Министерство образования и науки Российской Федерации

Санкт-Петербургский политехнический университет Петра Великого

—

Институт компьютерных наук и кибербезопастности

Высшая школа программной инженерии

**Лабораторная работа №3**

**по дисциплине «Вычислительная математика»**

Выполнил

Студент группы 5130904/20004 Машкин А.А.

Преподаватель Устинов С.М.

Оглавление

[Задание 2](#_Toc163212483)

[Результаты 3](#_Toc163212484)

[Вывод 4](#_Toc163212485)

[Код программы 5](#_Toc163212486)

[<DIR>/computational\_mathematics/lab\_2/main.cpp 5](#_Toc163212487)

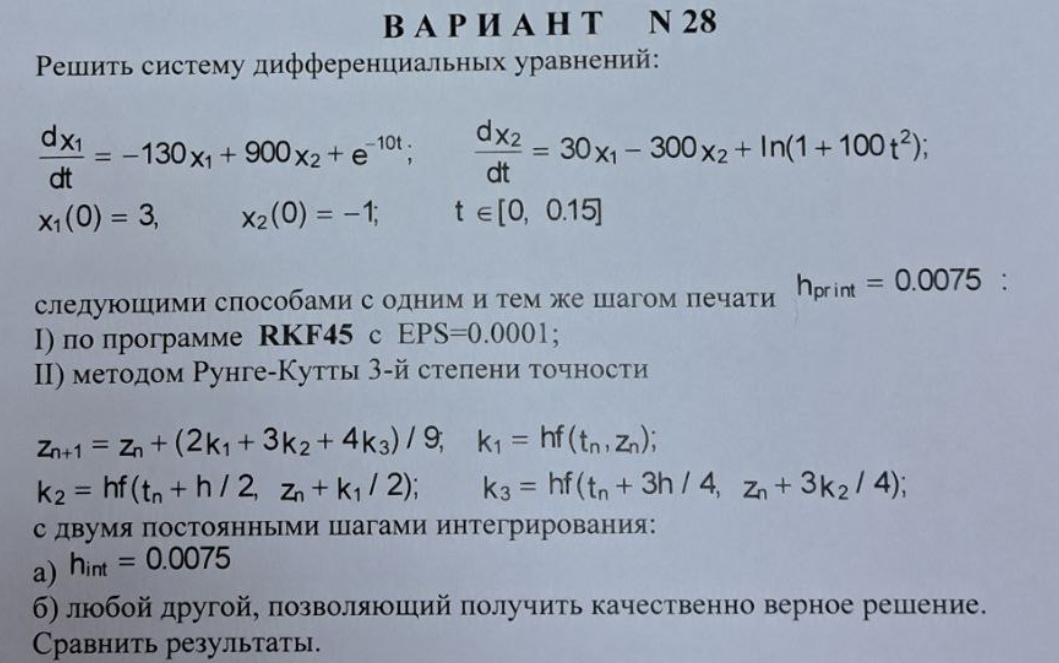
[<DIR>/computational\_mathematics/lab\_2/solve.h 6](#_Toc163212488)

[<DIR>/computational\_mathematics/lab\_2/solve.cpp 6](#_Toc163212489)

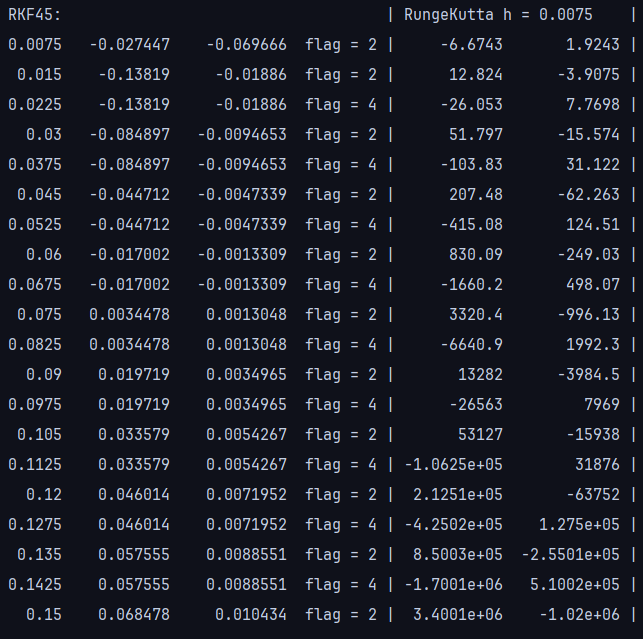
[<DIR>/computational\_mathematics/lab\_2/matrix\_function.h 7](#_Toc163212490)

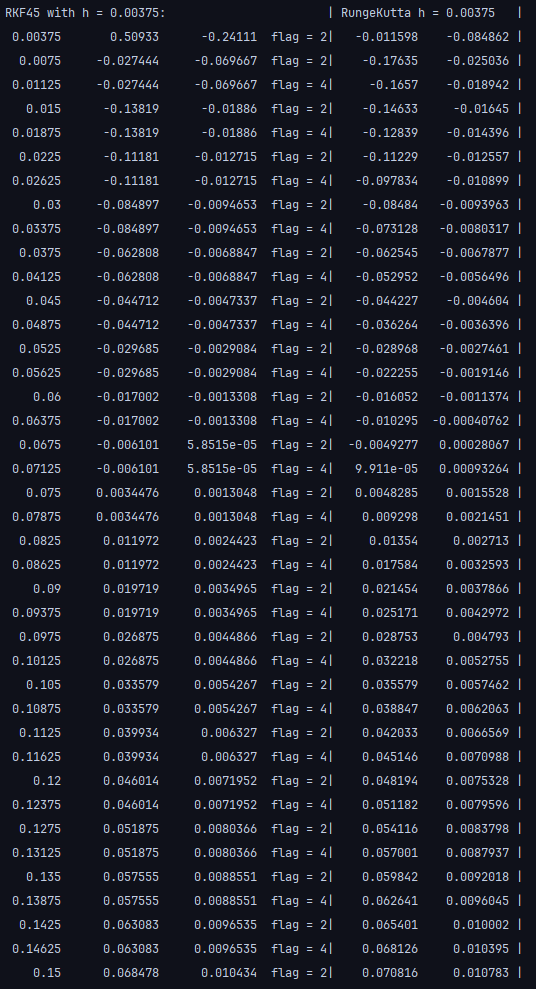
[<DIR>/computational\_mathematics/lab\_2/matrix\_function.cpp 7](#_Toc163212491)

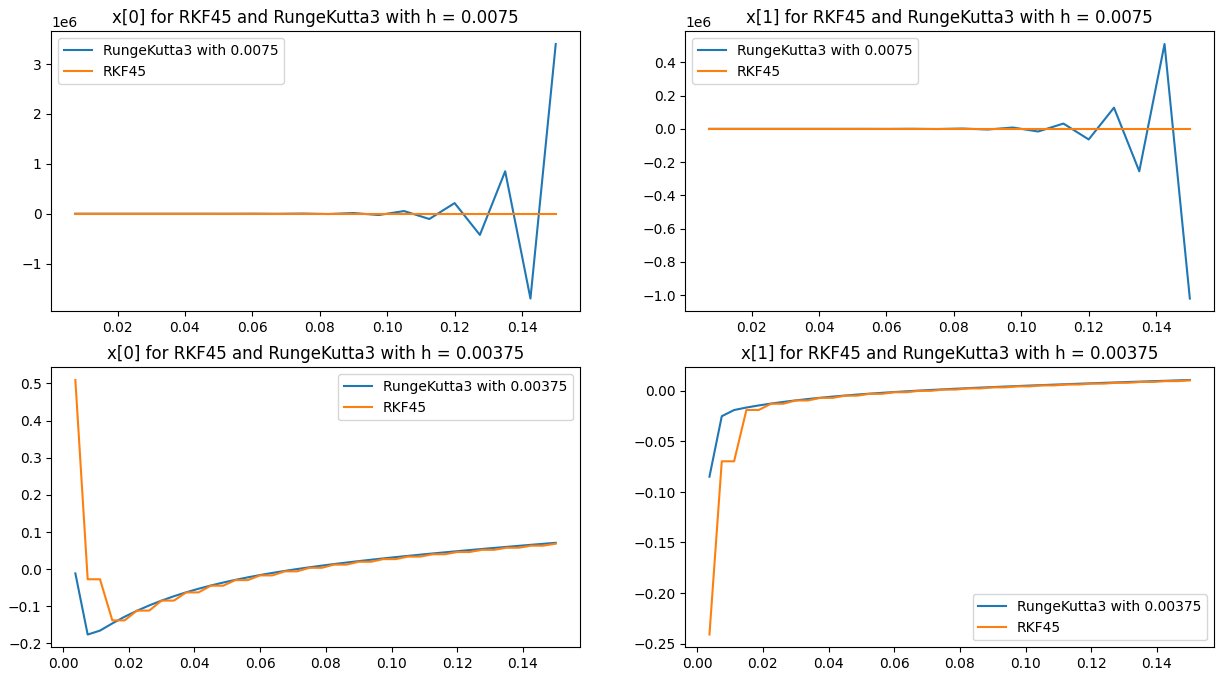
# Задание



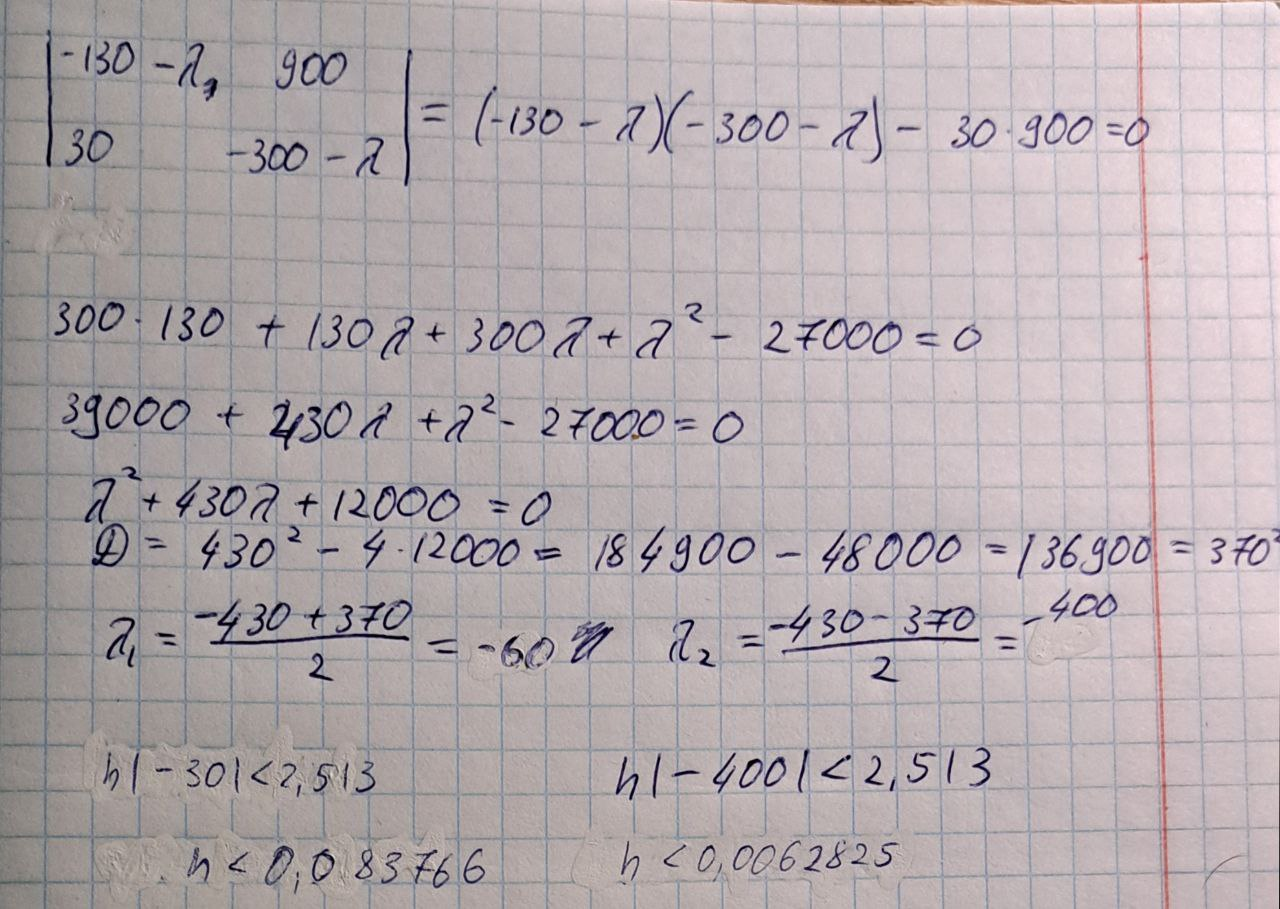
# Результаты







# Вывод



Расчитал собственные значения матрицы и нашел ограничения на шаг устойчивости. Вышло h < 0,0062825, хотя в самой программе у меня получались неверные результаты для метода Рунге-Кутты третьей степени с шагом равным 0,0075, но для шага 0,0037 результаты получились очень близкими.Их можно увидеть визуально на графике, там видно что хоть и не очень близко, но графики ведут себя одинаково для шага 0.0037, но для шага 0.0075 графики сильно отличаются.

# Код программы

## <DIR>/computational\_mathematics/lab\_3/main.cpp

#include <iostream>

#include <iomanip>

#include <fstream>

#include "RKF45.h"

#include "Function\_for\_laboratory.h"

#include "RungeKutta.h"

int main(int argc, char\*\* argv)

{

if (argc != 2)

{

std::cout << "Not enough argument\n";

return 1;

}

std::ofstream outFile;

outFile.exceptions(std::ofstream::badbit | std::ofstream::failbit);

try

{

outFile.open(argv[1]);

}

catch (const std::exception& ex)

{

std::cerr << ex.what() << "\n";

return 1;

}

double x\_first[2]{3.0, -1.0};

double x\_to\_csv[2]{3.0, -1.0};

double t(0);

double tOutFirst(0);

double relerr(0.0001);

double abserr(0.0001);

double work\_first[3+6\*2];

int flag = 1;

std::vector< double > x\_vect\_first{3.0, -1.0};

outFile << "t,rkf1,rkf2,rk1,rk2\n";

std::cout << "RKF45: | RungeKutta h = 0.0075 |\n";

for (double i = 0.0075; i <= 0.1501; i += 0.0075)

{

tOutFirst = i;

RKF45(mashkin::func< double\*, double\* >, 2, x\_first, t, tOutFirst, relerr, abserr, work\_first, flag);

std::cout << std::setw(6) << tOutFirst << " ";

outFile << tOutFirst << "," << x\_first[0] << "," << x\_first[1] << ",";

std::cout << std::setprecision(5) << std::setw(11) << x\_first[0] << " ";

std::cout << std::setw(12) << x\_first[1] << " flag = " << flag << " |" << std::setw(12);

std::cout << mashkin::rungeKutta3degree(tOutFirst, x\_vect\_first, 0.0075) << " |" << std::setw(10) << "\n";

outFile << x\_vect\_first[0] << "," << x\_vect\_first[1] << "\n";

}

double x\_second[2]{3.0, -1.0};

double tOutSecond(0);

relerr = 0.0001;

abserr = 0.0001;

double work\_second[3+6\*2];

flag = 1;

std::vector< double > x\_vect\_second{3.0, -1.0};

std::cout << "\nRKF45 with h = 0.00375: | RungeKutta h = 0.00375 |\n";

t = 0.0;

for (double i = 0.00375; i <= 0.1501; i += 0.00375)

{

tOutSecond = i;

RKF45(mashkin::func, 2, x\_second, t, tOutSecond, relerr, abserr, work\_second, flag);

std::cout << std::setw(8) << tOutSecond << " ";

outFile << tOutSecond << "," << x\_second[0] << "," << x\_second[1] << ",";

std::cout << std::setprecision(5) << std::setw(13) << x\_second[0] << " ";

std::cout << std::setw(13) << x\_second[1] << " flag = " << flag << "|" << std::setw(12);

std::cout << mashkin::rungeKutta3degree(tOutSecond, x\_vect\_second, 0.00375) << " |\n";

outFile << x\_vect\_second[0] << "," << x\_vect\_second[1] << "\n";

}

return 0;

}

## <DIR>/computational\_mathematics/lab\_3/Function\_for\_laboratory.h

#ifndef LAB3\_FUNCTION\_FOR\_LABORATORY\_H

#define LAB3\_FUNCTION\_FOR\_LABORATORY\_H

#include <cmath>

namespace mashkin

{

template< class X, class DX >

void func(double t, X x, DX dx)

{

dx[0] = -130 \* x[0] + 900 \* x[1] + std::exp(-10 \* t);

dx[1] = 30 \* x[0] - 300 \* x[1] + std::log(1 + 100 \* t \* t);

}

}

#endif

## <DIR>/computational\_mathematics/lab\_3/outputStructs.cpp

#include "outputStructs.h"

namespace mashkin

{

iofmtguard::iofmtguard(std::basic\_ios<char> &s):

s\_(s),

fill\_(s.fill()),

precision\_(s.precision()),

fmt\_(s.flags())

{

}

iofmtguard::~iofmtguard()

{

s\_.fill(fill\_);

s\_.precision(precision\_);

s\_.flags(fmt\_);

}

}

## <DIR>/computational\_mathematics/lab\_3/outputStructs.h

#ifndef LAB3\_OUTPUTSTRUCTS\_H

#define LAB3\_OUTPUTSTRUCTS\_H

#include <ios>

namespace mashkin

{

class iofmtguard

{

public:

iofmtguard(std::basic\_ios< char >& s);

~iofmtguard();

private:

std::basic\_ios< char >& s\_;

char fill\_;

std::streamsize precision\_;

std::basic\_ios< char >::fmtflags fmt\_;

};

}

#endif

## <DIR>/computational\_mathematics/lab\_3/RungeKutta.cpp

#include "RungeKutta.h"

#include <iostream>

#include <iomanip>

#include <cstddef>

#include <utility>

#include "Function\_for\_laboratory.h"

#include "outputStructs.h"

namespace mashkin

{

MyArr::MyArr(std::vector< double >& vect):

x(vect)

{

}

MyArr MyArr::operator=(MyArr&& rhs)

{

x = rhs.x;

return \*this;

}

MyArr MyArr::operator=(MyArr& rhs)

{

\*this = std::move(rhs);

return \*this;

}

double& MyArr::operator[](size\_t ind)

{

return this->x[ind];

}

MyArr MyArr::operator/(double num)

{

MyArr result = \*this;

for (size\_t i = 0; i < result.x.size(); i++)

{

result[i] /= num;

}

return result;

}

MyArr MyArr::operator+(MyArr&& rhs)

{

MyArr result = \*this;

for (size\_t i = 0; i < result.x.size(); i++)

{

result[i] += rhs[i];

}

return result;

}

MyArr MyArr::operator+(double num)

{

MyArr result = \*this;

for (size\_t i = 0; i < result.x.size(); i++)

{

result[i] += num;

}

return result;

}

MyArr MyArr::operator\*(double num)

{

MyArr result = \*this;

for (size\_t i = 0; i < result.x.size(); i++)

{

result[i] \*= num;

}

return result;

}

size\_t MyArr::size()

{

return x.size();

}

MyArr rungeKutta3degree(double t, std::vector< double >& x, double h)

{

std::vector< double > zero\_vect{0, 0};

MyArr k1(zero\_vect);

MyArr k2(zero\_vect);

MyArr k3(zero\_vect);

MyArr zn(x);

func< MyArr, MyArr& >(t, zn, k1);

k1 = k1 \* h;

func< MyArr&&, MyArr& >(t + h/2, zn + k1 / 2, k2);

k2 = k2 \* h;

func< MyArr&&, MyArr& >(t + (3 \* h) / 4, zn + (k2 \* 3) / 4, k3);

k3 = k3 \* h;

zn = zn + (k1 \* 2 + k2 \* 3 + k3 \* 4) / 9;

for (size\_t i = 0; i < x.size(); i++)

{

x[i] = zn[i];

}

return zn;

}

std::ostream& operator<<(std::ostream& out, MyArr&& dest)

{

std::ostream::sentry sentry(out);

if (!sentry)

{

return out;

}

iofmtguard fmtguard(out);

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

{

out << dest[i];

if (i + 1 != dest.size())

{

out << std::setw(13);

}

}

return out;

}

}

## <DIR>/computational\_mathematics/lab\_3/RungeKutta.h

#ifndef LAB3\_RUNGEKUTTA\_H

#define LAB3\_RUNGEKUTTA\_H

#include "Function\_for\_laboratory.h"

#include <cstddef>

#include <iostream>

#include <vector>

namespace mashkin

{

class MyArr

{

public:

MyArr() = default;

~MyArr() = default;

MyArr(std::vector< double >& vect);

MyArr(MyArr& rhs) = default;

MyArr(MyArr&& rhs);

MyArr operator/(double num);

MyArr operator=(MyArr& rhs);

MyArr operator=(MyArr&& rhs);

MyArr operator\*(double num);

MyArr operator+(MyArr&& rhs);

MyArr operator+(double num);

double& operator[](size\_t ind);

size\_t size();

private:

std::vector< double > x;

};

std::ostream& operator<<(std::ostream& out, MyArr&& dest);

MyArr rungeKutta3degree(double t, std::vector< double >& x, double h);

}

#endif