|  |
| --- |
|  |

Факультет компьютерных и фундаментальных наук

Кафедра программного обеспечения вычислительной техники  
 и автоматизированных систем

ОТЧЕТ

о прохождении практики

студента группы ПО(аб)-81 Чекулаева В. Ю.

Направление обучения 09.03.04 Программная инженерия

Наименование практики: Учебная практика: практика по получению первичных профессиональных умений и навыков, в том числе первичных умений и навыков научно-исследовательской деятельности

Сроки прохождения практики: 09.02.20-13.06.20

Место прохождения практики: Тихоокеанский государственный университет,   
 кафедра программного обеспечения ВТ и АС

Руководитель практики от кафедры: доцент каф. ПОВТАС Вихтенко Э.М.

Руководитель практики от профильной организации:   
и.о.завкафедрой ПОВТАС Син А.З.

Оценка за практику:

Руководитель практики

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись ФИО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата

Хабаровск 2020 г.

**Содержание**

[Введение 3](#_Toc43381804)

[Постановка задачи 4](#_Toc43381805)

[Задание на практику 4](#_Toc43381806)

[Метод Рунге-Кутты 5](#_Toc43381807)

[Программное обеспечение 6](#_Toc43381808)

[Вывод исходной системы 7](#_Toc43381809)

[Результаты тестирования 8](#_Toc43381810)

[Результаты решения задачи 10](#_Toc43381811)

[Заключение 12](#_Toc43381812)

[Список используемой литературы 13](#_Toc43381813)

[Приложение А 14](#_Toc43381814)

### Введение

В ходе практической работы нам необходимо выполнить следующие пункты:

1) Написать программу решения задачи Коши для произвольной системы ОДУ заданным методом Рунге – Кутты.

2) Решить тестовую задачу с целью отладки написанной программы и экспериментального подтверждения теоретического порядка точности реализованного метода Рунге – Кутты.

3) Решить основную задачу — заданную задачу Коши, содержащую несколько параметров. Изучить влияние одного из параметров на качественные и количественные свойства решения.

В первом пункте задания дается краткое описание предмета моделирования, основных предположений и законов, положенных в основу модели. Этот пункт написан для того, чтобы дать некоторые представления о том, каким примерно образом была получена соответствующая математическая модель. Как правило, исходная модель записывается в терминах размерных зависимых и независимых переменных. Простым масштабированием (с математической точки зрения) от размерных переменных и уравнений осуществляется переход к безразмерным переменным и уравнениям (основной задаче), которые и решаются.

Стоит отметить, что числовые данные основных задач имеют математический смысл и подобраны так, чтобы обеспечить необходимые свойства решений.

### Постановка задачи

Клетки опухоли содержат особые вещества (антигены), которые вызывают резкую иммунную реакцию у больного. Эта реакция состоит в том, что производятся клетки — лимфоциты, которые атакуют и уничтожают клетки опухоли. Модель оперирует следующими переменными (имеется в виду плотность названной популяции клеток):

• L — свободные лимфоциты на поверхности опухоли;

• C — опухолевые клетки внутри опухоли и на ее поверхности;

• CS — опухолевые клетки на поверхности опухоли;

• CN — опухолевые клетки на поверхности опухоли, не связанные лимфоцитами;

• CF — опухолевые клетки внутри и на поверхности опухоли, не связанные лимфоцитами;

Ясно, что C = CF − CN + CS. Предполагается, что опухоль всегда имеет форму шара, так что CS = K1C2/3 , где K1 — постоянная, и что взаимодействие опухолевых клеток с лимфоцитами происходит только на поверхности опухоли. Будем считать, что между количеством свободных и связанных лимфоцитами клеток опухоли выполняется соотношение CS − CN = K2CNL (правдоподобно ли это предположение?). Из приведенных соотношений следует, что

CF = C − K1K2LC2/3/(1 + K2L), CN = K1C2/3/(1 + K2L), т. е. переменные L и C можно взять за основные переменные модели, которая сводится к системе уравнений

L′ = (−λ1 + α1CN (1 − L/LM))L,

C′ = λ2CF − α2CN L.

Здесь λ1 характеризует уровень естественной смертности лимфоцитов, следующее слагаемое — их стимуляцию: когда L мало, стимуляция свободных лимфоцитов возрастает линейно с ростом CN и что существует максимальный размер популяции LM, при котором стимуляция обращается в нуль. Первое слагаемое во втором уравнении описывает рост опухоли, не подвергающейся атакам лимфоцитов, а второй член учитывает взаимодействие свободных лимфоцитов с опухолевыми клетками на поверхности опухоли. Подставляя в уравнения значения CN и CF , можно переписать их в виде

x′ = ( −λ1 + β1y 2/3 (1 − x/c)/(1 + x) ) x,

y′ = λ2y − β2xy2/3/(1 + x), (1)

где x = K2L,c = K2LM, y = K1C, а λ1, λ2, β1, β2 — положительные параметры. Так как x и y — размеры популяций, они должны быть неотрицательными, а x не может превышать c, поскольку L ограничено сверху величиной LM. Уравнения (1) дополняются начальными условиями

x(0) = x0, y(0) = y0. (2)

### Задание на практику

1. Напишите программу интегрирования задачи Коши для системы из *n* уравнений первого порядка вида

, , ,

на произвольном отрезке *[a, b],* используя метод Рунге-Кутты третьего порядка точности с постоянным шагом *h*:

,

1. Протестируйте программу на примере системы уравнений

На отрезке [0, 5] c точным решением

*, .*

1. Для тестовой задачи постройте графики зависимости максимальной погрешности решения и от выбранного шага .
2. Для значений параметра β2 = 3; 3.48; 5 при помощи разработанной процедуры рассчитайте динамику популяции при различных начальных значениях размера опухоли y0 ∈ [0.5, 9]. Постройте графики наиболее характерных решений в координатах (x, y) и дайте их интерпретацию. Параметры: λ1 = λ2 = 1, β1 = 1, c = 3, t ∈ [0, 20].

### Метод Рунге-Кутты

Наиболее эффективными и часто встречаемыми методами решениями задачи Коши являются методы Рунге - Кутта. Они основаны на аппроксимации искомой функции *у(х)* в пределах каждого шага многочленом, который получен при помощи разложения функции *у(х)* в окрестности шага *h* каждой *i*-ой точки в ряд Тейлора:

*u( + h) = u() + hu′() + u′′( + h), ∈ [0,1].*

Пусть требуется найти приближенное решение задачи *u′(t) = f(t,u(t)), t ∈ (a,b], u(a)=*, на равномерной с шагом h сетке узлов.

Поскольку в силу уравнения справедливо равенство *u′() = f(,u()),* то соотношение (5) перепишется в виде

*= + hf(,) + h,*

*= ,*

*u′′( + h).*

В этой формуле слагаемое = O() является малой величиной, если *h* достаточно мало. Отбрасывая его, придем к методу Эйлера*: = + hf(,),* *i* = 0,1,...,N −1, = . Эти соотношения позволяют вычислить приближенное решение в точке сетки , зная приближенное решение в предыдущей точке . Такие численные методы называются одношаговыми.

Методы Рунге – Кутты при *q* = 3: Формулы имеют вид

yi+1 = yi + h (b1k1 + b2k2 + b3k3),

где

k1 = f(ti, yi),

k2 = f(ti + c2h, yi + a21hk1),

k3 = f(ti + c3h, yi + a31hk1 + a32hk2).

Аналогично тому, как это было сделано выше, получается следующая система из шести уравнений для определения восьми коэффициентов метода при m = 3:

c2 = a21,

c3 = a31 + a32,

b1 + b2 + b3 = 1,

2(c2b2 + c3b3) = 1,

,

.

Эта система имеет два семейства решений: двухпараметрическое со свободными параметрами c2 и c3, причем, c2 c3 и c2 2/3, и однопараметрическое со свободным параметром a32 (при c2 = c3 = 2/3). В качестве примера укажем коэффициенты при c2 = 1/2, c3 = 1: b1 = b3 = 1/6, b2 = 4/6, a21 = 1/2, a31 = −1, a32 = 2.

В задаче с моим вариантом требуется написать программу, используя метод Рунге – Кутты 3-го порядка точности с постоянным шагом h:

,

### Программное обеспечение

Программа была написана на языке С# в интегрированной среде разработки Visual Studio, так как программа содержит визуальные компоненты, при желании она может быть запущена на любом компьютере под управлением Windows и с установленной платформой .NET.

Для реализации программы было написано несколько функций, код которых можно увидеть в приложении А.

### Результаты тестирования

В ходе решения тестовой задачи мы задаем отрезок [a,b] и шаг h.

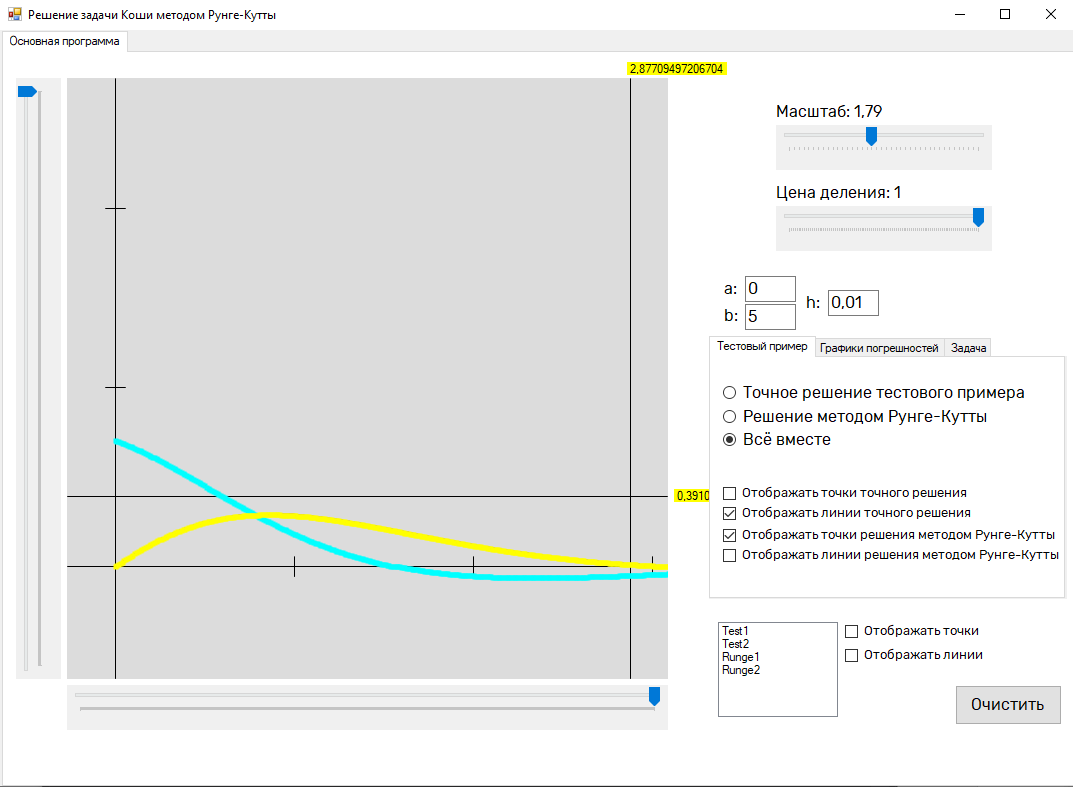


Рис. 1 – Решение тестовой задачи с шагом h=0.01

В ходе решения тестовой задачи мы задаем отрезок [a,b] и шаг h.

При шаге h=0.01 погрешность вычислений достаточно мала. Это видно по точкам, которые лежат на графиках точного решения задачи (рис. 1).

Однако если увеличить шаг сетки h в 5 (рис. 2) или в 10 (рис. 3) раз, то погрешность вычислений возрастает. Некоторые точки уже не лежат на графиках функций.

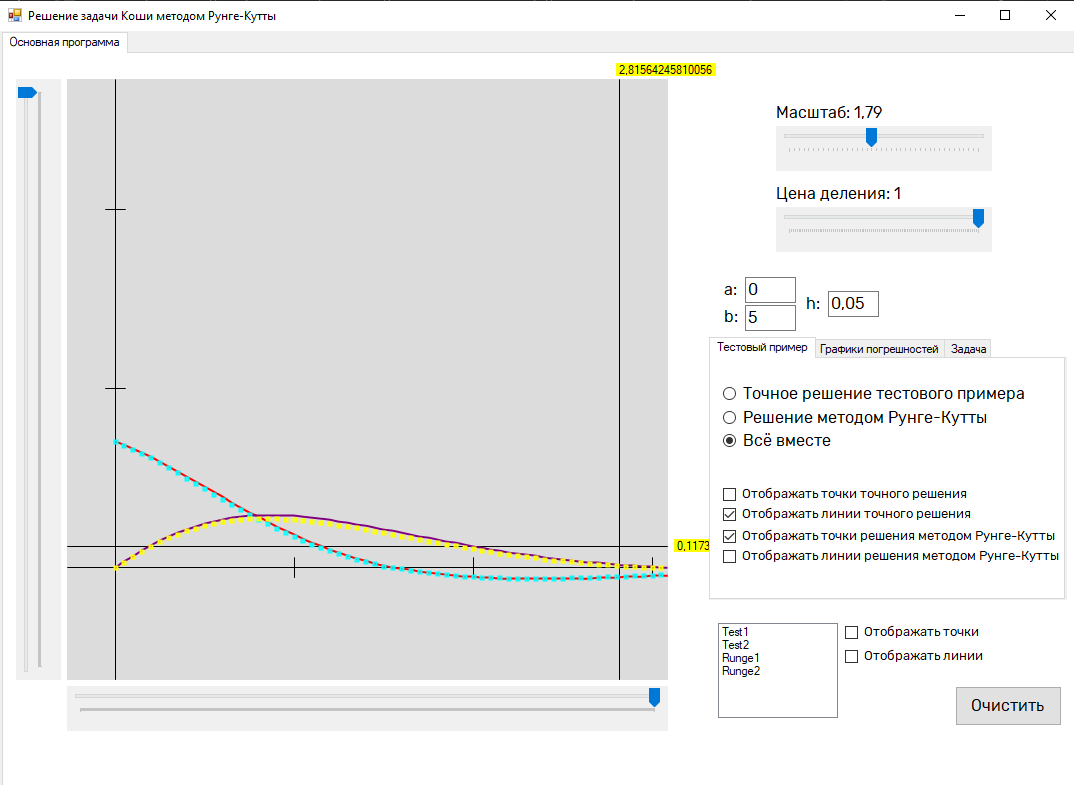


Рис. 2 – Решение тестовой задачи с шагом h=0.05

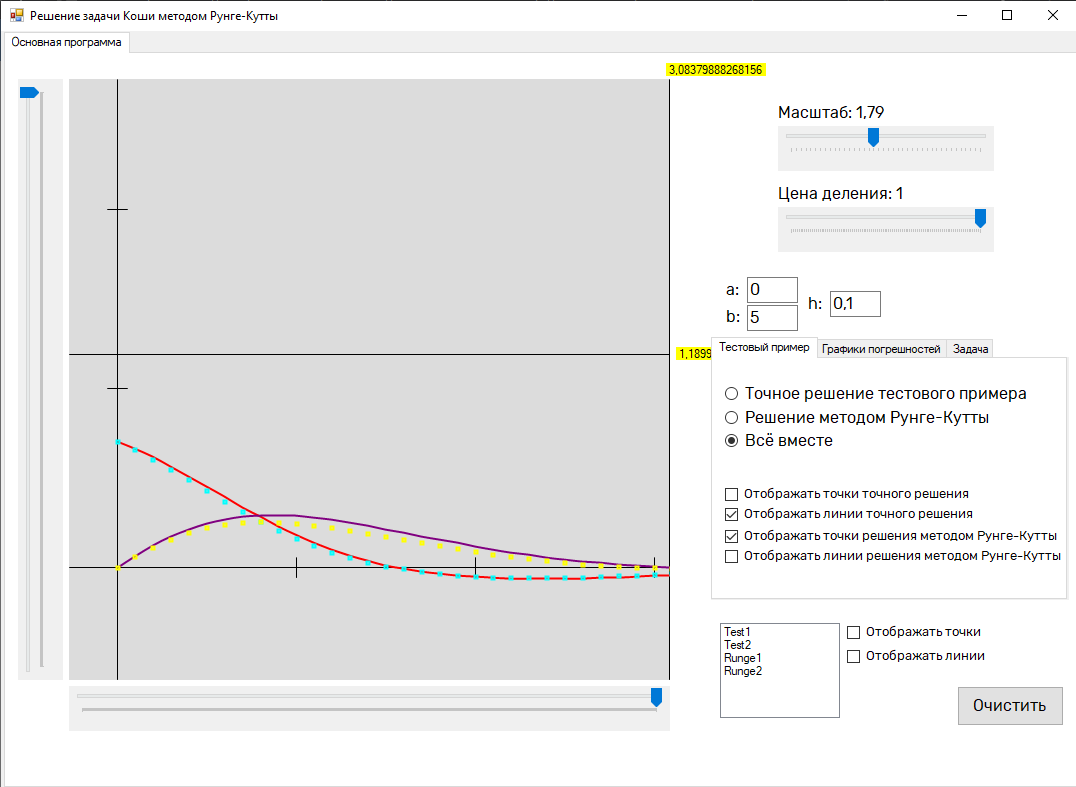


Рис. 3 – Решение тестовой задачи с шагом h=0.1

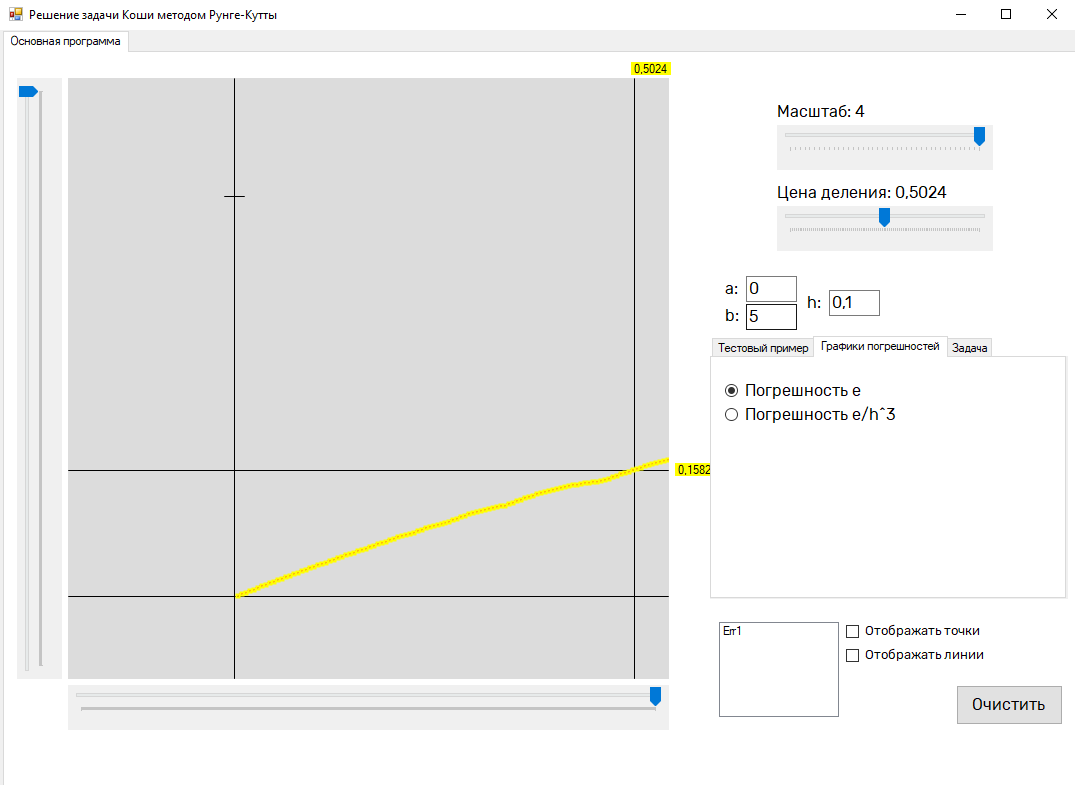


Рис.4 – График зависимости максимальной погрешности решения *e* от выбранного шага *h*

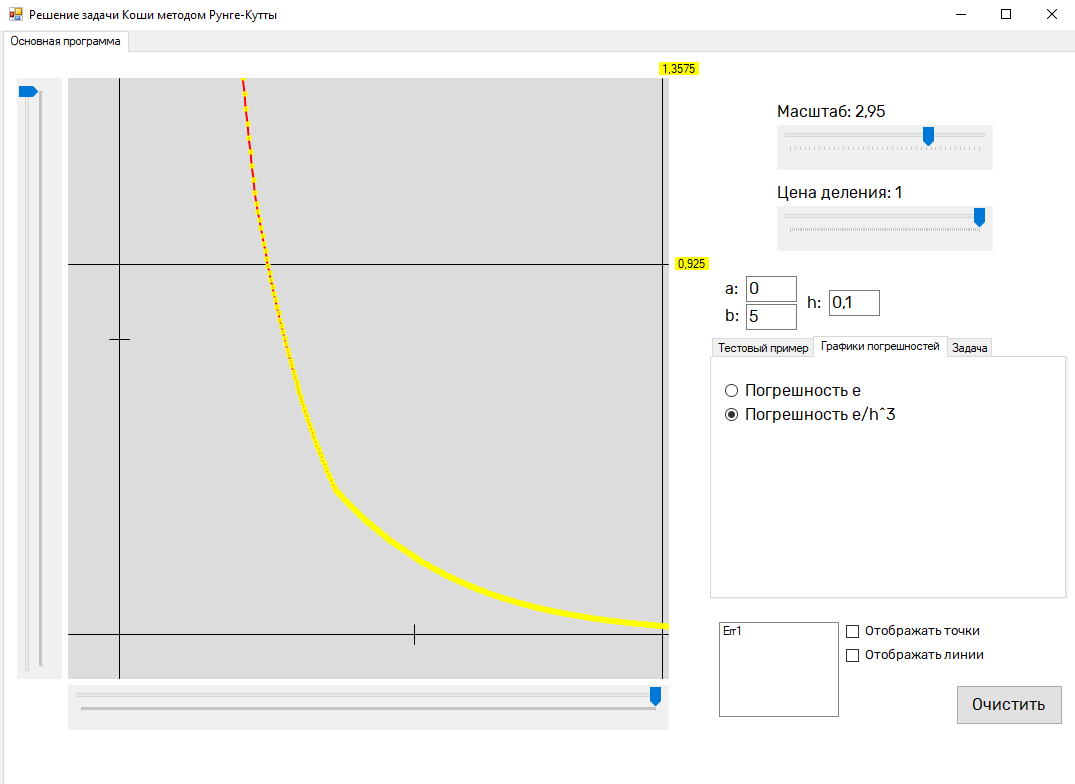
**

Рис.5 – График зависимости максимальной погрешности решения *e/* от выбранного шага *h*

Из графиков, приведенных на рисунках 5-6, можно сделать вывод о том, что порядок точности метода Рунге-Кутты, описанного в программе, соответствует заявленному порядку точности (третий порядок точности).

### Результаты решения задачи

При решении системы уравнений мы можем изменять только шаг сетки h, так как остальные параметры заданы.

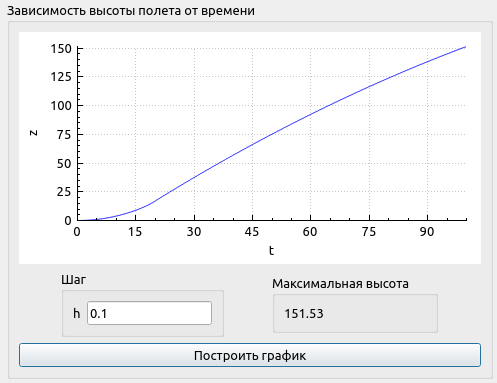


Рис. 6 – Результат решения заданной системы уравнений с шагом *h=*0.1

При уменьшении шага сетки погрешность решения уменьшается, численное решение приближается к некоторому точному решению (рис. 7, 8).

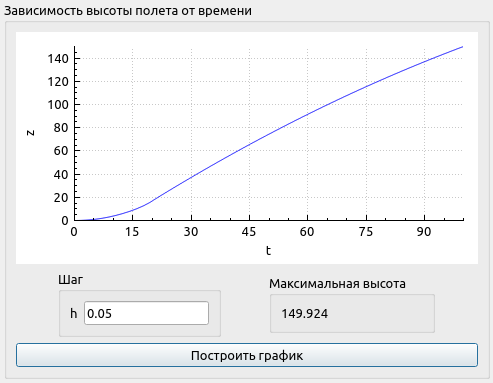


Рис. 7 – Результат решения заданной системы уравнений с шагом *h=*0.05

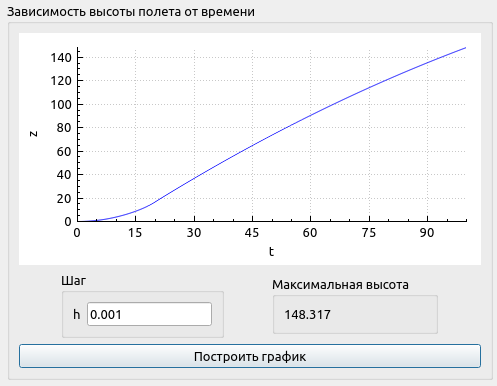


Рис. 8 – Результат решения заданной системы уравнений с шагом *h=*0.01

При решении системы уравнений мы можем изменять только шаг сетки h, так как остальные параметры заданы.

Так же дано задание по нахождению величины такой, чтобы максимизировать высоту подъема ракеты за время *T=*100. Результат его выполнения показан на рисунке 9.

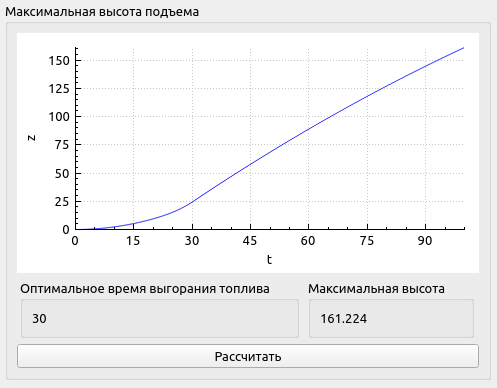


Рис. 9 – Результат нахождения оптимального времени выгорания топлива за время *T=*100

При решении этой задачи просматриваются все решения системы со значениями , изменяющимися от 3 до 30 с шагом 1. Шаг сетки не меняется и равен h=0.01.

### Заключение

В ходе практической работы рассмотрено решение задачи Коши методом Рунге-Кутты 2 порядка точности, примененное для решения задачи оптимизации вертикального полета ракеты. Метод Рунге-Кутты реализован в программе, проведены расчеты на тестовом примере. В ходе вычислительных экспериментов найдено значение , при котором ракета поднимется на максимальную высоту за время *T=*100.

### Список используемой литературы

1. Федоренко Р. П. Приближенное решение задач оптимального управления. – М.: Наука, 1978.
2. Хайрер Э., Нерсетт С., Ваннер Г. Решение обыкновенных дифференциальных уравнений. – М.: Мир, 1990.
3. Даутов, Р.З. Практикум по курсу численные методы. Решение задачи Коши для системы ОДУ / P.З. Даутов. – Изд-во Казанск. федер. ун-та, 2014 г. – 100 с.
4. Холодниок М., Клич А., Кубичек М., Марек М. Методы анализа нелинейных динамических моделей. — М.: Мир, 1991.
5. Самарский А.А., Гулин А.В. Численные методы.— М.:Наука,1989. 4. Бахвалов Н.С., Жидков Н.П., Кобельков Г.М. Численные методы. — М.: Наука, 1987.

### Приложение А

Код программы

#include "zadacha.h"

#include "ui\_zadacha.h"

#include"qcustomplot.h"

#include<functional>

#include<QPushButton>

#include<QVBoxLayout>

#include<QHBoxLayout>

#include<cmath>

#include<QGroupBox>

#include<algorithm>

#include<QLabel>

#include<QLineEdit>

#include<cstdlib>

using func\_t = std::function<double(int, double, QVector<double>)>;

static double g\_t\_s{20};

Zadacha::Zadacha(QWidget \*parent, bool flag) :

QDialog(parent),

ui(new Ui::Zadacha)

{

ui->setupUi(this);

if(flag){

this->setMinimumWidth(900);

this->setMinimumHeight(900);

QVBoxLayout\* mainLay = new QVBoxLayout(this);

QVBoxLayout\* box1Lay = new QVBoxLayout;

QVBoxLayout\* box2Lay = new QVBoxLayout;

QHBoxLayout\* gr\_err = new QHBoxLayout;

QHBoxLayout\* box11Lay = new QHBoxLayout;

QHBoxLayout\* box12Lay = new QHBoxLayout;

QHBoxLayout\* under = new QHBoxLayout;

QPushButton\* testB = new QPushButton("Построить график");

connect(testB, SIGNAL(clicked()), this, SLOT(testButton\_clicked()));

QGroupBox\* box1 = new QGroupBox("Результат работы программы(точки) и точное решение(линии)");

QCustomPlot\* graph = new QCustomPlot;

graph->xAxis->setLabel("t");

graph->yAxis->setLabel("u(t)");

graphics = graph;

box1Lay->addWidget(graph);

QGroupBox\* box11 = new QGroupBox("Границы");

box11->setSizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);

QHBoxLayout\* borders = new QHBoxLayout;

QLabel\* l1 = new QLabel("a:");

QLabel\* l2 = new QLabel("b:");

QLineEdit\* first = new QLineEdit("0");

start\_p = first;

QLineEdit\* second = new QLineEdit("2");

finish\_p = second;

borders->addWidget(l1);

borders->addWidget(first);

borders->addWidget(l2);

borders->addWidget(second);

box11Lay->addLayout(borders);

box11->setLayout(box11Lay);

QGroupBox\* box12 = new QGroupBox("Шаг");

box12->setSizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);

QLabel\* l3 = new QLabel("h");

QLineEdit\* stp = new QLineEdit("0.02");

step = stp;

box12Lay->addWidget(l3);

box12Lay->addWidget(stp);

box12->setLayout(box12Lay);

under->addWidget(box11);

under->addWidget(box12);

box1Lay->addLayout(under);

box1Lay->addWidget(testB);

box1->setLayout(box1Lay);

QGroupBox\* box2 = new QGroupBox("Графики зависимостимаксимальной погрешности решения e и e/h^2 от выбранного шага h");

QPushButton\* errorsB = new QPushButton("Построить графики");

connect(errorsB, SIGNAL(clicked()), this, SLOT(errorsButton\_clicked()));

QCustomPlot\* grErr1 = new QCustomPlot;

grErr1->xAxis->setLabel("h");

grErr1->yAxis->setLabel("e");

QCustomPlot\* grErr2 = new QCustomPlot;

grErr2->xAxis->setLabel("h");

grErr2->yAxis->setLabel("e/h^2");

maxErr1 = grErr1;

maxErr2 = grErr2;

gr\_err->addWidget(grErr1);

gr\_err->addWidget(grErr2);

box2Lay->addLayout(gr\_err);

box2Lay->addWidget(errorsB);

box2->setLayout(box2Lay);

mainLay->addWidget(box1);

mainLay->addWidget(box2);

mainLayout = mainLay;

setLayout(mainLay);

} else{

this->setMinimumWidth(1000);

this->setMinimumHeight(400);

QHBoxLayout\* mainLay = new QHBoxLayout(this);

QVBoxLayout\* boxGr1 = new QVBoxLayout;

QVBoxLayout\* boxGr2 = new QVBoxLayout;

QHBoxLayout\* box2Lay = new QHBoxLayout;

QHBoxLayout\* under1 = new QHBoxLayout;

QHBoxLayout\* under2 = new QHBoxLayout;

QPushButton\* osnB1 = new QPushButton("Построить график");

connect(osnB1, SIGNAL(clicked()), this, SLOT(osnButton1\_clicked()));

QGroupBox\* box2 = new QGroupBox("Шаг");

box2->setSizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);

QLabel\* l3 = new QLabel("h");

QLineEdit\* stp = new QLineEdit("0.02");

step = stp;

box2Lay->addWidget(l3);

box2Lay->addWidget(stp);

box2->setLayout(box2Lay);

//under1->addWidget(box1);

under1->addWidget(box2);

QCustomPlot\* graph = new QCustomPlot;

graph->xAxis->setLabel("t");

graph->yAxis->setLabel("z");

graphics = graph;

QVBoxLayout\* maximum1 = new QVBoxLayout;

QGroupBox\* boxMaxZ1 = new QGroupBox("Максимальная высота");

boxMaxZ1->setSizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);

QLabel\* l5 = new QLabel("0");

l5->setSizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);

max\_z1 = l5;

maximum1->addWidget(l5);

boxMaxZ1->setLayout(maximum1);

under1->addWidget(boxMaxZ1);

boxGr1->addWidget(graph);

boxGr1->addLayout(under1);

boxGr1->addWidget(osnB1);

QGroupBox\* gr1 = new QGroupBox("Зависимость высоты полета от времени");

gr1->setLayout(boxGr1);

QCustomPlot\* graph2 = new QCustomPlot;

maximum\_z = graph2;

graph2->xAxis->setLabel("t");

graph2->yAxis->setLabel("z");

QPushButton\* osnB2 = new QPushButton("Рассчитать");

connect(osnB2, SIGNAL(clicked()), this, SLOT(osnButton2\_clicked()));

QGroupBox\* gr2 = new QGroupBox("Максимальная высота подъема");

gr2->setLayout(boxGr2);

QGroupBox\* optimal = new QGroupBox("Оптимальное время выгорания топлива");

optimal->setSizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);

QVBoxLayout\* opt = new QVBoxLayout;

QLabel\* opt\_l = new QLabel("0");

optimal\_t\_s = opt\_l;

opt->addWidget(opt\_l);

optimal->setLayout(opt);

QGroupBox\* boxMaxZ2 = new QGroupBox("Максимальная высота");

boxMaxZ2->setSizePolicy(QSizePolicy::Fixed, QSizePolicy::Fixed);

QVBoxLayout\* maxZ2Lay = new QVBoxLayout;

QLabel\* maxZ2 = new QLabel("0");

max\_z2 = maxZ2;

maxZ2Lay->addWidget(maxZ2);

boxMaxZ2->setLayout(maxZ2Lay);

under2->addWidget(optimal);

under2->addWidget(boxMaxZ2);

boxGr2->addWidget(graph2);

boxGr2->addLayout(under2);

boxGr2->addWidget(osnB2);

mainLay->addWidget(gr1);

mainLay->addWidget(gr2);

setLayout(mainLay);

}

}

Zadacha::~Zadacha()

{

delete ui;

}

double primer\_f(int id, double t, QVector<double> y){

using namespace std;

switch (id){

case 0:

return y[0]/(2+2\*t) - 2\*t\*y[1];

break;

case 1:

return y[1]/(2+2\*t) + 2\*t\*y[0];

break;

default:

return 0;

break;

}

}

QVector<double> solution\_f(int id, QVector<double> t){

using namespace std;

QVector<double> ans;

switch(id){

case 0:

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

ans.push\_back(cos(t[i]\*t[i])\*sqrt(1+t[i]));

}

break;

case 1:

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

ans.push\_back(sin(t[i]\*t[i])\*sqrt(1+t[i]));

}

break;

default:

break;

}

return ans;

}

double osn\_f(int id, double t, QVector<double> y){

double g{0.01};

double alpha{2};

double c{0.05};

double gamma{0.01};

double m\_t{0.8};

double q;

if(t <= g\_t\_s){

q = m\_t/g\_t\_s;

} else{

q = 0;

}

switch(id){

case 0:

return -1\*q;

break;

case 1:

return y[2];

break;

case 2:

return -1\*g + (1/y[0])\*(alpha\*q - c\*exp(-1\*gamma\*y[1]\*y[1])\*y[2]\*y[2]);

break;

default:

return 0;

break;

}

}

QVector<double> addToElem(double d, QVector<double> v){

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

v[i] += d;

}

return v;

}

QVector<QVector<double>> runge\_kutt(func\_t f, QVector<QVector<double>> y, QVector<double> t, double h){ //в у содержатся стартовые значения

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

y.push\_back(QVector<double>());

for(int j = 0; j < y[i].size(); ++j){

double k\_1 = f(j, t[i], y[i]);

double k\_2 = f(j, t[i] + (h/2), addToElem(((h/2)\*k\_1), y[i]));

y[i+1].push\_back(y[i][j] + h\*(k\_1 + k\_2)/2);

}

}

QVector<QVector<double>> transp(y[0].size(), QVector<double>(y.size()));

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

for(int j = 0; j < y[i].size(); ++j){

transp[j][i] = y[i][j];

}

}

return transp;

}

void show\_graphs(QVector<double> t, QVector<QVector<double>> dots, QVector<QVector<double>> solution, QCustomPlot\* p){

int k{0};

std::srand(3213);

QVector<QColor> colors;

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

p->addGraph();

p->graph(i)->setData(t, dots[i]);

p->graph(i)->setLineStyle(QCPGraph::lsNone);

colors.push\_back(QColor(std::rand()%255, std::rand()%255, std::rand()%255));

p->graph(i)->setPen(colors.last());

p->graph(i)->setScatterStyle(QCPScatterStyle(QCPScatterStyle::ssCircle, 4));

k++;

}

for(int j = 0; j < solution.size(); j++){

p->addGraph();

p->graph(k)->setData(t, solution[j]);

p->graph(k)->setPen(colors[j]);

k++;

}

p->xAxis->setRange(0, t.last());

p->yAxis->setRange(-10, 10);

p->replot();

}

void Zadacha::max\_errors(func\_t f, std::function<QVector<double>(int, QVector<double>)> solution\_f, double a, double b, int n){// n - кол-во уравнений

QVector<double> ans1, ans2, h\_v;

double h = 0.09;

while(h > 0.0001){

QVector<double> t;

for(double i = a; i <= b; i+=h){

t.push\_back(i);

}

QVector<QVector<double>> solution;

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

solution.push\_back(solution\_f(i, t));

}

QVector<QVector<double>> y; y.push\_back(QVector<double>());

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

y[0].push\_back(solution[i][0]);

}

auto dots{runge\_kutt(f, y, t, h)};

QVector<double> norms;

for(int i = 0; i < dots[0].size(); ++i){

for(int j = 0; j < dots.size(); ++j){

norms.push\_back(std::abs(solution[j][i]-dots[j][i]));

}

}

ans1.push\_back(\*std::max\_element(norms.begin(), norms.end()));

h\_v.push\_back(h);

ans2.push\_back(ans1.last()/(h\*h));

h /= 1.2;

}

maxErr1->addGraph();

maxErr1->yAxis->setRange(0, \*std::max\_element(ans1.begin(), ans1.end()));

maxErr1->graph(0)->setData(h\_v, ans1);

maxErr1->graph(0)->setScatterStyle(QCPScatterStyle(QCPScatterStyle::ssCircle, 4));

maxErr1->xAxis->setRange(0, h\_v[0]);

maxErr2->addGraph();

maxErr2->graph(0)->setData(h\_v, ans2);

maxErr2->graph(0)->setScatterStyle(QCPScatterStyle(QCPScatterStyle::ssCircle, 4));

maxErr2->xAxis->setRange(0, h\_v[0]);

maxErr2->yAxis->setRange(0, 13500);

maxErr1->replot();

maxErr2->replot();

}

void Zadacha::testButton\_clicked(){

double a{start\_p->text().toDouble()}, b{finish\_p->text().toDouble()}, h{step->text().toDouble()};

int n{2};

QVector<double> t;

for(double i = a; i <= b; i+=h){

t.push\_back(i);

}

QVector<QVector<double>> solution;

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

solution.push\_back(solution\_f(i, t));

}

QVector<QVector<double>> y; y.push\_back(QVector<double>());

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

y[0].push\_back(solution[i][0]);

}

show\_graphs(t, runge\_kutt(primer\_f, y, t, h), solution, graphics);

}

void Zadacha::errorsButton\_clicked(){

double a{start\_p->text().toDouble()}, b{finish\_p->text().toDouble()};

int n{2};

max\_errors(primer\_f, solution\_f, a, b, n);

}

void Zadacha::osnButton1\_clicked(){

double a{0}, b{100}, h{step->text().toDouble()};

double m\_0{1}, z\_0{0}, v\_0{0};

QVector<double> t;

for(double i = a; i <= b; i+=h){

t.push\_back(i);

}

QVector<QVector<double>> y; y.push\_back(QVector<double>());

y[0].push\_back(m\_0);

y[0].push\_back(z\_0);

y[0].push\_back(v\_0);

auto ans = runge\_kutt(osn\_f, y, t, h);

max\_z1->setText(QString::number(\*std::max\_element(ans[1].begin(), ans[1].end())));

graphics->addGraph();

graphics->graph(0)->setData(t, ans[1]);

graphics->xAxis->setRange(a, b);

graphics->yAxis->setRange(0, max\_z1->text().toDouble());

graphics->replot();

}

void Zadacha::osnButton2\_clicked(){

double a{0}, b{100}, h{0.001};

double m\_0{1}, z\_0{0}, v\_0{0};

QVector<double> t;

for(double i = a; i <= b; i+=h){

t.push\_back(i);

}

QVector<QVector<double>> y; y.push\_back(QVector<double>());

y[0].push\_back(m\_0);

y[0].push\_back(z\_0);

y[0].push\_back(v\_0);

double m\_z = std::numeric\_limits<double>::min();

double mem\_t{0};

for(double i = 3; i <= 30; ++i){

g\_t\_s = i;

auto ans = runge\_kutt(osn\_f, y, t, h);

double tmp = \*std::max\_element(ans[1].begin(), ans[1].end());

if(tmp > m\_z){

m\_z = tmp;

mem\_t = i;

}

}

g\_t\_s = mem\_t;

auto fin = runge\_kutt(osn\_f, y, t, h);

max\_z2->setText(QString::number(\*std::max\_element(fin[1].begin(), fin[1].end())));

maximum\_z->addGraph();

maximum\_z->graph(0)->setData(t, fin[1]);

maximum\_z->yAxis->setRange(0, max\_z2->text().toDouble());

maximum\_z->xAxis->setRange(a, b);

maximum\_z->replot();

optimal\_t\_s->setText((QString::number(g\_t\_s)));

g\_t\_s = 20;

}