**Федеральное государственное бюджетное образовательное учреждение**

**высшего профессионального образования**

**«Санкт-Петербургский государственный политехнический университет»**

—

**Институт информационных технологий и управления**

**Кафедра «Распределенные вычисления и компьютерные сети»**

**Отчет по лабораторной работе №3**

**На тему: «Решение систем дифференциальных уравнений при помощи подпрограммы RKF45».   
Вариант 23.**

**по дисциплине «Вычислительная математика»**

Выполнил

студент гр. 23507/1 В.Б. Борисов

Руководитель

доц. Т.В. Леонтьева   
  
  
 ≪\_\_\_≫ \_\_\_\_\_\_\_\_\_\_ 201\_ г.

Санкт-Петербург

2015

Постановка задачи

Решить систему дифференциальных уравнений:

следующими способами с одним и тем же шагом печати :  
1) по программе RKF45 c EPS=0.0001;

2) методом семейства Рунге-Кутты:

с двумя постоянными шагами интегрирования:

a)

б) любой другой, позволяющий получить качественно верное решение.

Сравнить результаты.

Описание решения

Руководствуясь условию задания, мы вначале используем подпрограмму RKF45 чтобы решить систему, затем написав собственноручно семейства Рунге-Кутты, которые мы описываем в функции «main» и вызываем там же, решаем систему, полученные значения RKF45 и семейств Рунге-Кутты сравниваем.

При выполнении данной работы использовался язык C++, ОС Windows 8.1. Также было проведено тестирование работы программ.

Вывод результатов работы программы:

Результаты тестирования программы были выведены на экран (Результаты работы программы смотри ниже).

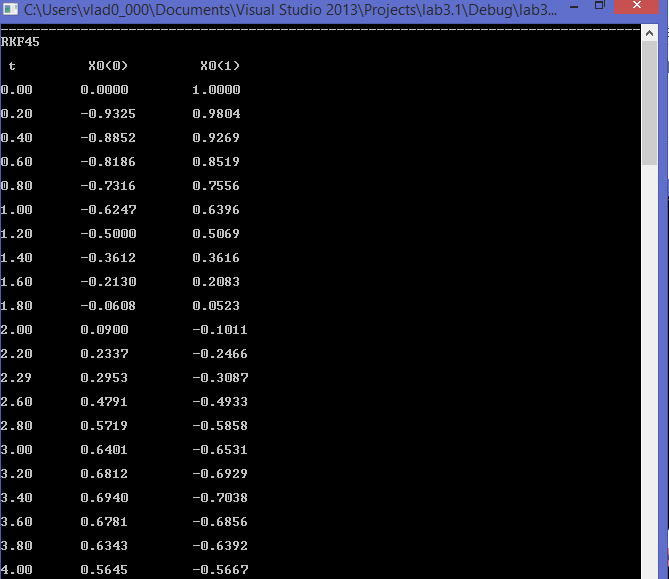


Рис 1.

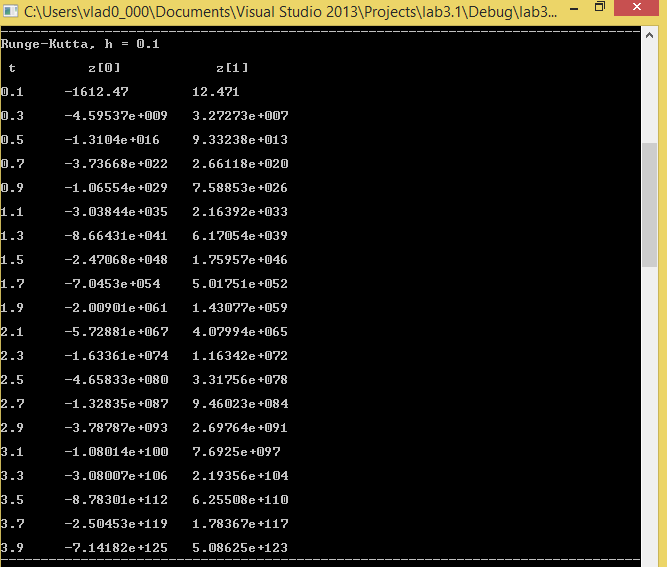
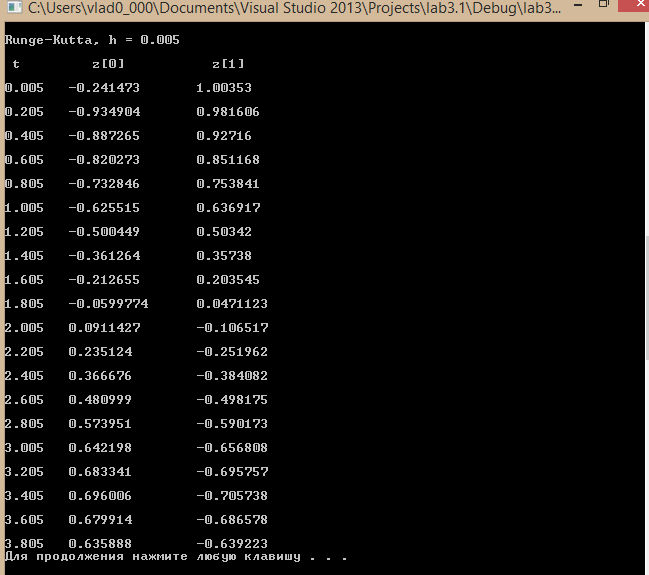


Рис. 2

  
 Рис. 3  
Анализ полученных результатов

Для заданной системы была протестирована программа RKF45. Также были самостоятельно реализованы семейства Рунге-Кутты.

Результаты работы программы показали, что подпрограмма RKF45 очень точно находит решения системы дифференциальных уравнений при корректном задании параметров вычисления. (продемонстрировано в Рис.1)

Для семейства Рунге-Кутты с шагом интегрирования 0,1 метод расходится (Рис. 2) Это связано с жесткостью системы. Т.к. она практически линейна, подсчитаем число обусловленности матрицы Это достаточно высокое число обусловленности. Поэтому шаг интегрирования нужно выбирать меньше. Возьмем 0.005 и шаг печати 0.2. Результат работы при таком шаге показан на (Рис.3). Как мы видим на Рис. 1 и Рис. 3, полученные результаты совпадают в 1-2 разрядах.

Исходный текст программы

**FILES CPP:**  
  
1) main.cpp

#include <iostream>

#include <cmath>

#include "rkf45.h"

using namespace std;

//Функция, в которую записана система дифференциальных ур-й

void function (double t, double \*x, double \*dxdt) {

dxdt[0] = -71 \* x[0] - 70 \* x[1] + exp(1-t\*t);

dxdt[1] = x[0] + sin(1 - t);

return;

}

int main() {

//Создание параметров RKF45

cout << "--------------------------------------------------------------------------------";

cout << "RKF45"<<'\n';

int N = 2; //кол-во уравнений

int iwork[30]; //рабочий массив размерности >=5

double work[15]; //рабочий массив размерности 6N+3

double X0[2] = { 0, 1}; //вектор решений

double T0 = 0; //начальная точка входа

double RELERR = 1.0e-10; // относительная погрешность

double ABSERR = 1.0e-10; //абслютная погрешность

int iflag = 1; //проверка корректности ответа

double tout = 0; // точка выхода

double h = 0.2; // шаг

for (int i = 0; i < 21; i++) {

RKF45 (function, N, X0, &T0, &tout, &RELERR, &ABSERR, &iflag, work, iwork);

printf("\nt=%.2f\tX0[0]=%.4f\tX0[1]=%.4f\n",T0,X0[0],X0[1]);

tout += h;

}

cout << "--------------------------------------------------------------------------------";

cout << "Runge-Kutta, h = 0.1" <<'\n';

//решение Рунге-Куттой с шагом 0.1

double z[2] = { 0, 1 };

double t = 0;

h = 0.1;

double k1[2], k2[2], k3[2], k4[2], k5[2], k6[2], dx[2], Temp[2];

for (int i = 0; i < 40; i++) {

//Нахождение k1

function(t, z, dx);

Temp[0] = z[0] + k1[0];

Temp[1] = z[1] + k1[1];

k1[0] = h\*dx[0];

k1[1] = h\*dx[1];

//Нахождение k2

Temp[0] = z[0] + k1[0] / 3;

Temp[1] = z[1] + k1[0] / 3;

function(t+h/3, Temp, dx);

k2[0] = h\*dx[0];

k2[1] = h\*dx[1];

//Нахождение k3

Temp[0] = z[0] + 0.16\*k1[0] + 0.24\*k2[0];

Temp[1] = z[1] + 0.16\*k1[0] + 0.24\*k2[0];

function(t+4/10\*h, Temp, dx);

k3[0] = h\*dx[0];

k3[1] = h\*dx[1];

//Нахождение k4

Temp[0] = z[0] + 0.25\*k1[0] - 3 \* k2[0] + 3.75\*k3[0];

Temp[1] = z[1] + 0.25\*k1[0] - 3 \* k2[0] + 3.75\*k3[0];

function(t+h, Temp, dx);

k4[0] = h\*dx[0];

k4[1] = h\*dx[1];

// Нахождение k5

Temp[0] = z[0] + (6 \* k1[0] + 90 \* k2[0] - 50 \* k3[0] + 8 \* k4[0]) / 81;

Temp[1] = z[1] + (6 \* k1[0] + 90 \* k2[0] - 50 \* k3[0] + 8 \* k4[0]) / 81;

function(t+2\*h/3, Temp, dx);

k5[0] = h\*dx[0];

k5[1] = h\*dx[1];

// Нахождение k6

Temp[0] = z[0] + (6 \* k1[0] + 36 \* k2[0] + 10 \* k3[0] + 8 \* k4[0]) / 75;

Temp[1] = z[1] + (6 \* k1[0] + 36 \* k2[0] + 10 \* k3[0] + 8 \* k4[0]) / 75;

function(t+4\*h/5, Temp, dx);

k6[0] = h\*dx[0];

k6[1] = h\*dx[1];

//Нахождение решения на i-ом шаге

t += h;

z[0] += (23 \* k1[0] + 125 \* k3[0] - 81 \* k5[0] + 125 \* k6[0]) / 192;

z[1] += (23 \* k1[1] + 125 \* k3[1] - 81 \* k5[1] + 125 \* k6[1]) / 192;

if (i % 2 == 0)

//printf("\n t z[0] z[1]");

cout << "\nt=" << t << "\tz[0]=" << z[0]<<"\tz[1]=" <<z[1]<<'\t'<<'\n';

}

//решение Рунге-Куттой 5-й степени с шагом 0.005

cout << "--------------------------------------------------------------------------------"<<'\n';

cout << "Runge-Kutta, h = 0.005" <<'\n';

z[0] = 0; z[1] = 1;

t = 0;

h = 0.005;

for (int i = 0; i < 800; i++) {

//Нахождение k1

function(t, z, dx);

Temp[0] = z[0] + k1[0];

Temp[1] = z[1] + k1[1];

k1[0] = h\*dx[0];

k1[1] = h\*dx[1];

//Нахождение k2

Temp[0] = z[0] + k1[0] / 3;

Temp[1] = z[1] + k1[0] / 3;

function(t+h/3, Temp, dx);

k2[0] = h\*dx[0];

k2[1] = h\*dx[1];

//Нахождение k3

Temp[0] = z[0] + 0.16\*k1[0] + 0.24\*k2[0];

Temp[1] = z[1] + 0.16\*k1[0] + 0.24\*k2[0];

function(t+4/10\*h, Temp, dx);

k3[0] = h\*dx[0];

k3[1] = h\*dx[1];

//Нахождение k4

Temp[0] = z[0] + 0.25\*k1[0] - 3 \* k2[0] + 3.75\*k3[0];

Temp[1] = z[1] + 0.25\*k1[0] - 3 \* k2[0] + 3.75\*k3[0];

function(t+h, Temp, dx);

k4[0] = h\*dx[0];

k4[1] = h\*dx[1];

// Нахождение k5

Temp[0] = z[0] + (6 \* k1[0] + 90 \* k2[0] - 50 \* k3[0] + 8 \* k4[0]) / 81;

Temp[1] = z[1] + (6 \* k1[0] + 90 \* k2[0] - 50 \* k3[0] + 8 \* k4[0]) / 81;

function(t+2\*h/3, Temp, dx);

k5[0] = h\*dx[0];

k5[1] = h\*dx[1];

// Нахождение k6

Temp[0] = z[0] + (6 \* k1[0] + 36 \* k2[0] + 10 \* k3[0] + 8 \* k4[0]) / 75;

Temp[1] = z[1] + (6 \* k1[0] + 36 \* k2[0] + 10 \* k3[0] + 8 \* k4[0]) / 75;

function(t+4\*h/5, Temp, dx);

k6[0] = h\*dx[0];

k6[1] = h\*dx[1];

//Нахождение решения на i-ом шаге

t += h;

z[0] += (23 \* k1[0] + 125 \* k3[0] - 81 \* k5[0] + 125 \* k6[0]) / 192;

z[1] += (23 \* k1[1] + 125 \* k3[1] - 81 \* k5[1] + 125 \* k6[1]) / 192;

if (i % 40 == 0)

printf("\nt=%.3f\tz[0]=%.4f\tz[1]=%.4f\n", t, z[0], z[1]);

}

system("pause");

}

**HEADER FILE:**

RKF45.h