Оглавление

[**Задание** 4](#_Toc153217998)

[**Алгоритм решения** 5](#_Toc153217999)

[**Теоретические сведения** 6](#_Toc153218000)

[**Определение параметров p и q** 9](#_Toc153218001)

[**Решение задачи Коши для системы уравнений** 10](#_Toc153218002)

[**Оценка погрешности** 11](#_Toc153218003)

[**Приложение** 14](#_Toc153218004)

# **Задание**

Вариант 6

Решить задачу Коши для системы уравнений:

)

= cos(x)

на отрезке [a, b] = [0, 4] с шагом h = 0,1;

(a) = p,

(a) = q;

где p, q удовлетворяют системе уравнений:

= 5

tg(pq) – p = 0.1

# **Алгоритм решения**

Сначала, необходимо определить параметры p и q. Для этого решим систему нелинейных уравнений. Будем использовать метод Ньютона.

Далее требуется решить саму задачу Коши для системы уравнений. Это будет сделано методом Адамса. А для вычисления значения функции на 4 предыдущих слоях будет использован метод Рунге-Кутты.

# **Теоретические сведения**

Метод Ньютона для системы нелинейных уравнений

Для решения системы нелинейных уравнений (и соответственно, определения параметров p и q) был выбран метод Ньютона, так как он имеет достаточно высокую скорость сходимости.

Описание метода

Пусть дана система из n нелинейных уравнений с n неизвестными.

, где : , i=1,…,n - нелинейные функции, определенные и непрерывно дифференцируемые в некоторой области

Запишем ее в векторном виде:

= , F(x) = , F(x) = 0

Требуется найти такой вектор = , который, при подстановке в исходную систему, превращает каждое уравнение в верное числовое равенство.

При таком подходе формула для нахождения решения является естественным обобщением формулы одномерного итеративного метода:

*–(*

W = – матрица Якоби.

В рассмотренных предположениях относительно функции F(⋅) при выборе начального приближения из достаточно малой окрестности решения имеет место сходимость последовательности {}. При дополнительном предположении F(⋅) имеет место квадратичная сходимость метода.

В качестве критерия окончания процесса итераций обычно берут условие || ­ - || <, где требуемая точность решения.

Основная сложность метода Ньютона заключается в обращении матрицы Якоби. Вводя обозначение = - получаем СЛАУ для вычисления : .

Тогда = +

Часто метод Ньютона модифицируют следующим образом. По ходу вычислений или заранее выбирают возрастающую последовательность чисел = 0, , …

При вычисление осуществляют по следующей формуле:

= .

Увеличение числа итераций, сопровождающее такую модификацию, компенсируется «дешевизной» одного шага итерации.

Основная вычислительная нагрузка алгоритма заключается в решении СЛАУ:

=

Для нахождения , по которому вычисляется значение вектора на очередной итерации:

+

Достаточные условия сходимости метода Ньютона для систем

Пусть функция F(x) непрерывно дифференцируема в открытом выпуклом множестве Предположим, что существуют такие, что N(, F( ( , причем ||( и W(x) (N(. Тогда существует > 0 такое, что для всех N( последовательность ,, … сходится к и удовлетворяет неравенству

|| - || , k = 0,1,2,…

N(x,r) — открытая окрестность радиуса r с центром в точке x: N(x,r) = {};

W(x) (N( означает, что W(x) непрерывна по Липшицу, где — константа Липшица, то есть ||W(y) – W(x)|| x,y N(.

Метод Адамса

В связи с медленно растущей погрешностью для решения задачи Коши для системы уравнений был выбран метод Адамса.

Метод Адамса относится к многошаговым методам. То есть для того, чтобы вычислить значение функции на данном слое, необходимо знать значения функции на нескольких предыдущих слоях. Представим решение дифференциального уравнения в виде

+

Если решение задачи Коши получено в узлах вплоть до к-го, то можно аппроксимировать подынтегральную функцию, например, интерпояционным многочленом какой-либо степени. Вычислив интеграл от построенного многочлена на отрезке [xk, xk+1], получим ту или иную формулу Адамса. При использовании интерполяционного многочлена третьей степени, построенного по значениям подынтегральной функции в последних четырех узлах, получим метод Адамса четвертого порядка точности:

(55 - 59 -9),

где – значение подынтегральной функции в узле xk­ .Дляиспользования данного метода необходимо знать значения функции на четырёх предыдущих слоях, они будут определины методом Рунге - Кутты четвертого порядка.

Метод Рунге-Кутты 4 порядка

Семейство явных методов Рунге-Кутты p-го порядка записывается в виде совокупности формул:

, ,

+ h ), i = 2,3, …, p.

Параметры , , подбираются так, чтобы значение совпадало со значением разложения в точке точного решения в ряд Тейлора с погрешностью O().

Для четвертого порядка:

a1 = 0, a2 = , a3 = , a4 = 1,

b21 = , b31 = 0, b32 = , b41 = 0, b42 = 0, b43 = ,

c1 = , c2 = , c3 = , c4 = .

Тогда:

, = ( + 2 + + ),

), ),

= hf(), = hf()

# **Определение параметров p и q**

Для вычисления параметров p и q была написана функция ***Newton***. Она принимает на вход ссылки на 6 функций (4 из них это производные по параметру p и q от первого и второго уравнения системы, оставшиеся 2 это сама система уравнений), начальные значения параметров p и q и переменную, обозначающую точность вычислений.

В этой программе используется функция ***Gaus***, принимающая вектор векторов (матрицу коэффициентов) и вектор (обозначающий значения выражений). В результате, функция ***Gaus*** возвращает нам вектор, показывающий на сколько будет меняться значение p и q на каждой итерации, по сравнению с предыдущей.

Результатом работы ***Newton***  будет вектор ***ans***, показывающей одну из возможных пар значений параметров p и q.

В данном случае при заданных начальных параметрах 0, 5 параметры принимают следующие значения:

**p** = 0.0794632

**q** = 2.23466

# **Решение задачи Коши для системы уравнений**

Начальные условия для системы:

(0) = 0.0794632,

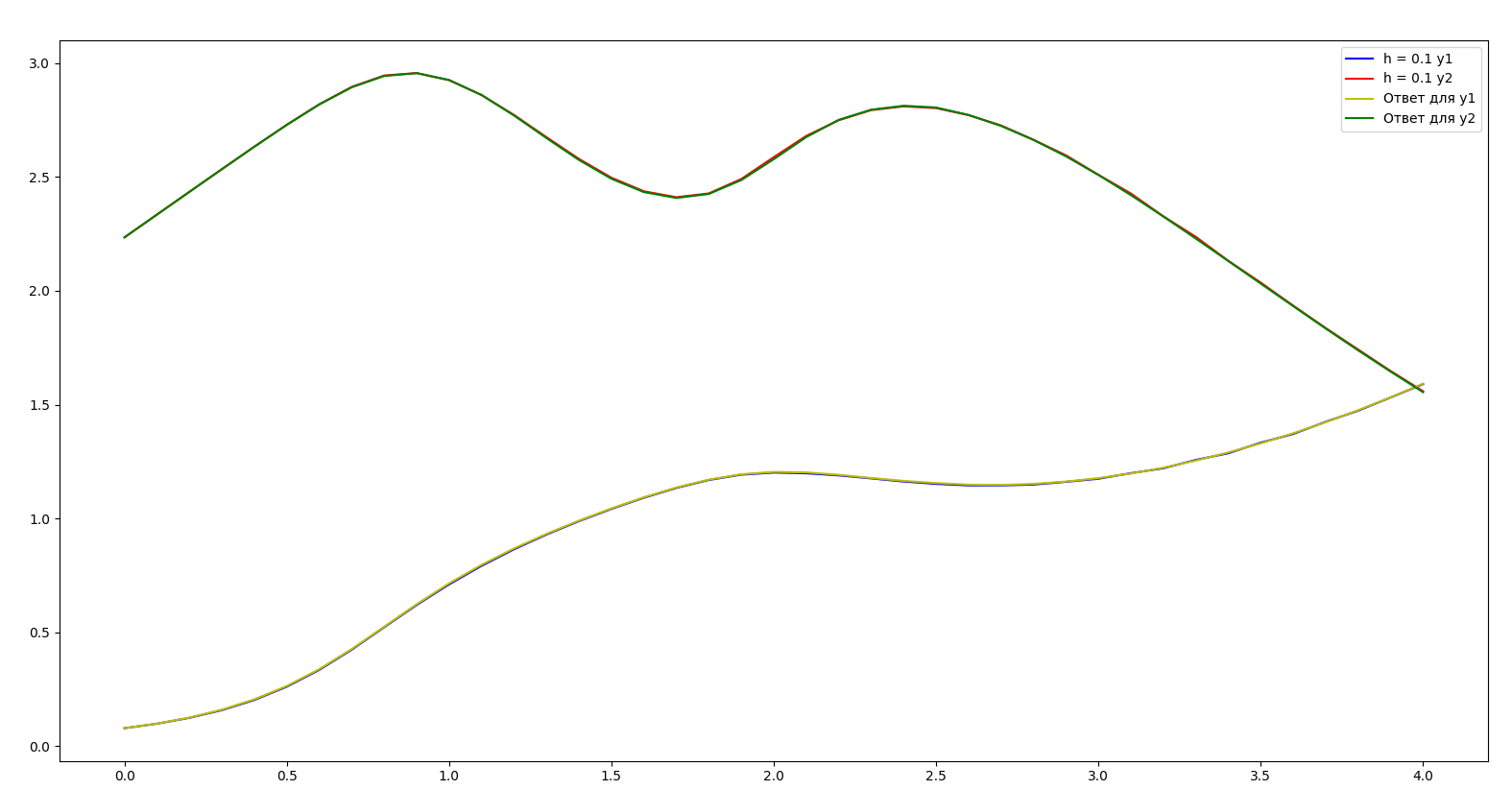
(0) = 2.23466;

Для решения поставленной задачи была написана функция ***Adams\_for\_Systems***. Она принимает на вход: крайние значения x, параметр точности, начальные значения , и функции, определяющие значения их производных.

***Adams\_for\_Systems*** имеет лишь небольшие отличия, по сравнению с функцией для решения задачи Коши для одного уравнения. Были измены функции для получения коэффициентов для метода Рунге-Кутты для работы с функциями от двух переменных. Сама же функция ***Adams\_for\_Systems*** вычисляет значения для поочередно.

В результате работы программы мы получаем пару массивов, в которых хранятся значения и . По этим значениям можно построить следующие графики:

Рисунок 1. Сравнение получившихся функций с результатом работы функции ***solve\_ivp***



Как видно из Рисунка 1, графики, полученные при работе программы ***Adams\_for\_Systems*** совпадают результатом работы функции ***solve\_ivp*** из библиотеки scipy.

# **Оценка погрешности**

Оценим погрешность решение с помощью правила Рунге. Для оценки требуется решить задачу на двух сетках, один раз с шагом h и второй с шагом h/2 . Формула имеет вид

Для метода Адамса p = 4, и формула принимает вид

Для погрешность будет равна 0.0378397.

А для 0.0677634.

**Заключение**

Передо мной была поставлена задача решить задачу Коши для системы обыкновенных дифференциальных уравнений. Для этого была написано 2 программы.

Первая находила начальные значения путем решения системы нелинейных уравнений методом Ньютона.

Вторая решала непосредственно задачу Коши методом Адамса.

Полученные мною решения были сравнены с результатом функции ***solve\_ivp*** из библиотеки scipy.

Также были проведены вычисления погрешность с помощью правила Рунге.

**Список использованной 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. Петров И.Б., Лобанов А.И. Лекции по вычислительной математике: учебное пособие. М.: БИНОМ. Лаборатория знаний. 2006
3. Бахвалов Н.С., Лапин А.В., Чижонков Е.В. Численные методы в задачах и упражнениях: учебное пособие, 2-е изд. М.: БИНОМ. Лаборатория знаний. 2010.
4. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. 7-е изд., М.: БИНОМ. Лаборатория знаний. 2011.

# **Приложение**

#include <iostream>

#include <cmath>

#include <vector>

#include <fstream>

#define \_USE\_MATH\_DEFINES

using namespace std;

void max\_el(int a, vector<vector <double>>& mass, vector<double>& values)

{

for (int i = a; i < mass.size(); i++)

{

if (abs(mass[a][a]) < abs(mass[i][a]))

{

swap(mass[a], mass[i]);

swap(values[a], values[i]);

}

}

}

vector<double> Gaus(vector<vector <double>> mass, vector<double> values)

{

vector<double> ans(mass[0].size());

double per;

double del;

double sum = 0;

for (int i = 0; i < mass.size() - 1; i++)

{

max\_el(i, mass, values);

per = mass[i][i];

for (int j = i + 1; j < mass.size(); j++)

{

del = -(mass[j][i] / per);

mass[j][i] = 0;

values[j] += values[i] \* del;

for (int k = i + 1; k < mass[0].size(); k++)

{

mass[j][k] += mass[i][k] \* del;

}

}

}

for (int i = mass.size() - 1; i >= 0; i--)

{

for (int j = i + 1; j < mass.size(); j++)

{

sum += mass[i][j] \* ans[j];

}

ans[i] = (values[i] - sum) / mass[i][i];

sum = 0;

}

return ans;

}

double eq1(double x1, double x2)

{

return x1\*x1+x2\*x2-5;

}

double eq2(double x1, double x2)

{

return tan(x1\*x2)-x1-0.1;

}

double eq\_P1(double x1, double x2)

{

return 2 \*x1;

}

double eq\_P2(double x1, double x2)

{

return 2\* x2;

}

double eq\_P3(double x1, double x2)

{

return x2/(cos(x1\*x2)\*cos(x1 \* x2))-1;

}

double eq\_P4(double x1, double x2)

{

return x1 / (cos(x1 \* x2) \* cos(x1 \* x2));

}

vector<double> Newton(double (\*f1)(double, double), double (\*f2)(double, double), double (\*f3)(double, double), double (\*f4)(double, double), double (\*f5)(double, double), double (\*f6)(double, double), double a, double b, double acc)

{

vector<double> ans{ 0,0 };

vector<double> lastans{ a,b };

bool accuracy = false;

vector<vector<double>> W;

vector<double> F;

vector<double> soul;

double temp;

while (accuracy != true)

{

W = { {f1(lastans[0],lastans[1]),f2(lastans[0],lastans[1])}, {f3(lastans[0],lastans[1]), f4(lastans[0],lastans[1])} };

F = { -f5(lastans[0],lastans[1]),-f6(lastans[0],lastans[1]) };

soul = Gaus(W, F);

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

{

temp = ans[i];

ans[i] = lastans[i] + soul[i];

lastans[i] = temp;

}

if ((abs(f5(ans[0], ans[1])) < acc) && (abs(f6(ans[0], ans[1])) < acc))

{

accuracy = true;

}

}

return ans;

}

double K1(double h, double x, double y1, double y2, double (\*f)(double, double, double))

{

return h \* f(x, y1,y2);

}

double K2(double h, double x, double y1, double y2, double (\*f)(double, double, double))

{

return h \* f(x + 0.5 \* h, y1 + 0.5 \* K1(h, x, y1,y2, f),y2);

}

double K3(double h, double x, double y1, double y2, double (\*f)(double, double, double))

{

return h \* f(x + 0.5 \* h, y1 + 0.5 \* K2(h, x, y1, y2, f),y2);

}

double K4(double h, double x, double y1, double y2, double (\*f)(double, double, double))

{

return h \* f(x + h, y1 + K3(h, x, y1, y2, f),y2);

}

pair<vector<double>, vector<double>> Adams\_for\_Systems(double x1, double x2, double h, double y10, double y20, double (\*f1)(double, double, double), double (\*f2)(double, double, double))

{

vector<double> dots1;

vector<double> dots2;

double x = x1;

double y1i = y10;

double y2i = y20;

dots1.push\_back(y1i);

dots2.push\_back(y2i);

for (int i = 0; i < 3; i++)

{

y1i = y1i + (1.0 / 6.0) \* (K1(h, x, y1i, y2i, f1) + 2 \* K2(h, x, y1i, y2i, f1) + 2 \* K3(h, x, y1i, y2i, f1) + K4(h, x, y1i, y2i, f1));

dots1.push\_back(y1i);

y2i = y2i + (1.0 / 6.0) \* (K1(h, x, y2i, y1i, f2) + 2 \* K2(h, x, y2i, y1i, f2) + 2 \* K3(h, x, y2i, y1i, f2) + K4(h, x, y2i, y1i, f2));

dots2.push\_back(y2i);

x += h;

}

int i = 3;

double c = double(h / 24);

while (x <= x2)

{

y1i += c \* (55 \* f1(x, dots1[i], dots2[i]) - 59 \* f1(x - h, dots1[i - 1], dots2[i - 1]) + 37 \* f1(x - 2 \* h, dots1[i - 2], dots2[i - 2]) - 9 \* f1(x - 3 \* h, dots1[i - 3], dots2[i - 3]));

y2i += c \* (55 \* f2(x, dots2[i], dots1[i]) - 59 \* f2(x - h, dots2[i - 1], dots1[i - 1]) + 37 \* f2(x - 2 \* h, dots2[i - 2], dots1[i - 2]) - 9 \* f2(x - 3 \* h, dots2[i - 3], dots1[i - 3]));

dots2.push\_back(y2i);

dots1.push\_back(y1i);

x += h;

i++;

}

return make\_pair(dots1,dots2);

}

double Y1(double x, double y1, double y2)

{

return sin(y1\*y2);

}

double Y2(double x, double y1, double y2)

{

return cos(x\*y1 \* y2);

}

int main()

{

double a = 0;

double b = 4;

double h = 0.1;

vector<double> ans;

ans = Newton(eq\_P1, eq\_P2, eq\_P3, eq\_P4, eq1, eq2, 5, 0, 0.00001);

cout << "Корни системы нелинейных уравнений, вычисленный методом Ньютона, равны: " << '\n';

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

{

cout << ans[i] << '\n';

}

pair<vector<double>, vector<double>> y = Adams\_for\_Systems(a, b, h, ans[0], ans[1], Y1, Y2);

pair<vector<double>, vector<double>> y2 = Adams\_for\_Systems(a, b, h/2, ans[0], ans[1], Y1, Y2);

std::ofstream out;

out.open("x1.txt");

double x = a;

if (out.is\_open())

{

out << x << "\n";

while (abs(x - b) > h / 10)

{

x += h;

out << x << "\n";

}

}

out.close();

double e = -1000;

out.open("y1\_1.txt");

if (out.is\_open())

{

for (int i = 0; i < size(y.first); i++)

{

out << y.first[i] << "\n";

e = max(e, abs((y.first[i] - y2.first[i]) / 15));

}

}

out.close();

cout << e << "\n";

e = -1000;

out.open("y1\_2.txt");

if (out.is\_open())

{

for (int i = 0; i < size(y.second); i++)

{

out << y.second[i] << "\n";

e = max(e, abs((y.second[i] - y2.second[i]) / 15));

}

}

out.close();

cout << e << "\n";

}

import matplotlib.pyplot as plt

import scipy

import math

from scipy.integrate import solve\_ivp

f = open('C:/Users/don20/source/repos/Kursach/x1.txt', 'r')

x1 = f.readlines()

x1 = list(map(float, x1))

f = open('C:/Users/don20/source/repos/Kursach/y1\_1.txt', 'r')

y1\_1 = f.readlines()

y1\_1 = list(map(float, y1\_1))

f = open('C:/Users/don20/source/repos/Kursach/y1\_2.txt', 'r')

y1\_2 = f.readlines()

y1\_2 = list(map(float, y1\_2))

def ode\_sys(x, Y1Y2):

y1 = Y1Y2[0]

y2 = Y1Y2[1]

dy1\_dt = math.sin(y1 \* y2)

dy2\_dt = math.cos(x \* y1 \* y2)

return [dy1\_dt, dy2\_dt]

num\_sol = solve\_ivp(ode\_sys, [0, 4], [0.0794632, 2.23466], method='RK23', dense\_output=True)

XY\_num\_sol = num\_sol.sol(x1)

y1\_num\_sol = XY\_num\_sol[0].T

y2\_num\_sol = XY\_num\_sol[1].T

plt.plot(x1, y1\_1, color='b', label='h = 0.1 y1')

plt.plot(x1, y1\_2, color='r', label='h = 0.1 y2')

plt.plot(x1, y1\_num\_sol, color='y', label='Ответ для y1')

plt.plot(x1, y2\_num\_sol, color='g', label='Ответ для y2')

plt.legend()

plt.show()