МИНИСТЕРСТВО НАУКИ
И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

НГТУ НЭТИ

Кафедра прикладной математики

Практическое задание № 5

по дисциплине « ЧМДСООДУ»

Системы ОДУ

Группа	ПМ-81
Студенты	ЕФРЕМОВ АРТУР
	ТЕТЮШКИН ВСЕВОЛОД
Преподаватели	ВАГИН ДЕНИС ВЛАДМИРОВИЧ
	СИВЕНКОВА АНАСТАСИЯ ПАВЛОВНА
Дата	14.12.2020

Новосибирск

1. Задание:

Параметры системы:

$$p^{атм} = 1e+5$$

$$q^0 = 0.001$$

$$l = 1$$

$$d = 0.01$$

$$\rho = 1000$$

$$C_{snd} = 1260$$

$$\zeta = 1 - \cos\left(\frac{\pi}{4}\right)$$

$$C = \frac{lS}{\rho C_{snd}^2}$$

$$S = \frac{\pi d^2}{4}$$

$$q^н(t) = \begin{cases} q^0 t, t \le 1 \\ q^0, t > 1 \end{cases}$$

Интервал интегрирования - t∈ [0,20].

1) С помощью двух шагов h=1e-4 и h=1e-5 проинтегрировать систему

$$\begin{cases} \dfrac{dp_1(t)}{dt} = \dfrac{q^н(t) - q_2(t)}{C} \\ \dfrac{dq_2(t)}{dt} = \sqrt{\dfrac{\zeta\left|p_1(t) - p^{атм}\right|}{2\rho}}\left(S\sqrt{\dfrac{2\left|p_1(t) - p^{атм}\right|}{\rho\zeta}}sign\left(p_1(t) - p^{атм}\right) - q_2(t)\right) \end{cases}$$

с начальными условиями

$$p_1(0) = p^{атм}$$

$$q_2(0) = 0$$

2) Использовать последовательное интегрирование для элементов системы (для тех же шагов интегрирования).

2. Текст программы:

// Файл «DifLab.h»

```
#pragma once
#include <vector>
#include <cmath>

using namespace std;

const double PI = 3.14159265358979323846,
eps = pow(10, -14), patm = 1E+5,
q0 = 0.001, l = 1.0, d = 0.01, ro = 1000, Csnd = 1260,
z = 1 - cos(PI / 4), t0 = 0.0, tn = 20.0;
const vector<double> H = { 1e-4, 1e-5 };

const double S = PI * d * d / 4.0;
const double C = l * S / (ro * Csnd * Csnd);

inline int sign(const double d)
{
   return -1 * (d < 0) + 1 * (d > 0);
}

inline double qh(const double t)
{
   return (t < 1.0) * q0 * t + q0 * (t > 1.0);
}

inline double p(const double t, const vector<double>& resn)
{
   return (qh(t) - resn[1]) / C;
}

double q(const double t, const vector<double>& resn)
{
   const double dif = resn[0] - patm;
   const double root_1 = sqrt(z   * abs(dif) / (2.0 * ro));
   const double root_2 = sqrt(2.0 * abs(dif) / (z   * ro));

   return root_1 * (S * root_2 * sign(dif) - resn[1]);
}

inline vector<double> f(const double t, const vector<double>& vec)
{
   return { p(t, vec), q(t, vec) };
}

vector<double> fill_grid(const double t0, const  double tn, const double h)
{
   int N = (tn - t0) / h + 1;
   vector<double> res(N);
```

```
    for (int n = 0; n < N; n++)
        res[n] = t0 + n * h;

    return res;
}
```

// Файл «adams.h»

```
#pragma once
#include <vector>
#include "runge.h"

using namespace std;

vector<double> adams3_exp_n(const int n,
    const vector<double>& T,
    const double h,
    const vector<vector<double>>& Y,
    vector<double> func(double, const vector<double>&))
{
    return Y[n - 1] + h *
        (23.0 * func(T[n - 1], Y[n - 1]) -
         16.0 * func(T[n - 2], Y[n - 2]) +
          5.0 * func(T[n - 3], Y[n - 3])) * (1 / 12.0);
}

vector<vector<double>> adams3_exp(const vector<double>& T,
    const double h,
    vector<double> func(const double, const vector<double>&))
{
    int N = T.size();
    vector<vector<double>> res (N);

    for (int i = 0; i < N; i++)
        res[i].resize(2);

    res[0][0] = patm;
    res[0][1] = 0.0;

    res[1] = runge_n(1, T, h, res, func);
    res[2] = runge_n(2, T, h, res, func);

    for (int n = 3; n < N; n++)
        res[n] = adams3_exp_n(n, T, h, res, func);

    return res;
}
```

// Файл «runge.h»

```
#pragma once
#include <vector>
#include "Vector.h"
#include "DifLab.h"

using namespace std;

inline vector<double> kn1(const double tn, const vector<double>& yn,
    vector<double> func(const double, const vector<double>&))
{
    return func(tn, yn);
}
```

```
inline vector<double> kn2(const double tn, const vector<double>& yn, const double h,
   vector<double> func(const double, const vector<double>&))
{
   return func(tn + h / 2, yn + h / 2 * kn1(tn, yn, func));
}

inline vector<double> kn3(const double tn, const vector<double>& yn, const double h,
   vector<double> func(const double, const vector<double>&))
{
   return func(tn + h / 2, yn + h / 2 * kn2(tn, yn, h, func));
}

inline vector<double> kn4(const double tn, const vector<double>& yn, const double h,
   vector<double> func(const double, const vector<double>&))
{
   return func(tn + h, yn + h * kn3(tn, yn, h, func));
}

inline vector<double> kn(const double tn, const vector<double>& yn, const double h,
   vector<double> func(const double, const vector<double>&))
{
   return 1 / 6.0 * (kn1(tn, yn, func) +
                2 * kn2(tn, yn, h, func) +
                2 * kn3(tn, yn, h, func) +
                    kn4(tn, yn, h, func));
}

vector<double> runge_n(const int n,
               const vector<double>& T,
               const double h,
               const vector<vector<double>>& Y,
               vector<double> func(const double, const vector<double>&))
{
   return Y[n - 1] + h * kn(T[n - 1], Y[n - 1], h, func);
}
```

// Файл «Vector.h»

```
#pragma once
#include <vector>
#include <fstream>
using namespace std;

// Умножение вектора на число
vector<double> operator * (double val, const vector<double>& vec)
{
   vector<double> res(vec.size());

   for (size_t i = 0; i < vec.size(); ++i)
      res[i] = val * vec[i];
   return res;
}

vector<double> operator * (const vector<double>& vec, double val)
{
   return val * vec;
}

// Сложение вектора с числом
vector<double> operator + (double val, const vector<double>& vec)
{
   vector<double> res(vec.size());
```

```
    for (size_t i = 0; i < vec.size(); ++i)
        res[i] = val + vec[i];
    return res;
}

vector<double> operator + (const vector<double>& vec, double val)
{
    return val + vec;
}

// Сложение векторов
vector<double> operator + (const vector<double>& vec1, const vector<double>& vec2)
{
    vector<double> res(vec1.size());

    for (size_t i = 0; i < vec1.size(); ++i)
        res[i] = vec1[i] + vec2[i];
    return res;
}

// Вычитание векторов
vector<double> operator - (const vector<double>& vec1, const vector<double>& vec2)
{
    vector<double> res(vec1.size());

    for (size_t i = 0; i < vec1.size(); ++i)
        res[i] = vec1[i] - vec2[i];
    return res;
}
```

// Файл «main.cpp»

```
#include <fstream>
#include <iostream>
#include <string>
#include "Vector.h"
#include "adams.h"
#include "runge.h"
#include "DifLab.h"

using namespace std;

void report(const string file_name,
            vector<vector<double>> method(const vector<double>&, const double,
            vector<double> (const double, const vector<double>&)))
{
    ofstream fout;
    double kp = 0, kq = 0;

    fout.open("info/info_step_" + file_name + ".txt");
    for (int i = 0; i < H.size(); i++)
        fout << log10(H[i]) << " ";
    fout.close();

    for (int i = 0; i < H.size(); i++)
    {
        string filename_e = file_name + "_1e" + to_string((int)log10(H[i]));
        fout.open(path + filename_e + ".txt");
        vector<double> T = fill_grid(t0, tn, H[i]);

        vector<vector<double>> Y_num = method(T, H[i], f);

        fout.close();

        fout.open("values/" + filename_e + "_vals.txt");
```

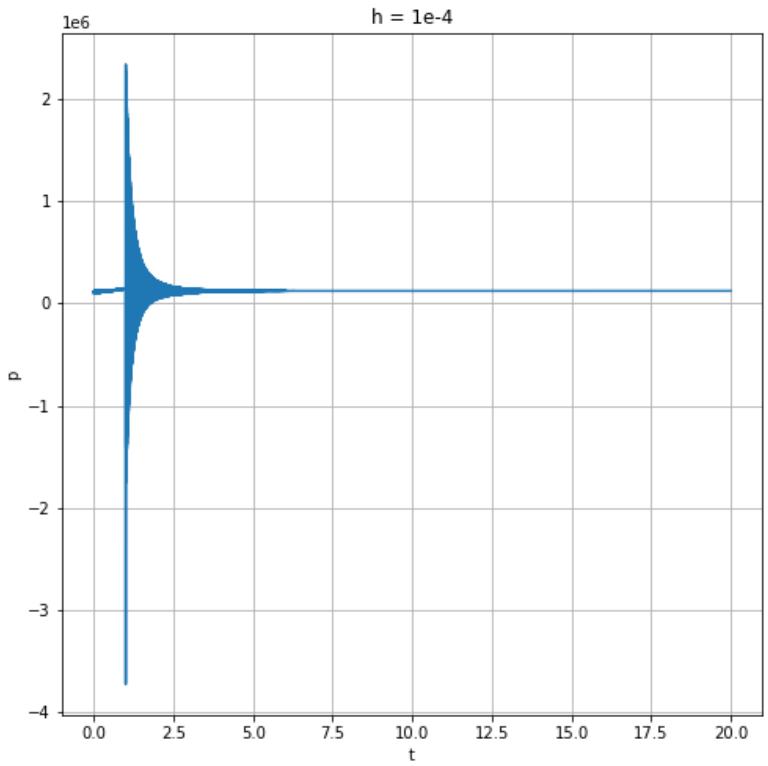
```
      int size = T.size();
      for (int j = 0; j < size; j++)
      {
         fout << T[j] << ",";
         fout << Y_num[j][0] << ",";
         fout << Y_num[j][1] << endl;
      }

      fout.close();
   }
}

int main()
{
   report("adams3", adams3_exp);
}
```

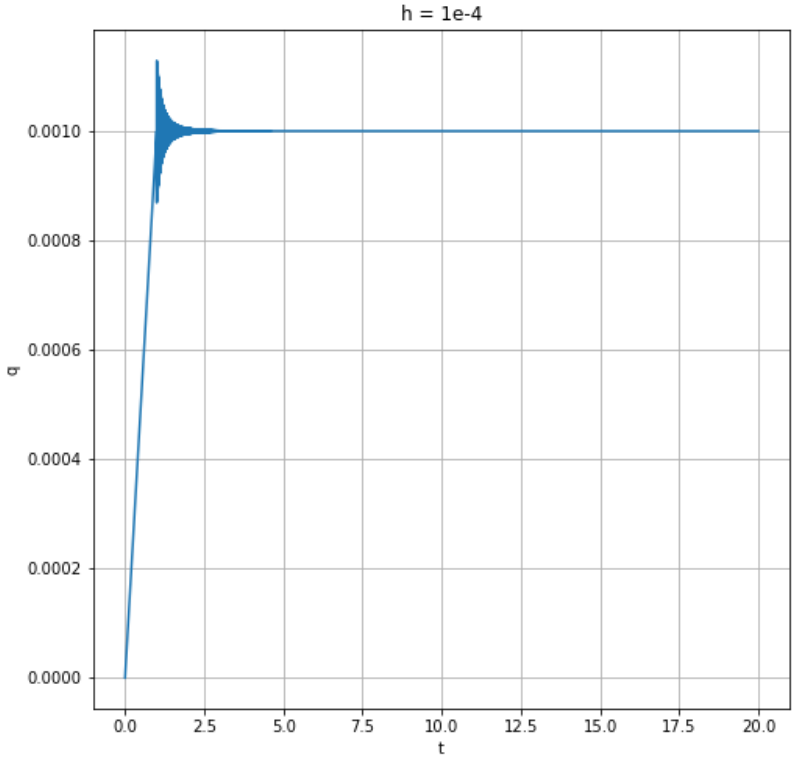
Программа на языке python для формирования графиков из файлов:

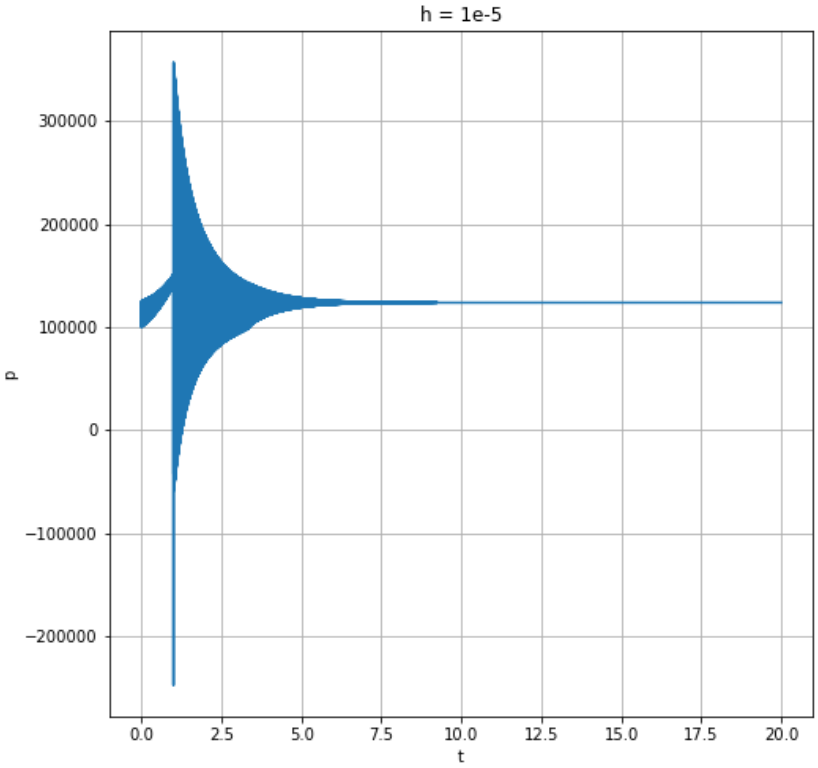
```
import numpy as np
import matplotlib.pyplot as plt

def save_graph(x, y, x_name, y_name, path, test, filename):
    fig, ax = plt.subplots(figsize=(15,15))
    ax.plot(x, y)

    ax.set(xlabel=x_name, ylabel=y_name,
        title='h = ' + test )
    ax.grid()

    fig.savefig(path + filename + y_name + ".png")

def build_graph(method):
    infofile = open("info/info_step_" + method + ".txt", "r")
    info_step = infofile.readline().split(' ')
    info_step.remove("")

    print(info_step)

    for info_item in info_step:
        test = "1e" + info_item

        filename = method + "_" + test + "_vals"
        openfile = open("values/" + filename  + ".txt", 'r')
        lines = openfile.readlines()
        coords = np.empty((len(lines),3))

        for i in range(len(lines)):
            l = lines[i].split(',')
            coords[i] = l

        openfile.close()

        t = coords[:, 0]
```

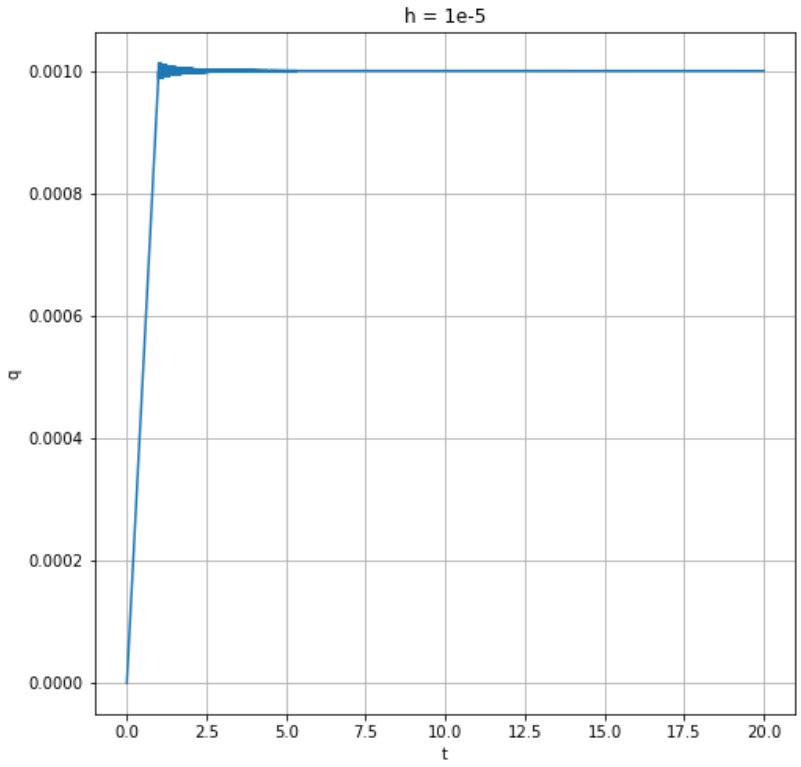
```
        p = coords[:, 1]
        q = coords[:, 2]

        save_graph(t, p, "t", "p", "graphs/", test, filename)
        save_graph(t, q, "t", "q", "graphs/", test, filename)
build_graph("adams3")
```

3. Графики:


h = 1e-4
1e6
2
1
0
−1
−2
−3
−4
p
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
t


h = 1e-4
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
q
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
t


h = 1e-5
300000
200000
100000
0
−100000
−200000
p
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
t


h = 1e-5
0.0010
0.0008
0.0006
0.0004
0.0002
0.0000
q
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
t

4. Значения функций при t = 10:

	p(t)	q(t)
1e-4	123746	0.001
1e-5	123780	0.000999999