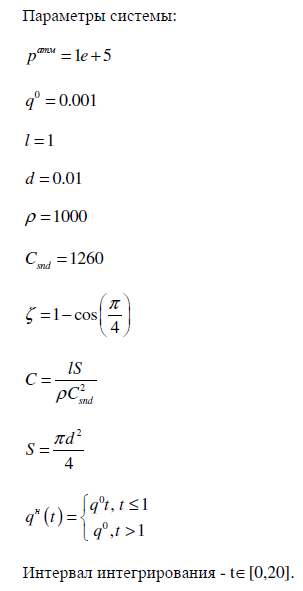
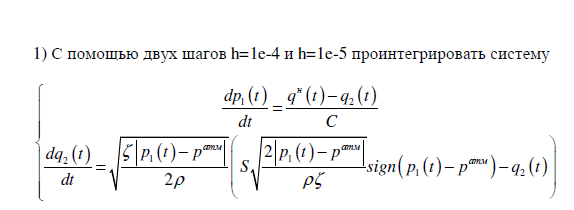
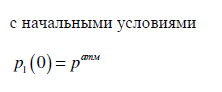
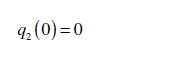
|  |  |  |  |
| --- | --- | --- | --- |
|  | | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования «Новосибирский государственный технический университет» | | | |
|  | | | |
| Кафедра прикладной математики | | | |
|  | | | |
| Практическое задание № 5 | | | |
| 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:**









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

1. **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]

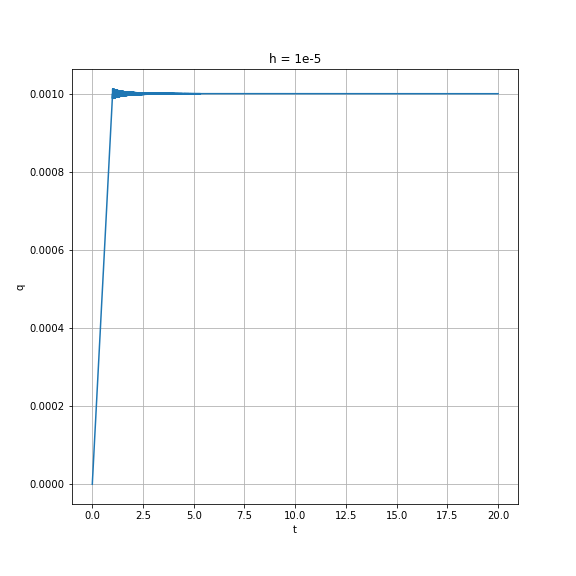
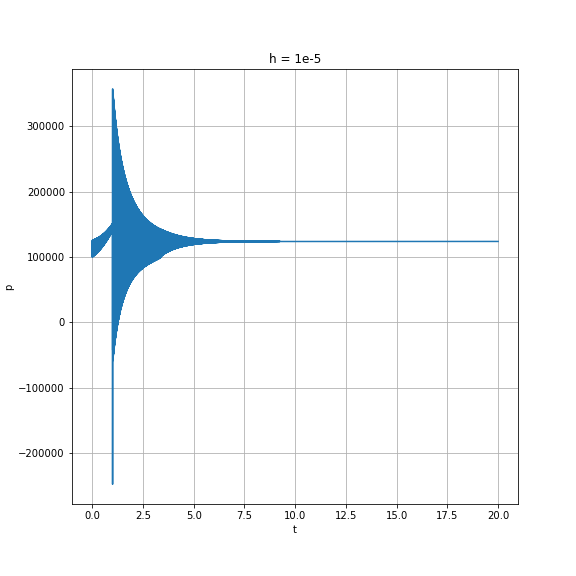
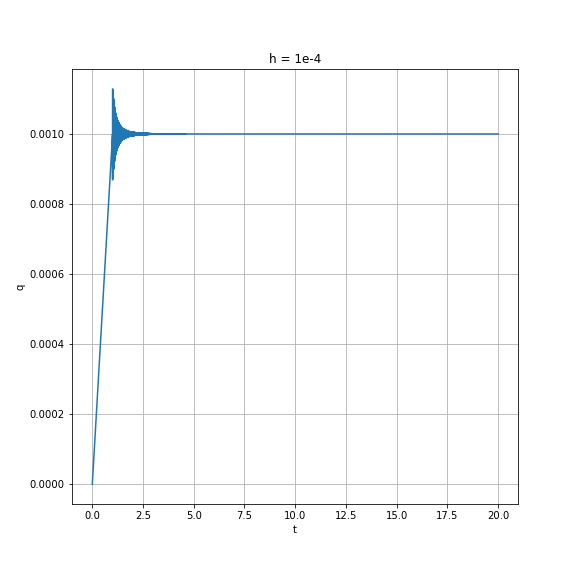
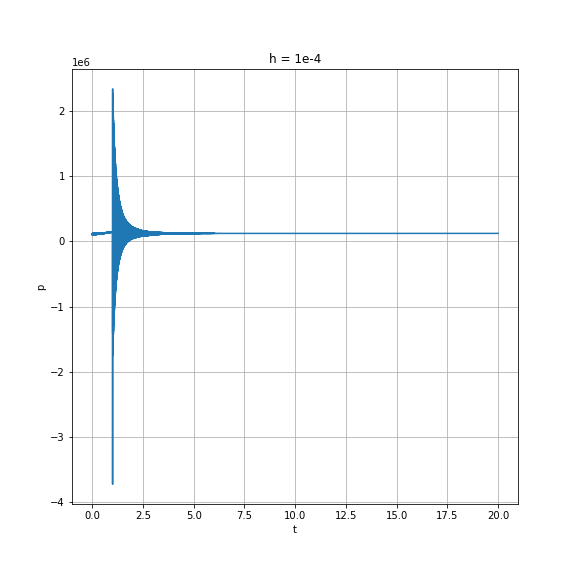
q = coords[:, 2]

save\_graph(t, p, "t", "p", "graphs/", test, filename)

save\_graph(t, q, "t", "q", "graphs/", test, filename)

build\_graph("adams3")

1. **Графики:**



1. **Значения функций при t = 10:**

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