0.3	1.29492	1.290461449
0.4	1.3855644	1.377127562
0.5	1.468698264	1.454991191

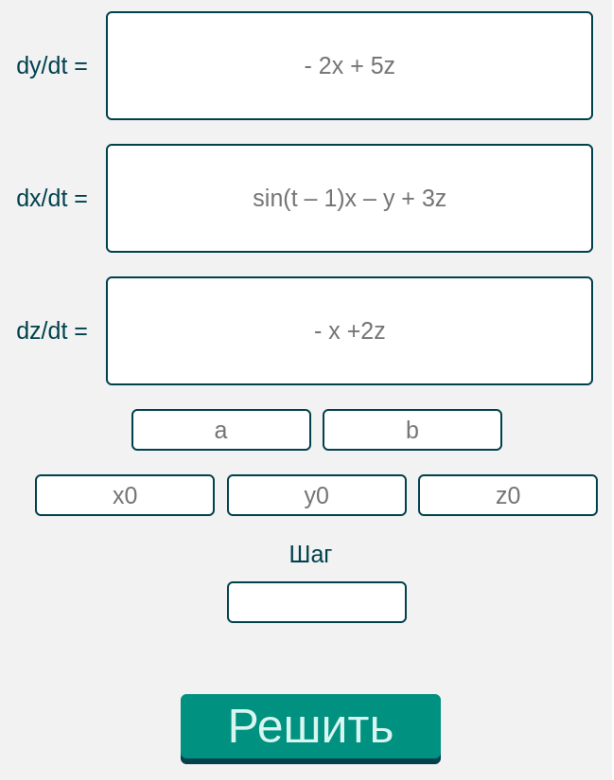
0.6	1.542133177	1.521961316
0.7	1.603818504	1.576173133
0.8	1.651933059	1.616074144
0.9	1.684971721	1.640497977
1	1.701821438	1.648721007

Вывод:
Среднеквадратическое отклонение результатов равно 0.2341, что достаточно невелико. С повышением количества разбиений повышается точность метода Эйлера, но она всегда ниже точности метода Рунге-Кутта. Максимально схожие значения получаются при выборе *одиннадцати* разбиений.

Часть 2

Был разработан программный модуль для дифференциальных уравнений второго порядка методами Эйлера и Рунге-Кутта:

а также модуль для решение дифференциальных уравнений:



В первом модуле пользователь может выбрать пределы интегрирования и начальные значения для x, y, z, а также указать количество разбиений и выбрать метод вычислений.

Во втором пользователь может указать уравнения дифференциальных уравнений, а также пределы и начальные значения.

Анализ результатов вычислений

1. Таблица результатов вычисления дифференциального уравнения второго порядка:

Значение X	Метод Эйлера	Метод Рунге-Кутта
0	0.77	0.77
0.1	0.7874608426	0.748
0.2	0.8076288336	0.726825
0.3	0.8306049965	0.7065116667
0.4	0.8565043671	0.6870920625
0.5	0.8854558857	0.6685944069
0.6	0.9176024328	0.6510437458
0.7	0.953100991	0.6344625442
0.8	0.9921229168	0.6188712135
0.9	1.034854312	0.6042885825
1	1.075991491	0.5985825523

Вывод:

Среднеквадратическое отклонение результатов равно 0.2472804602 что сравнимо с значением, полученным в предыдущей части. Также, с повышением количества разбиений повышается точность метода Эйлера, но она всегда ниже точности метода Рунге-Кутта. Максимально схожие значения получаются при выборе *семнадцати* разбиений.

2. Таблица результатов вычисления дифференциальной системы уравнений:

Значение X	Значение Y
2	1
2.006	1
2.009	0.999991
2.012	0.999973
2.015	0.9999460001
2.018	0.9999100004
2.021	0.9998650012

2.024	0.9998110028
2.026	0.9997480057
2.029	0.9996760102
2.032	0.999595017
2.035	0.9995050267
2.038	0.9994060401
2.041	0.9992980579
2.043	0.9991810811
2.046	0.9990551106
2.049	0.9989201474
2.052	0.9987761928
2.054	0.9986232478
2.057	0.9984613139
2.06	0.9982903924
2.062	0.9981104847
2.065	0.9979215925
2.067	0.9977237172
2.07	0.9975168606
2.073	0.9973010245
2.075	0.9970762108
2.078	0.9968424213
2.08	0.9965996582
2.083	0.9963479235
2.085	0.9960872194
2.088	0.9958175481
2.09	0.9955389121
2.092	0.9952513137
2.095	0.9949547555
2.097	0.99464924
2.099	0.9943347699
2.102	0.994011348
2.104	0.993678977
2.106	0.99333766

2.109	0.9929873998
2.111	0.9926281996
2.113	0.9922600625
2.115	0.9918829918
2.118	0.9914969907
2.12	0.9911020627
2.122	0.9906982112
2.124	0.9902854397
2.126	0.989863752
2.128	0.9894331518
2.13	0.9889936427
2.133	0.9885452288
2.135	0.9880879139
2.137	0.9876217021
2.139	0.9871465975
2.141	0.9866626043
2.143	0.9861697268
2.145	0.9856679694
2.146	0.9851573364
2.148	0.9846378324
2.15	0.984109462
2.152	0.9835722298
2.154	0.9830261407
2.156	0.9824711994
2.158	0.9819074108
2.159	0.9813347801
2.161	0.9807533121
2.163	0.9801630122
2.165	0.9795638854
2.166	0.9789559372
2.168	0.978339173
2.17	0.9777135981
2.172	0.9770792182

2.173	0.9764360388
2.175	0.9757840658
2.176	0.9751233048
2.178	0.9744537617
2.18	0.9737754426
2.181	0.9730883533
2.183	0.9723925001
2.184	0.9716878891
2.186	0.9709745266
2.187	0.9702524188
2.189	0.9695215723
2.19	0.9687819935
2.191	0.9680336891
2.193	0.9672766656
2.194	0.9665109297
2.196	0.9657364884
2.197	0.9649533485
2.198	0.964161517
2.199	0.9633610009
2.201	0.9625518073
2.202	0.9617339435
2.203	0.9609074167
2.204	0.9600722344
2.206	0.9592284038
2.207	0.9583759326
2.208	0.9575148284
2.209	0.9566450987

Вывод

В ходе лабораторной работы были разработаны программы для решения дифференциальных уравнений первого и второго порядка, а также для системы

дифференциальных уравнений, с использованием численных методов Эйлера и Рунге-Кутта.

Данные, полученные в ходе работы программы являются достаточно точными, среднеквадратическое отклонение которых равно 0.2341 для дифференциальных уравнений первого порядка и 0.2472804602 для дифференциальных уравнений второго порядка. С повышением количества разбиений повышается точность метода Эйлера, но она всегда ниже точности метода Рунге-Кутта.

Максимально схожие значения получаются при выборе одиннадцати разбиений и семнадцати разбиений для дифференциальных уравнений первого и второго порядка соответственно

Приложение 1.

```
export function EulerMethodI(
  a: number,
  b: number,
  x0: number,
  y0: number,
  n: number,
  fn: (x: number, y: number) => number
) {
  const resX = [x0],
    resY = [y0];
  const h = (b - a) / n;

  for (let x = x0, y = y0; x < b - h; x += h) {
    y += h * fn(x, y);
    resX.push(Math.round((x + h) * 100) / 100);
    resY.push(y);
  }

  return { x: resX, y: resY };
}

export function RungeKuttaMethodI(
  a: number,
  b: number,
  x0: number,
  y0: number,
  n: number,
  fn: (x: number, y: number) => number
) {
  const resX = [x0],
    resY = [y0];
  const h = (b - a) / n;
```

```
  for (let x = x0, y = y0; x < b - h; x += h) {
    const k1 = h * fn(x, y);
    const k2 = h * fn(x + h / 2, y + k1 / 2);
    const k3 = h * fn(x + h / 2, y + k2 / 2);
    const k4 = h * fn(x + h, y + k3);
    y += (k1 + 2 * k2 + 2 * k3 + k4) / 6;

    resX.push(Math.round((x + h) * 100) / n);
    resY.push(y);
  }

  return { x: resX, y: resY };
}

export function EulerMethodII(
  a: number,
  b: number,
  x0: number,
  y0: number,
  z0: number,
  n: number,
  fn: (x: number, y: number, z: number) => number
) {
  const resX = [x0],
    resY = [y0],
    resZ = [z0];
  const h = (b - a) / n;
  for (let x = x0, y = y0, z = z0; x < b - h; x += h) {
    y = y + h * z;
    z = z + h * fn(x, y0, z);
    y0 = y;
    resX.push(x + h);
    resY.push(y);
    resZ.push(z);
  }
  return { x: resX, y: resY, z: resZ };
}

export function RungeKuttaMethodII(
  a: number,
  b: number,
  x0: number,
  y0: number,
  z0: number,
  n: number,
  fn: (x: number, y: number, z: number) => number
) {
```

```
  const resX = [x0],
    resY = [y0],
    resZ = [z0];
  const h = (b - a) / n;
  for (let x = x0, y = y0, z = z0; x < b - h; x += h) {
    const k1 = h * fn(x, y, z);
    const k2 = h * fn(x + h / 2, y + k1 / 2, z);
    const k3 = h * fn(x + h / 2, y + k2 / 2, z);
    const k4 = h * fn(x + h, y + k3, z);
    y = y + (k1 + 2 * k2 + 2 * k3 + k4) / 6;
    z = z + h * fn(x, y0, z);
    y0 = y;
    resX.push(x + h);
    resY.push(y);
    resZ.push(z);
  }
  return { x: resX, y: resY, z: resZ };
}

export function EulerMethodIII(
  a: number,
  b: number,
  x0: number,
  y0: number,
  z0: number,
  n: number
) {
  const resX = [x0],
    resY = [y0],
    resZ = [z0],
    resT = [a];
  const h = 0.003;
  for (let x = x0, y = y0, z = z0; a < b - h; a += h) {
    x = x0 + h * (-2 * x0 + 5 * z0);
    y = y0 + h * (Math.sin(a - 1) * x0 - y0 + 3 * z0);
    z = z0 + h * (-x0 + 2 * z0);
    x0 = x;
    y0 = y;
    z0 = z;
    resX.push(+(x + h).toFixed(3));
    resY.push(y);
    resZ.push(z);
    resT.push(a);
  }
  return { x: resX, y: resY, z: resZ, t: resT };
}

const initialValues = {
```

```
  a: 0,
  b: 1,
  x0: 0,
  y0: 1,
  fn: (x: number, y: number) => y * (1 - x),
};
for (let n = 10; n < 1000; n++) {
  const { a, b, x0, y0, fn } = initialValues;
  const euler = EulerMethodI(a, b, x0, y0, n, fn);
  const runge = RungeKuttaMethodI(a, b, x0, y0, n, fn);

  const average = runge.y.reduce((cur, total) => cur + total) /
runge.y.length;
  const disp =
    euler.x.map((e) => (e - average) ** 2).reduce((cur, total) => cur +
total) /
    runge.x.length;
  const mean = Math.sqrt(disp);
  console.log(mean);
  break;
}
```