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.

Санкт-Петербург
2024

# Оглавление



# Задание

**ВАРИАНТ N 28**

Решить систему дифференциальных уравнений:

$$\frac{dx_1}{dt} = -130x_1 + 900x_2 + e^{-10t}; \qquad \frac{dx_2}{dt} = 30x_1 - 300x_2 + \ln(1 + 100t^2);$$

$$x_1(0) = 3, \qquad x_2(0) = -1; \qquad t \in [0,\ 0.15]$$

следующими способами с одним и тем же шагом печати $h_{print} = 0.0075$ :

I) по программе **RKF45** с EPS=0.0001;

II) методом Рунге-Кутты 3-й степени точности

$$z_{n+1} = z_n + (2k_1 + 3k_2 + 4k_3)/9; \quad k_1 = hf(t_n, z_n);$$

$$k_2 = hf(t_n + h/2,\ z_n + k_1/2); \qquad k_3 = hf(t_n + 3h/4,\ z_n + 3k_2/4);$$

с двумя постоянными шагами интегрирования:

а) $h_{int} = 0.0075$

б) любой другой, позволяющий получить качественно верное решение.

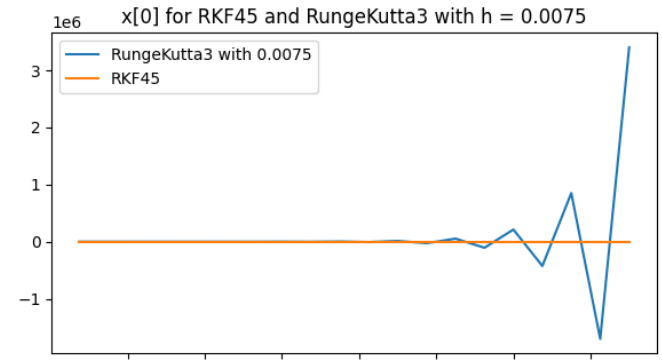
Сравнить результаты.

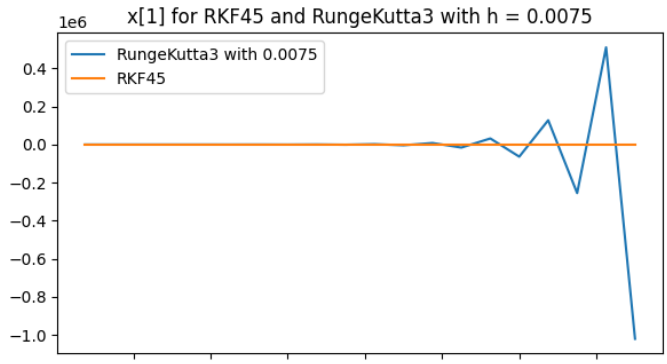
Санкт-Петербург
2024

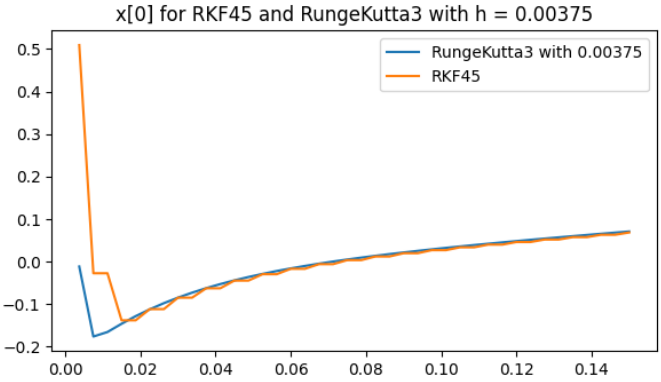
# Результаты

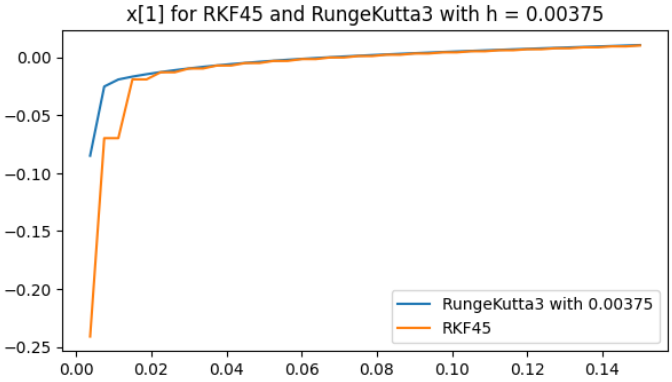
| RKF45: | | | | RungeKutta h = 0.0075 | |
|---|---|---|---|---|---|
| 0.0075 | -0.027447 | -0.069666 | flag = 2 | -6.6743 | 1.9243 |
| 0.015 | -0.13819 | -0.01886 | flag = 2 | 12.824 | -3.9075 |
| 0.0225 | -0.13819 | -0.01886 | flag = 4 | -26.053 | 7.7698 |
| 0.03 | -0.084897 | -0.0094653 | flag = 2 | 51.797 | -15.574 |
| 0.0375 | -0.084897 | -0.0094653 | flag = 4 | -103.83 | 31.122 |
| 0.045 | -0.044712 | -0.0047339 | flag = 2 | 207.48 | -62.263 |
| 0.0525 | -0.044712 | -0.0047339 | flag = 4 | -415.08 | 124.51 |
| 0.06 | -0.017002 | -0.0013309 | flag = 2 | 830.09 | -249.03 |
| 0.0675 | -0.017002 | -0.0013309 | flag = 4 | -1660.2 | 498.07 |
| 0.075 | 0.0034478 | 0.0013048 | flag = 2 | 3320.4 | -996.13 |
| 0.0825 | 0.0034478 | 0.0013048 | flag = 4 | -6640.9 | 1992.3 |
| 0.09 | 0.019719 | 0.0034965 | flag = 2 | 13282 | -3984.5 |
| 0.0975 | 0.019719 | 0.0034965 | flag = 4 | -26563 | 7969 |
| 0.105 | 0.033579 | 0.0054267 | flag = 2 | 53127 | -15938 |
| 0.1125 | 0.033579 | 0.0054267 | flag = 4 | -1.0625e+05 | 31876 |
| 0.12 | 0.046014 | 0.0071952 | flag = 2 | 2.1251e+05 | -63752 |
| 0.1275 | 0.046014 | 0.0071952 | flag = 4 | -4.2502e+05 | 1.275e+05 |
| 0.135 | 0.057555 | 0.0088551 | flag = 2 | 8.5003e+05 | -2.5501e+05 |
| 0.1425 | 0.057555 | 0.0088551 | flag = 4 | -1.7001e+06 | 5.1002e+05 |
| 0.15 | 0.068478 | 0.010434 | flag = 2 | 3.4001e+06 | -1.02e+06 |

| RKF45 with h = 0.00375: | | | | RungeKutta h = 0.00375 | |
|---|---|---|---|---|---|
| 0.00375 | 0.50933 | -0.24111 | flag = 2 | -0.011598 | -0.084862 |
| 0.0075 | -0.027444 | -0.069667 | flag = 2 | -0.17635 | -0.025036 |
| 0.01125 | -0.027444 | -0.069667 | flag = 4 | -0.1657 | -0.018942 |
| 0.015 | -0.13819 | -0.01886 | flag = 2 | -0.14633 | -0.01645 |
| 0.01875 | -0.13819 | -0.01886 | flag = 4 | -0.12839 | -0.014396 |
| 0.0225 | -0.11181 | -0.012715 | flag = 2 | -0.11229 | -0.012557 |
| 0.02625 | -0.11181 | -0.012715 | flag = 4 | -0.097834 | -0.010899 |
| 0.03 | -0.084897 | -0.0094653 | flag = 2 | -0.08484 | -0.0093963 |
| 0.03375 | -0.084897 | -0.0094653 | flag = 4 | -0.073128 | -0.0080317 |
| 0.0375 | -0.062808 | -0.0068847 | flag = 2 | -0.062545 | -0.0067877 |
| 0.04125 | -0.062808 | -0.0068847 | flag = 4 | -0.052952 | -0.0056496 |
| 0.045 | -0.044712 | -0.0047337 | flag = 2 | -0.044227 | -0.004604 |
| 0.04875 | -0.044712 | -0.0047337 | flag = 4 | -0.036264 | -0.0036396 |
| 0.0525 | -0.029685 | -0.0029084 | flag = 2 | -0.028968 | -0.0027461 |
| 0.05625 | -0.029685 | -0.0029084 | flag = 4 | -0.022255 | -0.0019146 |
| 0.06 | -0.017002 | -0.0013308 | flag = 2 | -0.016052 | -0.0011374 |
| 0.06375 | -0.017002 | -0.0013308 | flag = 4 | -0.010295 | -0.00040762 |
| 0.0675 | -0.006101 | 5.8515e-05 | flag = 2 | -0.0049277 | 0.00028067 |
| 0.07125 | -0.006101 | 5.8515e-05 | flag = 4 | 9.911e-05 | 0.00093264 |
| 0.075 | 0.0034476 | 0.0013048 | flag = 2 | 0.0048285 | 0.0015528 |
| 0.07875 | 0.0034476 | 0.0013048 | flag = 4 | 0.009298 | 0.0021451 |
| 0.0825 | 0.011972 | 0.0024423 | flag = 2 | 0.01354 | 0.002713 |
| 0.08625 | 0.011972 | 0.0024423 | flag = 4 | 0.017584 | 0.0032593 |
| 0.09 | 0.019719 | 0.0034965 | flag = 2 | 0.021454 | 0.0037866 |
| 0.09375 | 0.019719 | 0.0034965 | flag = 4 | 0.025171 | 0.0042972 |
| 0.0975 | 0.026875 | 0.0044866 | flag = 2 | 0.028753 | 0.004793 |
| 0.10125 | 0.026875 | 0.0044866 | flag = 4 | 0.032218 | 0.0052755 |
| 0.105 | 0.033579 | 0.0054267 | flag = 2 | 0.035579 | 0.0057462 |
| 0.10875 | 0.033579 | 0.0054267 | flag = 4 | 0.038847 | 0.0062063 |
| 0.1125 | 0.039934 | 0.006327 | flag = 2 | 0.042033 | 0.0066569 |
| 0.11625 | 0.039934 | 0.006327 | flag = 4 | 0.045146 | 0.0070988 |
| 0.12 | 0.046014 | 0.0071952 | flag = 2 | 0.048194 | 0.0075328 |
| 0.12375 | 0.046014 | 0.0071952 | flag = 4 | 0.051182 | 0.0079596 |
| 0.1275 | 0.051875 | 0.0080366 | flag = 2 | 0.054116 | 0.0083798 |
| 0.13125 | 0.051875 | 0.0080366 | flag = 4 | 0.057001 | 0.0087937 |
| 0.135 | 0.057555 | 0.0088551 | flag = 2 | 0.059842 | 0.0092018 |
| 0.13875 | 0.057555 | 0.0088551 | flag = 4 | 0.062641 | 0.0096045 |
| 0.1425 | 0.063083 | 0.0096535 | flag = 2 | 0.065401 | 0.010002 |
| 0.14625 | 0.063083 | 0.0096535 | flag = 4 | 0.068126 | 0.010395 |
| 0.15 | 0.068478 | 0.010434 | flag = 2 | 0.070816 | 0.010783 |



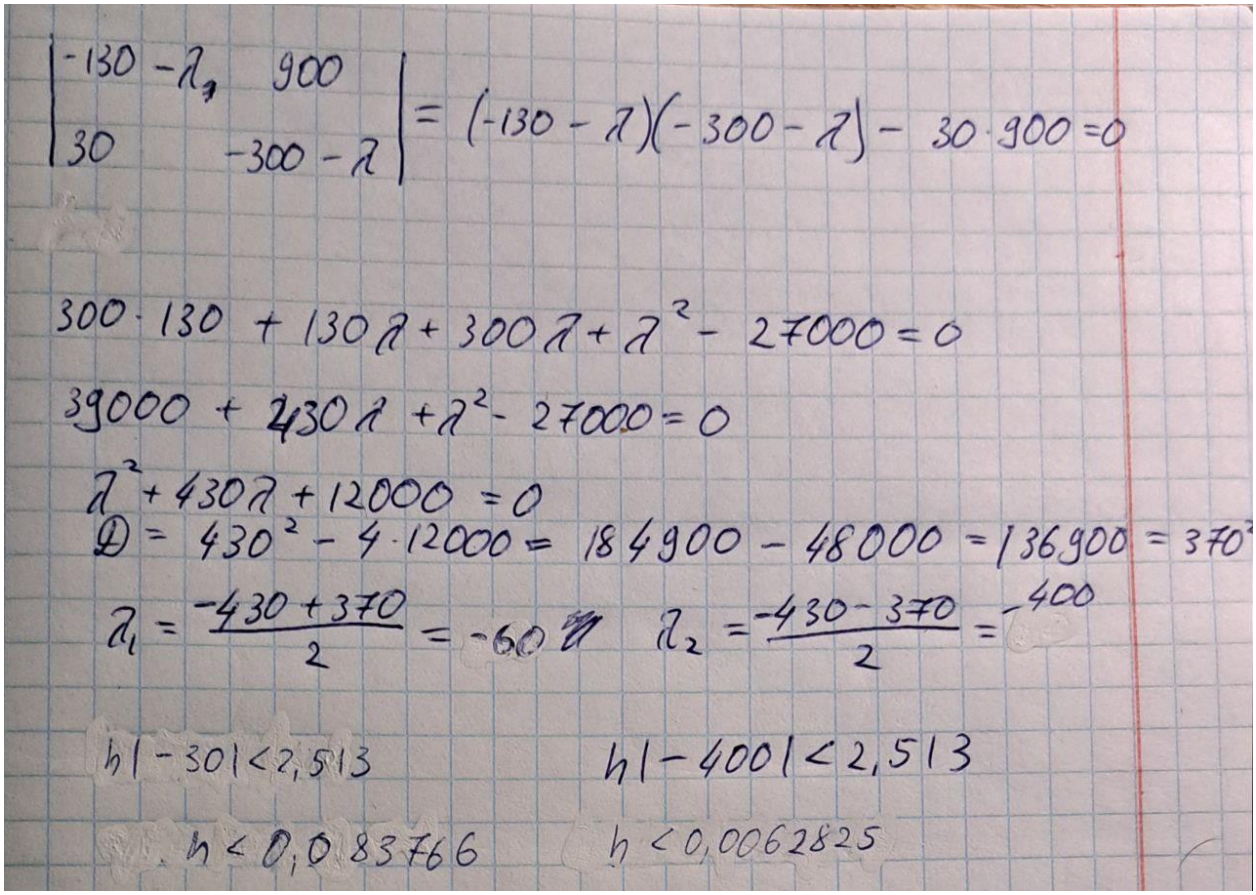




Санкт-Петербург
2024

# 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.

Санкт-Петербург
2024

# 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
```