Отлично
МСсу
27.12.21

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт искусственного интеллекта

Кафедра высшей математики

КУРСОВАЯ РАБОТА

по дисциплине

«Сеточные модели уравнений с частными производными»

Тема курсовой работы:

«Решение задачи Коши для системы уравнений на заданном отрезке, где краевые условия удовлетворяют другой системе уравнений»

Студент группы *КМБО-03-18* *Мусатов Д.Ю.*

Руководитель курсовой работы *Сенявин М.М.*

Работа представлена к защите «16» декабря 2021 г. (подпись студента)

«Допущен к защите» «27» декабря 2021 г. (подпись руководителя)

МОСКВА — 2021

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт искусственного интеллекта

Кафедра высшей математики

Утверждаю

Заведующий кафедрой ______________ *Ю.И.Худак*

«8» ноября 2021г.

ЗАДАНИЕ
на выполнение курсовой работы
по дисциплине «Сеточные модели уравнений с частными производными»

Студент *Мусатов Д.Ю.* Группа *КМБО-03-18*

1. **Тема: «Решение задачи Коши для системы уравнений на заданном отрезке, где краевые условия удовлетворяют другой системе уравнений».**

2. **Перечень заданий:** решить задачу Коши для системы уравнений

$$y'_1 = \sin(y_2)$$
$$y'_2 = \cos(y_1)$$

на отрезке $[a, b] = [1,3]$ с шагом $h = 0{,}1$; $y_1(a) = p, y_2(b) = q$, где p и q удовлетворяют системе уравнений

$$p^2 + q^2 = 1$$
$$\sin(p - q) + 0{,}3p = 1.$$

3. **Срок представления к защите курсовой работы: до «   » ___________ 2021 г.**

Задание на курсовую работу выдал	«8» ноября 2021г.	______________	(Сенявин)
Задание на курсовую работу получил	«8» ноября 2021г.	______________	(Мусатов Д.)

Содержание

1 Введение. Постановка задачи

Вариант №15

Необходимо решить задачу коши для системы уравнений:

$$y_1' = sin(y_2)$$
$$y_2' = cos(y_1)$$

на отрезке $[a, b] = [1, 3]$ с шагом $h = 0.1$; $y_1(a) = p$, $y_2(a) = q$, где p и q удовлетворяют системе уравнений:

$$p^2 + q^2 = 1$$
$$sin(p - q) + 0.3p = 1$$

Разобъём поставленную задачу на два этапа:

1. Нахождение p и q, удовлетворяющих системе уравнений

$$p^2 + q^2 = 1$$
$$sin(p - q) + 0.3p = 1$$

2. Решение задачи Коши для системы уравнений

$$y_1' = sin(y_2)$$
$$y_2' = cos(y_1)$$

на отрезке $[a, b] = [1, 3]$ с шагом $h = 0.1$; и начальными условиями $y_1(a) = p$, $y_2(a) = q$

Решение этапов будем проводить последовательно: найдем p и q (первый этап - решение нелинейной системы алгебраических уравнений (НСЛАУ)), затем будем использовать этот результат как входную информацию для решения задачи Коши для системы уравнений (второй этап - решение системы обыкновенных дифференцциальных уравнений (СДУ)).

2 Теоретический обзор

2.1 Решение нелинейных уравнений

Рассмотрим уравнение вида: $f(x) = 0$, где $f(x)-$ функция, определенная и непрерывная на некотором промежутке. Требуется найти корни данного уравнения. Оно ищется в два этапа:

1. Находятся отрезки $[a_i, b_i]$, внутри каждого из которых содержится ровно один корень.

2. С помощью того или иного итерационного метода значения корней уточняются.

Одним из итерационных методов вычисления корней является метод Ньютона.

2.2 Метод Ньютона (метод касательных)

Метод Ньютона или касательных заключается в том, что если x_n- некоторое приближение к корню x_* уравнения $f(x) = 0, f \in C^1$, то следующее приближение определяется как корень касательной к функции $f(x)$, проведенной в точке x_n.

Уравнение касательной к функции $f(x)$ в точке x_n имеет вид:

$$f'(x_j) = \frac{y - f(x_n)}{x - x_n}$$

В уравнении касательной положим $y = 0$ и $x = x_{n+1}$.

Тогда алгоритм последовательных вычислений в методе Ньютона состоит в следующем:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Сходимость метода касательных квадратичная, порядок сходимости равен 2.

Таким образом, метод Ньютона (метод касательных) применяется в том случае, если уравнение $f(x) = 0$ имеет корень $x \in [a; b]$, и выполняются условия:

1) функция $y = f(x)$ определена и непрерывна при $x \in [a; b]$;

2) $f(a) \cdot f(b) < 0$ (функция принимает значения разных знаков на концах отрезка $[a; b]$);

3) производные $f(x)$ и $f'(x)$ сохраняют знак на отрезке $[a; b]$ (т.е. функция $f(x)$ либо возрастает, либо убывает на отрезке $[a; b]$, сохраняя при этом направление выпуклости);

4) $f'(x) \neq 0$ при $x \in [a; b]$.

Геометрически этот метод означает замену на каждой итерации графика $y = f(x)$ касательной к нему. Для метода Ньютона имеет место следующая оценка:

$$|x_n - x^*| \leq \frac{M_2}{2m_1} \cdot |x_n - x_{n-1}|^2, \text{ где } M_2 = \max_{a \leq xb} |f''(x)|, m_1 = \min_{a \leq xb} |f'(x)|.$$

2.3 Решение систем нелинейных уравнений

Систему нелинейных уравнений можно записать в координатном виде

$$f_k\left(x_1, x_2, x_3, \ldots, x_n\right) = 0, 1 \ll k \ll n.$$

Такие системы решаются практически только итерационными методами. Нулевое приближение в случае двух переменных можно найти графически: построить на плоскости (x_1, x_2) кривые $f_1\left(x_1, x_2\right) = 0$ и $f_2\left(x_1, x_2\right) = 0$ и найти точки их пересечения.

2.4 Метод Ньютона для решения системы нелинейных уравнений

Пусть известно некоторое приближение $x^{(s)}$ к корню $\bar{x}$. Как и для одной переменной, запишем исходную систему в виде $f\left(x^{(s)} + \Delta x\right) = 0$, где $\Delta x = \bar{x} - x^{(s)}$. Разлагая эти уравнения в ряды и ограничиваясь первыми дифференциалами, т.е. линеаризуя функцию, получим

$$\sum_{i=1}^{n} \frac{\partial f_k\left(x^{(s)}\right)}{\partial x_i} \Delta x_i^{(s)} = -f_k\left(x^{(s)}\right), 1 \leq k \leq n.$$

Эта система уравнений, линейных относительно приращений $\Delta x_i^{(s)}$, все коэффициенты этой системы выражаются через последнее приближение $x^{(s)}$. Решив эту систему, найдем новое приближение $x^{(s+1)} = x^{(s)} + \Delta x^{(s)}$.

Метод Ньютона можно свести к методу последовательных приближений, положив $\varphi(x) = x - \left[\frac{\partial f}{\partial x}\right]^{-1} f(x)$, где $\left[\frac{\partial f}{\partial x}\right]^{-1}$ есть матрица, обратная матрице производных.

Аналогично проводится теоретический анализ условий сходимости. Если нулевое приближение выбрано удачно, то метод Ньютона сходится, причем очень быстро. Поэтому на практике этот метод используют чаще всего.

В отличие от других методов решения систем нелинейных уравнений, для метода Ньютона хорошим критерием окончания итераций является условие $\left\|x^{(s)} - x^{(s+1)}\right\| \leq \varepsilon$. В самом деле, вблизи корня ньютоновские итерации сходятся квадратично, поэтому если этот критерий выполнен, то $\left\|x^{(s+1)} - \bar{x}\right\| \approx \varepsilon^2 \leq \varepsilon$.

Сходимость итераций исследуем так же, как и для одной переменной. Обозначим компоненты решения через $\bar{x}_k$ и преобразуем погрешность очередной итерации

$$x_k^{(s+1)} - \overline{x_k} = f_k\left(x_1^{(s)}, \ldots, x_n^{(s)}\right) - f_k\left(\overline{x_1}, \ldots, \overline{x_n}\right) = f_k\left(x^{(s)}\right) - f_k(\overline{\mathrm{x}}) = \left[\frac{\partial f_k\left(\xi_k\right)}{\partial l}\right] \rho\left(x^{(s)}, \overline{\mathrm{x}}\right)$$
$$= \sum_{i=1}^{n}\left(x_i^{(s)} - \bar{x}_l\right)\left[\frac{\partial f_k\left(\xi_k\right)}{\partial x_i}\right],$$

где l - направление, соединяющее многомерные точки $x^{(s)}$ и $\bar{x}$, а ξ_k- некоторая точка, лежащая между ними на этом направлении. Это равенство означает, что вектор погрешности нового приближения равен матрице производных, умноженной на вектор погрешности предыдущего приближения.

2.5 Задача Коши

Обыкновенное дифференциальное уравнение p-го порядка

$$u^{(p)}(x) = f\left(x, u, u', u'', \ldots, u^{(p-1)}\right)$$

при помощи замены $u^{(k)}(x) \equiv u_k(x)$ можно свести к эквивалентной системе p уравнений первого порядка

$$u_k'(x) = u_{k+1}(x), 0 \leq k \leq p-2,$$
$$u_{p-1}'(x) = f\left(x, u_0, u_1, \ldots, u_{p-1}\right)$$

где $u_0(x) \equiv u(x)$. Аналогично, произвольную систему дифференциальных уравнений любого порядка можно заменить некоторой эквивалентной системой уравнений первого порядка

$$u_k'(x) = f_k\left(x, u_1, u_2, \ldots, u_p\right), \quad 1 \leq k \leq p$$

Система p-го порядка имеет множество решений, которое в общем случае зависит от p параметров $c = \{c_1, c_2, \ldots, c_p\}$. Для определения значений этих параметров, т.е. для выделения единственного (или нужного) решения, надо наложить p дополнительных условий на функции $u_k(x)$.

Задача Коши (задача с начальными условиями) имеет дополнительные условия вида $u_k(\xi) = \eta_{k'} 1 \leq k \leq p$, т.е. заданы значения всех функций в одной и той же точке $x = \xi$. Эти условия можно рассматривать как задание координат начальной точки $(\xi, \eta_1, \eta_2, \ldots, \eta_p)$ интегральной кривой в (p + 1)-мерном пространстве $(x, u_1, u_2, \ldots, u_p)$. Решение при этом требуется найти на некотором отрезке $\xi \leq x \leq X$, так что точку x = ξ можно считать начальной точкой этого отрезка.

2.6 Метод Эйлера

Простейший численный метод решения систем обыкновенных дифференциальных уравнений. Пусть дана задача Коши для уравнения первого порядка:

$$\frac{dy}{dx} = f(x, y),$$
$$y_{|x=x_0} = y_0,$$

где функция f определена на некоторой области $D \subset \mathbb{R}^2$. Решение ищется на интервале $(x_0, b]$. На этом интервале введем узлы: $x_0 < x_1 < \cdots < x_n \leq b$. Приближенное решение в узлах x_i, которое обозначим через y_i, определяется по формуле:

$$y_i = y_{i-1} + (x_i - x_{i-1}) f(x_{i-1}, y_{i-1}), \quad i = 1, 2, 3, \ldots, n.$$

Эти формулы непосредственно обобщаются на случай систем обыкновенных дифференциальных уравнений.

2.7 Метод Рунге-Кутты

Метод, позволяющий строить схемы различного порядка точности. Наиболее используемы схемы четвертого порядка точности, образующие семейство четырехчленных схем. Метод Рунге- Кутты четвёртого порядка при вычислениях с постоянным шагом интегрирования столь широко распространён, что его часто называют просто методом Рунге — Кутты.

Рассмотрим задачу Коши для системы обыкновенных дифференциальных уравнений первого порядка

$$y' = f(x, y), \quad y\left(x_0\right) = y_0.$$

Тогда приближенное значение в последующих точках вычисляется по итерационной формуле:

$$y_{n+1} = y_n + \frac{h}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right)$$

Вычисление нового значения проходит в четыре стадии:

$$\begin{gathered} K_1 = f\left(x_n, y_n\right), \\ K_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_1\right), \\ K_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_2\right), \\ K_4 = f\left(x_n + h, y_n + hK_3\right), \end{gathered}$$

где h - величина шага сетки по x. Этот метод имеет четвёртый порядок точности. Это значит, что ошибка на одном шаге имеет порядок $O\left(h^5\right)$, а суммарная ошибка на конечном интервале интегрирования имеет порядок $O\left(h^4\right)$.

На случай систем уравнений схему Рунге-Кутты легко переносятся, как во всех других методах, при помощи формальной замены y, $f(x, y)$ на $\boldsymbol{y}, f(x, \boldsymbol{y})$. Например, для системы двух уравнений

$$\begin{gathered} u'(x) = f(x, u(x), v(x)), \\ v'(x) = q(x, u(x), v(x)), \end{gathered}$$

обозначая через y, z приближенные значения функций $u(x), v(x)$, запишем аналогичную четырехчленную схему следующим образом:

$$y_{n+1} = y_n + \frac{h}{6}\left(K_1 + 2K_2 + 2K_3 + K_4\right),$$
$$z_{n+1} = z_n + \frac{h}{6}\left(Q_1 + 2Q_2 + 2Q_3 + Q_4\right),$$

где

$$K_1 = f\left(x_n, y_n, z_n\right),$$
$$Q_1 = q\left(x_n, y_n, z_n\right),$$
$$K_2 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_1, z_n + \frac{h}{2}Q_1\right),$$
$$Q_2 = q\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_1, z_n + \frac{h}{2}Q_1\right)$$
$$K_3 = f\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_2, z_n + \frac{h}{2}Q_2\right)$$
$$Q_3 = q\left(x_n + \frac{h}{2}, y_n + \frac{h}{2}K_2, z_n + \frac{h}{2}Q_2\right)$$
$$K_4 = f\left(x_n + h, y_n + hK_3, z_n + hQ_3\right)$$
$$Q_4 = q\left(x_n + h, y_n + hK_3, z_n + hQ_3\right)$$

3 Решение поставленной задачи

3.1 1 этап - Решение НСЛАУ

Найдём значения p и q, удовлетворяющие системе уравнений:

$$p^2 + q^2 = 1$$
$$sin(p - q) + 0.3p = 1$$

Построим график. Воспользуемся для построения онлайн калькулятором Desmos. Получим следующее изображение:

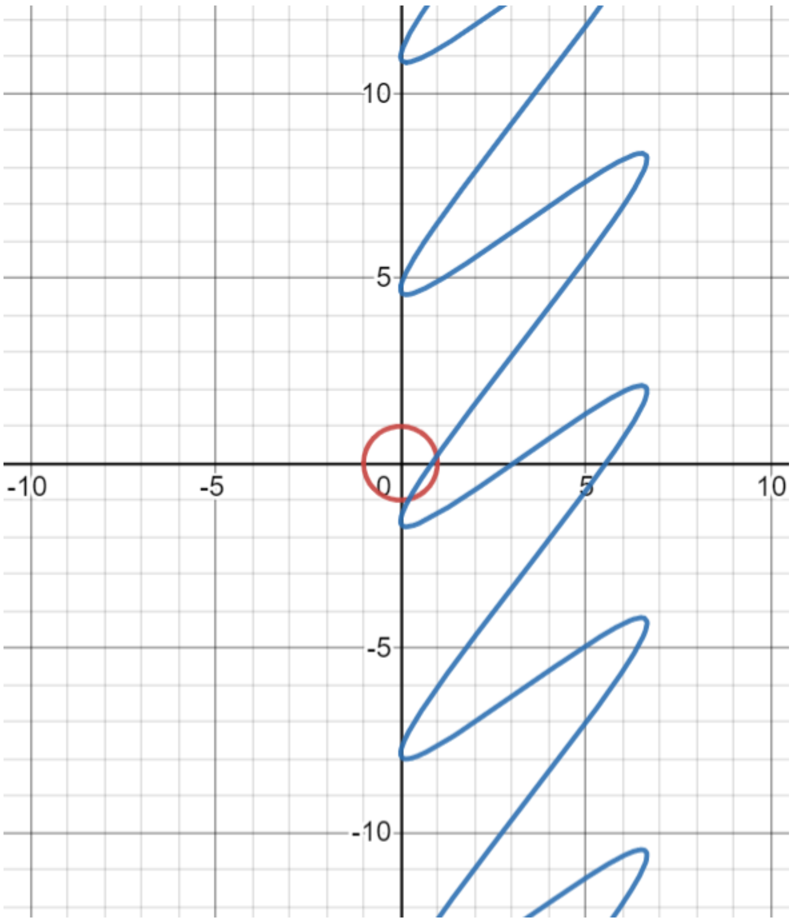


Рисунок 1. график функций принадлежащих системе

Как видно из графика, есть два пересечения функций — существует два решения данной системы нелинейных уравнений.

Метод Ньютона для решения системы нелинейных уравнений:

Рассмотрим сначала точку пересечения в первой четверти. По графику определим начальное приближение $x_0 = (1, 0)$. Найдем его решение с помощью метода Ньютона.

Найдем матрицу Якоби w, это матрица частных производных каждого уравнения. Определитель матрицы Якоби для x_0 должен быть не равен 0.

Найдем вектор приращений, который рассчитывается как $dx = -w^{-1} \cdot f(x_0)$. Найдем вектор решения $x = x_0 + dx$. Проверяем условие сходимости: разность $x - x_0$ должна быть меньше точности вычислений.

Зададим точность вычислений: $e = 0.00001$.

Итерация	Матрица Якоби w	p	q	$f_1(p, q)$	$f_2(p, q)$	$max(p - p_0, q - q_0)$
1	$\begin{bmatrix} 2 & 0 \\ 0.840302 & -0.540302 \end{bmatrix}$	1	0.261837	0.0685585	-0.0270695	0.261837
2	$\begin{bmatrix} 2 & 0.523673 \\ 1.03971 & -0.739706 \end{bmatrix}$	0.981947	0.199867	0.00416625	-0.00065927	0.06197
3	$\begin{bmatrix} 1.96389 & 0.399733 \\ 1.00945 & -0.709449 \end{bmatrix}$	0.980448	0.196806	0.0000115	-0.00000084	0.00306121
4	$\begin{bmatrix} 1.9609 & 0.393611 \\ 1.00835 & -0.708347 \end{bmatrix}$	0.980444	0.196798	0	-0.00000008	0.00000745

Таблица 1. результаты вычислений методом Ньютона начальное приближение (1,0)

Ответ: p = 0.980444, q = 0.196798;

Рассмотрим точку пересечения функций в третьей четверти. По графику определим начальное приближение $x_0 = (0, -1)$.

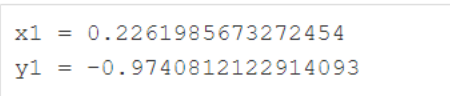
Методом Ньютона получим следующее решение:

Итерация	Матрица Якоби w	p	q	$f_1(p, q)$	$f_2(p, q)$	$max(p - p_0, q - q_0)$
1	$\begin{bmatrix} 0 & -2 \\ 0.804302 & -0.540302 \end{bmatrix}$	0.188657	-1	0.0355915	-0.0155338	0.188657
2	$\begin{bmatrix} 0.377314 & -2 \\ 0.672906 & -0.372906 \end{bmatrix}$	0.22545	-0.975263	0.00196564	-0.0000676	0.0367932
3	$\begin{bmatrix} 0.450901 & -1.95053 \\ 0.661693 & -0.361693 \end{bmatrix}$	0.226198	-0.974081	0.0000019	-0.00000006	0.00118059
4	$\begin{bmatrix} 0.452396 & -1.94816 \\ 0.662097 & -0.362097 \end{bmatrix}$	0.226199	-0.974081	0	-0.00000002	0.00000113

Таблица 2. результаты вычислений методом Ньютона начальное приближение (0,-1)

Ответ: p = 0.226199, q = -0.974081;

Выполним проверку с помощью онлайн калькуляторов, позволяющих решить НСЛАУ



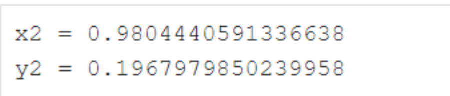


Рисунок 2. Результат решения системы с помощью онлайн калькуляторов

Как можем видеть, результаты совпали с точностью до 10^{-5}. Проверку выполнили. Код программы представлен в приложении.

3.2 2 этап - Решение СДУ

Найдем решение задачи Коши для системы уравнений:

$$y_1' = sin(y_2)$$

$$y_2' = cos(y_1)$$

на отрезке $[a, b] = [1, 3]$ с шагом $h = 0.1$;

На первом этапе, в ходе вычислений, получили два решения системы.

$$p_1 = 0.980444, q_1 = 0.196798;$$

$$p_2 = 0.226199, q_2 = -0.974081;$$

Значит, краевые условия следующие, обозначим с помощью верхних индексов номера решений:

$$y_1^1(1) = 0.980444, \quad y_2^1(1) = 0.196798;$$

$$y_1^2(1) = 0.226199, \quad y_2^2(1) = -0.974081;$$

Будем искать решение, используя методы Эйлера и Рунге-Кутта 4-го порядка.

Использую решение методом Рунге-Кутты получим следующий график и таблицу:

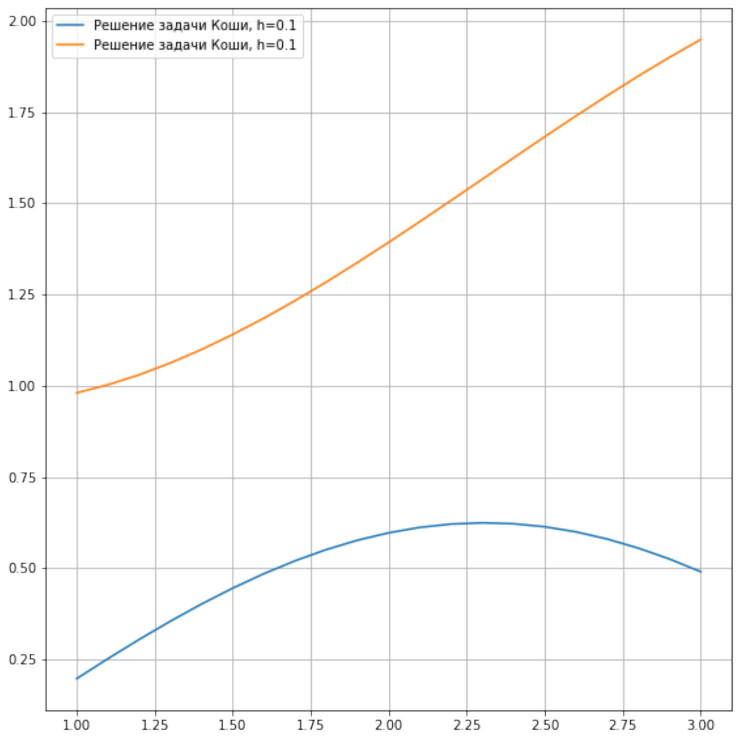


Рисунок 3. Графики функций СДУ с краевыми условиями y_1^{rq1}, y_2^{rq1}

y_1	k_1	k_2	k_3	k_4	y_2	q_1	q_2	q_3	q_4
0.98044	0.01955	0.02227	0.02223	0.02489	0.1968	0.05567	0.05485	0.05474	0.0538
1.00269	0.02489	0.02749	0.02744	0.02995	0.25157	0.0538	0.05275	0.05264	0.05147
1.03014	0.02996	0.0324	0.03234	0.0347	0.30425	0.05147	0.05018	0.05007	0.04867
1.06249	0.0347	0.03697	0.0369	0.03907	0.35436	0.04867	0.04715	0.04705	0.04541
1.09941	0.03907	0.04115	0.04107	0.04305	0.40143	0.04541	0.04366	0.04357	0.04172
1.14051	0.04305	0.04492	0.04483	0.04659	0.44503	0.04171	0.03975	0.03966	0.0376
1.18537	0.0466	0.04825	0.04816	0.04969	0.48472	0.0376	0.03543	0.03535	0.03309
1.23355	0.0497	0.05113	0.05102	0.05233	0.52009	0.03309	0.03073	0.03067	0.02823
1.28461	0.05234	0.05353	0.05343	0.0545	0.55078	0.02823	0.02571	0.02565	0.02307
1.33807	0.05451	0.05547	0.05536	0.0562	0.57645	0.02306	0.0204	0.02036	0.01764
1.39346	0.0562	0.05693	0.05682	0.05742	0.59682	0.01764	0.01487	0.01483	0.01202
1.45031	0.05742	0.05791	0.0578	0.05817	0.61167	0.01202	0.00916	0.00914	0.00626
1.50815	0.05817	0.05842	0.05831	0.05844	0.62082	0.00626	0.00336	0.00334	0.00043
1.56649	0.05844	0.05846	0.05834	0.05824	0.62416	0.00043	-0.00249	-0.00249	-0.0054
1.62487	0.05824	0.05802	0.0579	0.05756	0.62167	-0.00541	-0.00831	-0.0083	-0.01117
1.68281	0.05756	0.05711	0.05699	0.05641	0.61338	-0.01118	-0.01403	-0.01401	-0.01682
1.73984	0.05641	0.05571	0.0556	0.05479	0.59936	-0.01682	-0.0196	-0.01956	-0.02228
1.79548	0.05478	0.05385	0.05374	0.05269	0.57979	-0.02228	-0.02494	-0.0249	-0.02748
1.84925	0.05268	0.05151	0.0514	0.05012	0.55488	-0.02749	-0.03001	-0.02995	-0.03239
1.90069	0.05011	0.04871	0.0486	0.04708	0.52492	-0.03239	-0.03475	-0.03469	-0.03695
1.94933					0.49021				

Таблица №3. таблица коэффициентов для метода Рунге-Кутты на кажоый итерации для краевых условий y_1^{rq1}, y_2^{rq1}

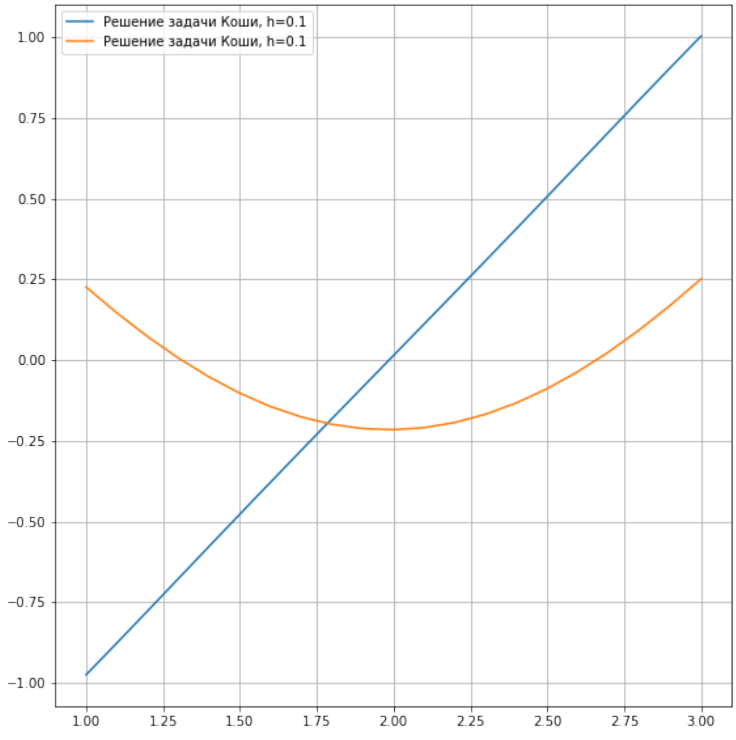


Рисунок 4. Графики функций СДУ с краевыми условиями y_1^{rq2}, y_2^{rq2}

y_1	k_1	k_2	k_3	k_4	y_2	q_1	q_2	q_3	q_4
0.2262	-0.08272	-0.07988	-0.07986	-0.07681	-0.97408	0.09745	0.0983	0.09827	0.09893
0.14637	-0.07681	-0.07355	-0.07353	-0.07007	-0.87583	0.09893	0.09942	0.0994	0.09973
0.07286	-0.07007	-0.06643	-0.06642	-0.06261	-0.77644	0.09973	0.09993	0.09992	0.1
0.00646	-0.06261	-0.05863	-0.05864	-0.05451	-0.67654	0.1	0.09997	0.09997	0.09986
-0.05215	-0.05452	-0.05026	-0.05027	-0.0459	-0.57658	0.09986	0.09968	0.0997	0.09948
-0.1024	-0.0459	-0.04143	-0.04144	-0.03687	-0.47689	0.09948	0.09922	0.09924	0.09897
-0.14382	-0.03688	-0.03223	-0.03225	-0.02753	-0.37767	0.09897	0.09869	0.09872	0.09845
-0.17604	-0.02754	-0.02277	-0.02278	-0.01797	-0.27896	0.09845	0.0982	0.09825	0.09803
-0.19882	-0.01797	-0.01313	-0.01314	-0.00827	-0.18073	0.09803	0.09785	0.0979	0.09776
-0.21195	-0.00828	-0.0034	-0.0034	0.00149	-0.08285	0.09776	0.09767	0.09773	0.09769
-0.21535	0.00149	0.00637	0.00637	0.01124	0.01486	0.09769	0.09771	0.09776	0.09782
-0.20898	0.01124	0.01608	0.01609	0.0209	0.1126	0.09782	0.09794	0.09799	0.09815
-0.1929	0.0209	0.02567	0.02568	0.03041	0.21057	0.09815	0.09834	0.09838	0.09861
-0.16723	0.0304	0.03506	0.03507	0.03966	0.30894	0.0986	0.09885	0.09888	0.09913
-0.13217	0.03966	0.04416	0.04417	0.04857	0.4078	0.09913	0.09937	0.09939	0.09961
-0.08803	0.04857	0.05286	0.05287	0.05704	0.50718	0.09961	0.0998	0.09981	0.09994
-0.03518	0.05704	0.06107	0.06107	0.06495	0.60698	0.09994	0.1	0.1	0.09997
0.02587	0.06495	0.06867	0.06867	0.07221	0.70696	0.09997	0.09983	0.09982	0.09955
0.09451	0.07221	0.07556	0.07554	0.0787	0.80676	0.09955	0.09915	0.09913	0.09856
0.17002	0.0787	0.08164	0.08162	0.08434	0.90587	0.09856	0.09782	0.09779	0.09685
0.25162					1.00364				

Таблица №4. Коэффициенты для метода Рунге-Кутты на каждой итерации для краевых условий y_1^{rq2}, y_2^{rq2}

Получаем значения:

$$y_1^{rq1} = 1.94933 \quad y_2^{rq1} = 0.49021$$

$$y_1^{rq2} = 0.25162 \quad y_2^{rq2} = 1.00364$$

Теперь выполним решение данной задачи с помощью метода Эйлера:

Получаем значения:

$$y_1^{e1} = 1.95225 \quad y_2^{e1} = 0.495014$$

$$y_1^{e2} = 0.242563 \quad y_2^{e2} = 1.003172$$

Графики будут аналогичны графикам построенным при решении методом Рунге-Кутты.

Сравним значения $y(1)$ и $y(2)$, полученные обоими методами.

$$|y_1^{rq1} - y_1^{e1}| = 0.00292 \quad |y_2^{rq1} - y_2^{e1}| = 0.004804$$

$$|y_1^{rq2} - y_1^{e2}| = 0.009057 \quad |y_2^{rq2} - y_2^{e2}| = 0.000468$$

Максимальное отличие между решениями не превосходит 0.01 Код обоих методов представлен в приложении.

Оценка погрешности методов решения СДУ с помощью правила Рунге

Для оценки точности решения СДУ будем использовать следующую формулу:

$$error = \frac{|y_{i,h} - y_{i,\frac{h}{2}}|}{2^p - 1}$$

где $y_{i,h}$ - решение задачи с шагом h, а $y_{i,\frac{h}{2}}$ - решение с шагом h/2.

Под p понимается порядок точности использованного численного метода. В нашем случае используется метод Рунге-Кутты 4 порядка (т. е. $p = 4$) и метод Эйлера, порядок которого равен 1 (т. е. $p = 1$).

На отрезке от 1 до 3 погрешность решения методом Рунге-Кутты с шагом 0.1 и 0.05 для функции y_1^{rq1} и начальными условиями $y_1^1(1)$ составила 3.8979e-08, а для функции y_2^{rq1} и начальных условий $y_2^1(1)$ составила 2.2665e-08.

На отрезке от 1 до 3 погрешность решения методом Рунге-Кутты с шагом 0.1 и 0.05 для функции y_1^{rq2} и начальными условиями $y_1^2(1)$ составила 5.2398e-08, а для функции y_2^{rq2} и начальных условий $y_2^2(1)$ составила 6.2388e-08.

На отрезке от 1 до 3 погрешность решения методом Эйлера с шагом 0.1 и 0.05 для функции y_1^{eu1} и начальными условиями $y_1^1(1)$ составила 0.00145588, а для функции y_2^{eu1} и начальных условий $y_2^1(1)$ составила 0.00242047.

На отрезке от 1 до 3 погрешность решения методом Эйлера с шагом 0.1 и 0.05 для функции y_1^{eu1} и начальными условиями $y_1^2(1)$ составила 0.00452879, а для функции y_2^{eu1} и начальных условий $y_2^2(1)$ составила 0.00048038.

4 Заключение

В процессе выполнения курсовой работы, при решении задачи Коши для системы обыкновенных дифференциальных уравнений(СДУ), было написано две программы на языках C++ и Python. Первая основана на методе Ньютона и реализована на C++ (для получения краевых условий СДУ), вторая – на методах Эйлера и Рунге-Кутты на Python (для решения СДУ и построения графиков функций).

На первом этапе найдены два решения НСЛАУ, на втором в качестве краевых условий взято каждое из них и выполнено решение СДУ двумя методами, сравнение результатов которых было выполено после. Также была сделана оценка точности решения СДУ.

5 Список литературы

1. Аристов В. В., Строганов А. В. Лабораторный практикум по численным методам / Федеральное государственное бюджетное образовательное учреждение высшего профессионального обучения «Московский государственный технический университет радиотехники, электроники и автоматики» — М., 2012. — 46 с., электронное издание.

2. Калиткин Н.Н. «Численные методы» – М.: Наука, 1978 - 512 с.

3. Ковязин, В.Ф. Введение в численные методы: Учебное пособие для вузов. / В.Ф. Ковязин. - СПб.: Лань, 2009. - 288 с.

4. Бахвалов Н.С. «Численные методы» – М.: Наука, 1975 – 369 с.

5. Арушанян И. О. «Численные алгоритмы решения нелинейных уравнений. Учебное пособие» – М.: Изд-во ЦПИ при механико-математическом факультете МГУ, 2018 – 39 с.

6 Приложение

Листинг 1: polynoms.h

```
//This code is the intellectual property of Danila Musatov .
//Contacts danilarumus2000@gmail.com
//This program contains a number of numerical methods for finding the root on the interval.
// Copyright reserved 2021

#pragma once
#include<iostream>
#include<vector>
#include<cmath>
#include<Eigen/Dense>
#include<Eigen/LU>


using std::cout;
using std::vector;
template <typename T>
using v_func_v_x = vector < double(*)(T)>;
template <typename T>
using m_func_x = vector < vector <T>>;
using v_d = vector<double>;
using EVX = Eigen::VectorXf;
using EMX = Eigen::MatrixXf;


template <typename X>
double my_polynom_1(X x) {
  return x[0] * x[0] + x[1] * x[1] - 1;
}


template <typename X>
double my_polynom_2(X x) {
  return sin(x[0] - x[1]) + 0.3 * x[0] - 1;
}

template <typename X>
double my_f_1_d_1(X x) {
  return 2 * x[0];
}
```

```

template <typename X>
double my_f_1_d_2(X x) {
  return 2*x[1];
}

template <typename X>
double my_f_2_d_1(X x) {
  return cos(x[0] - x[1]) + 0.3;
}

template <typename X>
double my_f_2_d_2(X x) {
  return -cos(x[0] - x[1]);
}

template <typename X>
double my_fi_1(X x) {
  return sqrt(1 - x[1] * x[1]);
}

template <typename X>
double my_fi_2(X x) {
return x[0] - asin(1 - 0.3 * x[0]);
}
```

Листинг 2: functions.h

```
#include<iostream>
#include<vector>
#include<cmath>
#include<Eigen/Dense>
#include<Eigen/LU>

using std::cout;
using std::vector;
template <typename T>
using v_func_v_x = vector < double(*)(T)>;
template <typename T>
using m_func_x = vector < vector <T>>;
using v_i = vector<int>;
using v_d = vector<double>;
using EVX = Eigen::VectorXf;
using EMX = Eigen::MatrixXf;
```

```
using V_EMX = vector<Eigen::MatrixXf>;

template <typename T>
double max_delt(const T& x, const T& x_1) {
  double max = -1;
  for (unsigned int i = 0; i < x.size(); ++i) {
    max = (fabs(x[i] - x_1[i]) > max) ? fabs(x[i] - x_1[i]) : max;
  }
  return max;
}

EVX conv_v_e(const v_d& x) {
  EVX e_v_X(x.size());
  for (unsigned int i = 0; i < x.size(); ++i) {
    e_v_X(i) = x[i];
  }
  return e_v_X;
}

v_d conv_e_v(const EVX& x) {
  v_d x_v(x.size());
  for (int i = 0; i < x.size(); ++i) {
    x_v[i]=x[i];
  }
  return x_v;
}

template<typename T>
EMX W_completion(const m_func_x< double(*)(EVX x)>& d_fi, const T& e_v_x)
   {
  EMX W(e_v_x.size(), e_v_x.size());
  for (int i = 0; i < e_v_x.size(); ++i) {
    for (int j = 0; j < e_v_x.size(); ++j) {
      W(i, j) = d_fi[i][j](e_v_x);
    }
  }
  return W;
}


EVX Fx(const EVX& evx, const v_func_v_x<EVX>& f) {
  EVX FX(f.size());
  for (unsigned int i = 0; i < f.size(); ++i) {
```

```
    FX[i] = f[i](evx);
  }
  return FX;
}

v_d converter_e_v(const EVX& x) {
  v_d b_x(x.size());
  for (int i = 0; i < x.size(); i++) b_x[i]=x[i];
  return b_x;
}

template < typename F, typename X>
double max_f_x(const F& f_x, const X& x) {
  double max = -1;
  for (unsigned int i = 0; i < x.size(); ++i) {
    max = (fabs(f_x[i](x)) > max) ? fabs(f_x[i](x)) : max;
  }
  return max;
}

template <typename F, typename X>
void print_disc(const F& f, const X& x) {
  cout << "\n\nx[0] = "<< x[0] << "      x[1] = " << x[1] <<"        f_0(x) = " <<f[0](x) << "        f_1(x) = " << f[1](x) <<"        discrepancy = " << max_f_x<F, X>(f, x) << "\n\n";
  return;
}

template <typename F, typename X>
void print_disc(const F& f, const X& x, EMX W) {
  cout << "\n\nx[0] = " << x[0] << "      x[1] = " << x[1] << "        f_0(x) = " << f[0](x) << "        f_1(x) = " << f[1](x) << "        discrepancy = " << max_f_x<F, X>(f, x) <<std::endl <<
    "Jacobi Matrix: " << std::endl << W << "\n\n";
  return;
}

v_d Newtonw_s(const m_func_x< double(*)(EVX x)>& d_fi,
        const v_func_v_x<EVX>& f,
        const v_d& x, const double& eps) {

  EVX e_v_X(x.size());
  EVX e_v_X_1(x.size());
```

```
  e_v_X = conv_v_e(x);
  EMX W(x.size(), x.size());
  W = W_completion(d_fi, e_v_X);
  e_v_X_1 = e_v_X - W.inverse() * Fx(e_v_X, f);
  print_disc<v_func_v_x<EVX>, EVX>(f, e_v_X);
  print_disc<v_func_v_x<EVX>, EVX>(f, e_v_X_1,W);

  while (max_delt<EVX>(e_v_X, e_v_X_1) > eps) {
    cout << (max_delt<EVX>(e_v_X, e_v_X_1)) << std::endl;
    e_v_X = e_v_X_1;
    cout << "+1" ;
    for (unsigned int i = 0; i < f.size(); i++) {
      W = W_completion(d_fi, e_v_X);
      e_v_X_1 = e_v_X - W.inverse() * Fx(e_v_X, f);
    }
    print_disc<v_func_v_x<EVX>, EVX>(f, e_v_X_1, W);

  }
  cout << (max_delt<EVX>(e_v_X, e_v_X_1)) << "\n\n";

  return converter_e_v(e_v_X_1);
}
```

Листинг 3: main.cpp

```
//This code is the intellectual property of Danila Musatov .
//Contacts danilarumus2000@gmail.com
//This program contains a number of numerical methods for finding the root on the interval.
// Copyright reserved 2021

#include<iostream>
#include<vector>
#include<cmath>
#include"functions.h"
#include"polynoms.h"

int main() {
  double eps = 0.00001;

  v_func_v_x<v_d> my_sys_f{ &my_polynom_1<v_d> ,&my_polynom_2<v_d> };
  v_func_v_x<v_d> my_sys_fi{ & my_fi_1<v_d>, & my_fi_2<v_d>};
  m_func_x<double(*)(EVX)> d_f{ {my_f_1_d_1<EVX>, my_f_1_d_2<EVX>}, {
      my_f_2_d_1<EVX>, my_f_2_d_2<EVX> }};
  //std::cin » eps;
```

```
    v_d x_0{ 1, 0 };
    v_d x_1;
    v_func_v_x<EVX> my_sys_e_f{ &my_polynom_1<EVX> ,&my_polynom_2<EVX> };
    std::cout << "\n\n Newton\n\n";
    x_1 = Newtonw_s(d_f, my_sys_e_f, x_0, eps);
    auto iter = x_1.begin();   // получаем итератор
    while (iter != x_1.end())      // пока не дойдем до конца
    {
        std::cout << *iter << std::endl;// получаем элементы через итератор
        ++iter;                  // перемещаемся вперед на один элемент
    }
    system("pause");
    return 0;
}
```

Листинг 4: main.py

```
import numpy as np
import matplotlib.pyplot as plt
from math import pi, sin, cos, fabs, log
from tabulate import tabulate
from texttable import Texttable
import latextable

# Функция сохранение в таблицы tex
def savetable(array,numberOfTable):
    table = Texttable()

    table.set_cols_align(["c"] * len(array[0]))
    table.set_cols_dtype(['t'] * len(array[0]))
    table.set_deco(Texttable.HEADER | Texttable.VLINES | Texttable.HLINES)
    table.add_rows(array)

    path = "C:/Users/Danila/Documents/Study/7 semestor/Numerical methods(
        grid models of partial differential equations)/Courses work/Report/
        table_" + str(numberOfTable) + ".tex"
    my_file = open(path, 'w+')
    my_file.write(latextable.draw_latex(table))
    my_file.close()
    return

# Метод эйлера для решения системы исходя из заданных условий
def Method_Euler_sys(func1, func2, y1Start, y2Start, a, b, h):
    xs = np.arange(a, b+h, h)
```

```
    ys1, ys2 = [y1Start], [y2Start] # первые значения приближений
    for point in enumerate(xs[:-1]):
        ys1.append(ys1[-1] + h*func1(ys2[-1]))
        ys2.append(ys2[-1] + h*func2(ys1[-2]))
    return [xs, ys1, ys2]

# Правило рунге для оценки погрешности
def Runge_rule(ys1, ys2, p):
    return  max([np.abs(y2Now - y1) for y1, y2Now in zip(ys1, ys2[::2])])
        / (2**p - 1)

# Вывод значений на каждой итерации и запись в таблицу tex
def print_table(xs, ys1, ys2, prec=6):T
    able5 = []
    for i, (x,y1,y2Now) in enumerate(zip(xs, ys1, ys2)):T
        able5.append([i, round(xs[i], prec), round(ys1[i], prec), round(
            ys2[i], prec)])
        print(f"Итерация {i} : x={round(x, prec)}  y1={round(y1, prec)}
            y2Now={round(y2Now, prec)}")
    return Table5

dy_2_func = lambda x, y, z: cos(y) # заданная функция с новой переменной
dy_1_func = lambda x, y, z: sin(z) # введение новой переменной

# определение коэффициентов K и L
K1 = lambda x, y, z, h: h*dy_1_func(x, y, z)
L1 = lambda x, y, z, h: h*dy_2_func(x, y, z)

K2 = lambda x, y, z, h: h*dy_1_func(x + h/2, y + K1(x, y, z, h)/2, z + L1(
    x, y, z, h)/2)
L2 = lambda x, y, z, h: h*dy_2_func(x + h/2, y + K1(x, y, z, h)/2, z + L1(
    x, y, z, h)/2)

K3 = lambda x, y, z, h: h*dy_1_func(x + h/2, y + K2(x, y, z, h)/2, z + L2(
    x, y, z, h)/2)
L3 = lambda x, y, z, h: h*dy_2_func(x + h/2, y + K2(x, y, z, h)/2, z + L2(
    x, y, z, h)/2)

K4 = lambda x, y, z, h: h*dy_1_func(x + h, y + K3(x, y, z, h), z + L3(x, y
    , z, h))
L4 = lambda x, y, z, h: h*dy_2_func(x + h, y + K3(x, y, z, h), z + L3(x, y
    , z, h))

```

```python

def Runge_Kutta_sys(a=1, b=3, h=0.01, y_1 = 0.980444, y_2 =  0.196798):
    x = np.arange(a, b + h, h)
    y1Now = y_1 # начальное значение y1
    y2Now = y_2 # начальное значение y2
    y1 = []
    y2 = []
    y1.append(y1Now)
    y2.append(y2Now)
    mx = 0
    Table3 = []
    j = 0

    # Рассчет согласно формулам
    for index, i in enumerate(x):
        #print("!")
        k1 = K1(i, y1Now, y2Now, h)
        k2 = K2(i, y1Now, y2Now, h)
        k3 = K3(i, y1Now, y2Now, h)
        k4 = K4(i, y1Now, y2Now, h)
        l1 = L1(i, y1Now, y2Now, h)
        l2 = L2(i, y1Now, y2Now, h)
        l3 = L3(i, y1Now, y2Now, h)
        l4 = L4(i, y1Now, y2Now, h)
        #print(k1, k2, k3, k4)
        #print(l1, l2, l3, l4)
        t = y1Now + (1/6) * (k1 + 2*k2 + 2*k3 + k4)
        y2Now = y2Now + (1/6) * (l1 + 2*l2 + 2*l3 + l4)
        y1Now = t
        Table3.append([round(y1[-1],5), round(k1, 5), round(k2, 5), round(k3, 5), round(k4, 5), round(y2[-1], 5), round(l1, 5), round(l2, 5), round(l3, 5), round(l4, 5)])
        if(i != x[len(x)-1]):
            y1.append(y1Now)
            y2.append(y2Now)

    # Сохранение результатов в таблицу
    savetable(Table3,31)

    # Вывод программы и график

    plt.rcParams["figure.figsize"] = [8.0, 8.0]
    plt.rcParams["figure.autolayout"] = True
```

```
    plt.grid(True)
    #print(x)
    #print(y)
    #print(z)
    plt.plot(x, y2, label="Решение задачиКоши , h={}".format(h))
    plt.plot(x, y1, label="Решение задачиКоши , h={}".format(h))
    plt.legend()
    plt.savefig('plot.png')
    plt.show()
    return x, y1, y2

def Euler_start(a=1, b=3, h=0.01, y_1 = 0.980444, y_2 =  0.196798):
    y1_start_e = y_1
    y2_start_e = y_2
    func1 = lambda y2 : np.sin(y2) # первая функция
    func2 = lambda y1 : np.cos(y1) # вторая функция
    h1 = 0.01
    h2 = h1/2
    xs_h1, ys1_h1, ys2_h1 = Method_Euler_sys(func1, func2, y1_start_e, y2_start_e, a, b, h1)
    xs_h2, ys1_h2, ys2_h2 = Method_Euler_sys(func1, func2, y1_start_e, y2_start_e, a, b, h2)

    plt.plot(xs_h1, ys1_h1, label=f"y1(x), h={h1}")
    plt.plot(xs_h1, ys2_h1, label=f"y2(x), h={h1}")
    plt.grid(True)
    plt.legend()
    plt.savefig('plot.png')
    plt.show()

    plt.plot(xs_h2, ys1_h2, label=f"y1(x), h={h2}")
    plt.plot(xs_h2, ys2_h2, label=f"y2(x), h={h2}")
    plt.grid(True)
    plt.legend()
    plt.show()


    print("Погрешность для y1 : ", Runge_rule(ys1_h1, ys1_h2, 1))
    print("Погрешность для y2 : ", Runge_rule(ys2_h1, ys2_h2, 1))

    Table_5 = print_table(xs_h1, ys1_h1, ys2_h1)
    savetable(Table_5,51)
    print_table(xs_h2, ys1_h2, ys2_h2)
```

```python
    return

def Runge_Kutta_start(a=1, b=3, h=0.01, y_1 = 0.980444, y_2 =  0.196798):
    #решение СДУ с первыми краевыми условиями
    x_h, y_1_rq_h, y_2_rq_h = Runge_Kutta_sys(a, b, h, y_1, y_2)
    x_h2, y_1_rq_h2, y_2_rq_h2 = Runge_Kutta_sys(a, b, h/2, y_1, y_2)

    print("Погрешность для y1 : ", Runge_rule(y_1_rq_h, y_1_rq_h2, 4))
    print("Погрешность для y2 : ", Runge_rule(y_2_rq_h, y_2_rq_h2, 4))

    return


def main():
    a = 1
    b = 3
    h_min = 0.1
    #первое решение НСЛАУ
    y_1_1=0.980444
    y_2_1=0.196798

    #второе решение НСЛАУ
    y_1_2=0.226199
    y_2_2=-0.974081

    #решение СДУ с первыми краевыми условиями методом Рунге-Кутты
    Runge_Kutta_start(a, b, h_min, y_1_1, y_2_1)

    #решение СДУ со вторыми краевыми условиями методом Рунге-Кутты
    Runge_Kutta_start(a, b, h_min, y_1_2, y_2_2)

    #решение СДУ с первыми краевыми условиями методом Эйлера
    Euler_start(a, b, h_min, y_1_1, y_2_1)

    #решение СДУ со вторыми краевыми условиями методом Эйлера
    Euler_start(a, b, h_min, y_1_2, y_2_2)

    return 1

main()
```

Отзыв
о курсовой работе Мусатова Д. Ю.

Работа посвящена решению задачи Коши для системы уравнений на заданном отрезке, краевые условия которой удовлетворяют неоднородной системе алгебраических уравнений (НСЛАУ). Для решения НСЛАУ применяется метод Ньютона. Система дифференциальных уравнений решена двумя методами Эйлера и Рунге-Кутты.

Возможность применения методов обоснована, точность решения задач каждого этапа оценивается. Также сравниваются два решения, полученные разными методами. Результаты достоверны.

Работа заслуживает оценки “отлично”.

Руководитель работы (Сенявин М. М.)