Санкт-Петербургский государственный университет

Прикладная математика и информатика

Отчет по лабораторной работе №6 по вычислительному практикуму

**Численное решение Задачи Коши для обыкновенного дифференциального уравнения первого порядка**

Выполнил:

Чернов Павел Олегович

группа 221

Санкт-Петербург

2022

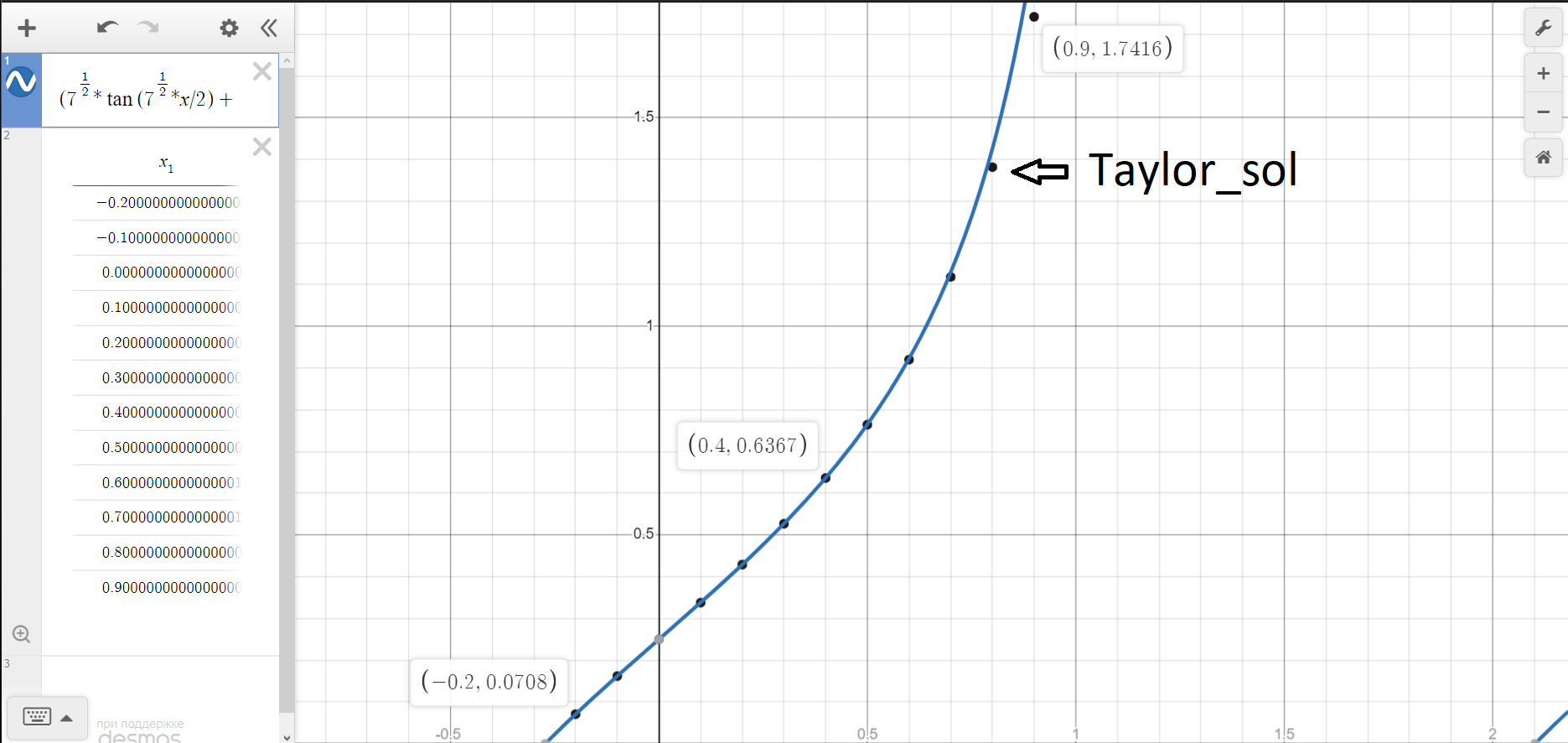
1. **Введение**

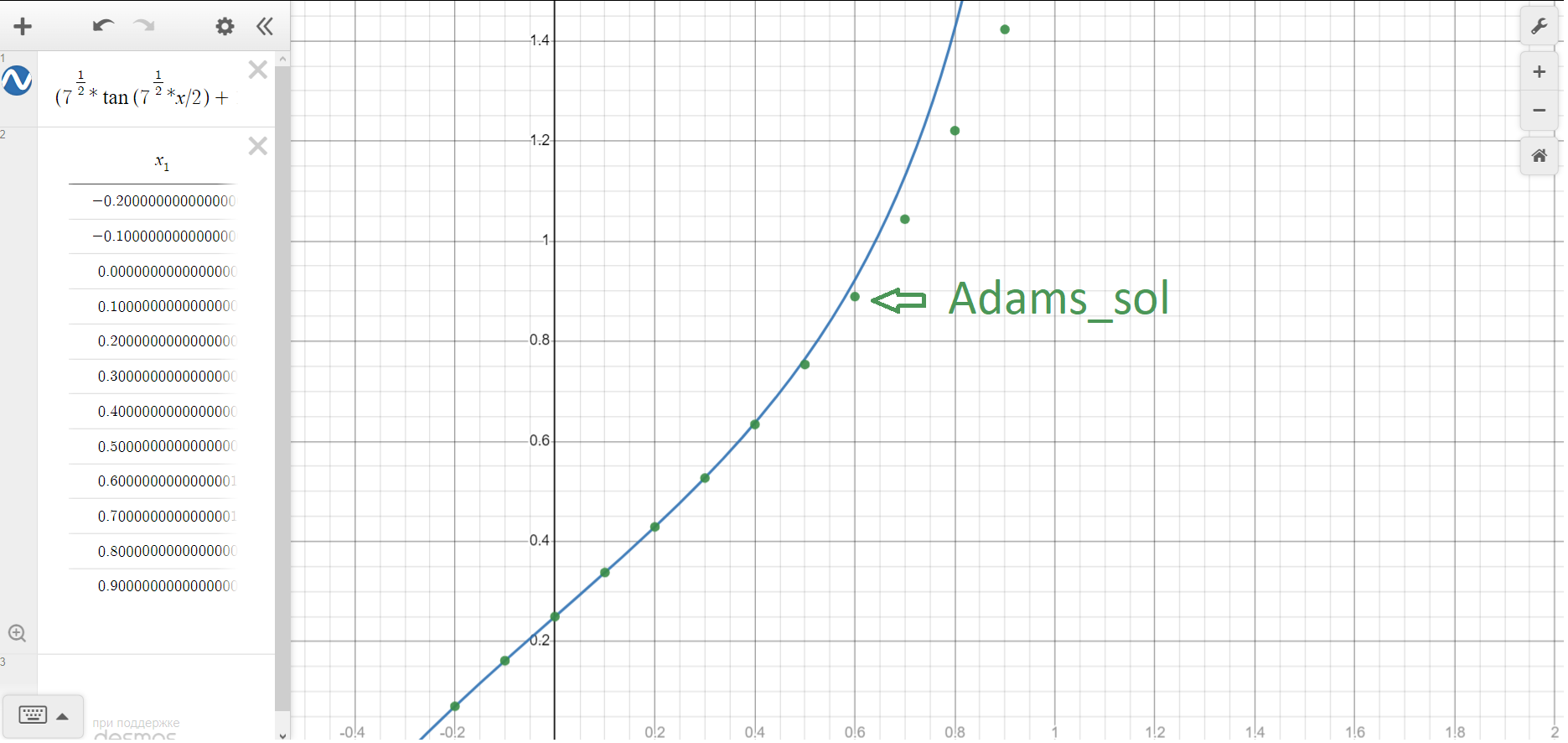
**Численное решение Задачи Коши для обыкновенного дифференциального уравнения первого порядка**

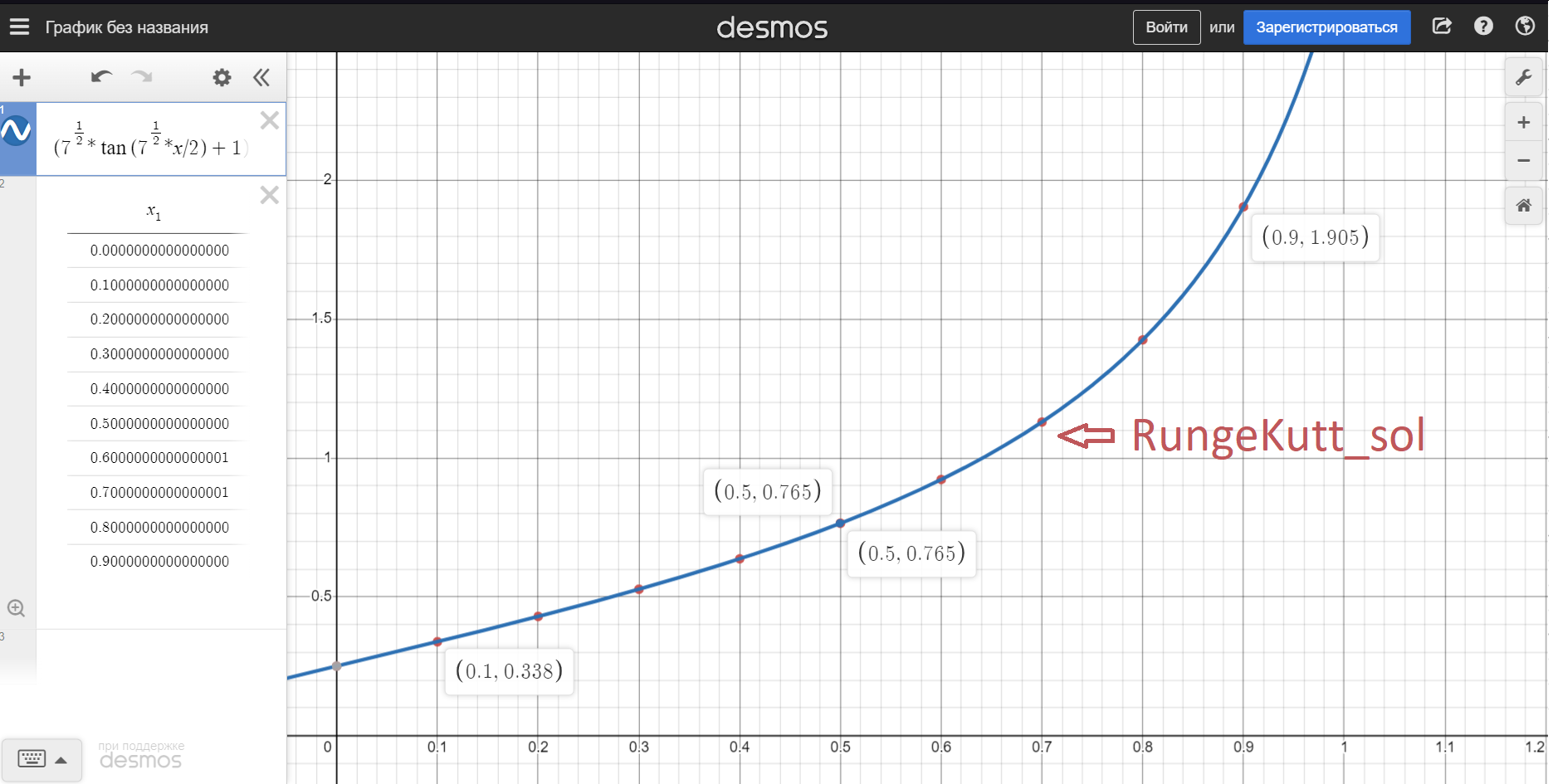
Найти решение Задачи Коши для уравнения y ́(x)=f(x,y) с начальным условием y(x0)=y0следующими методами:

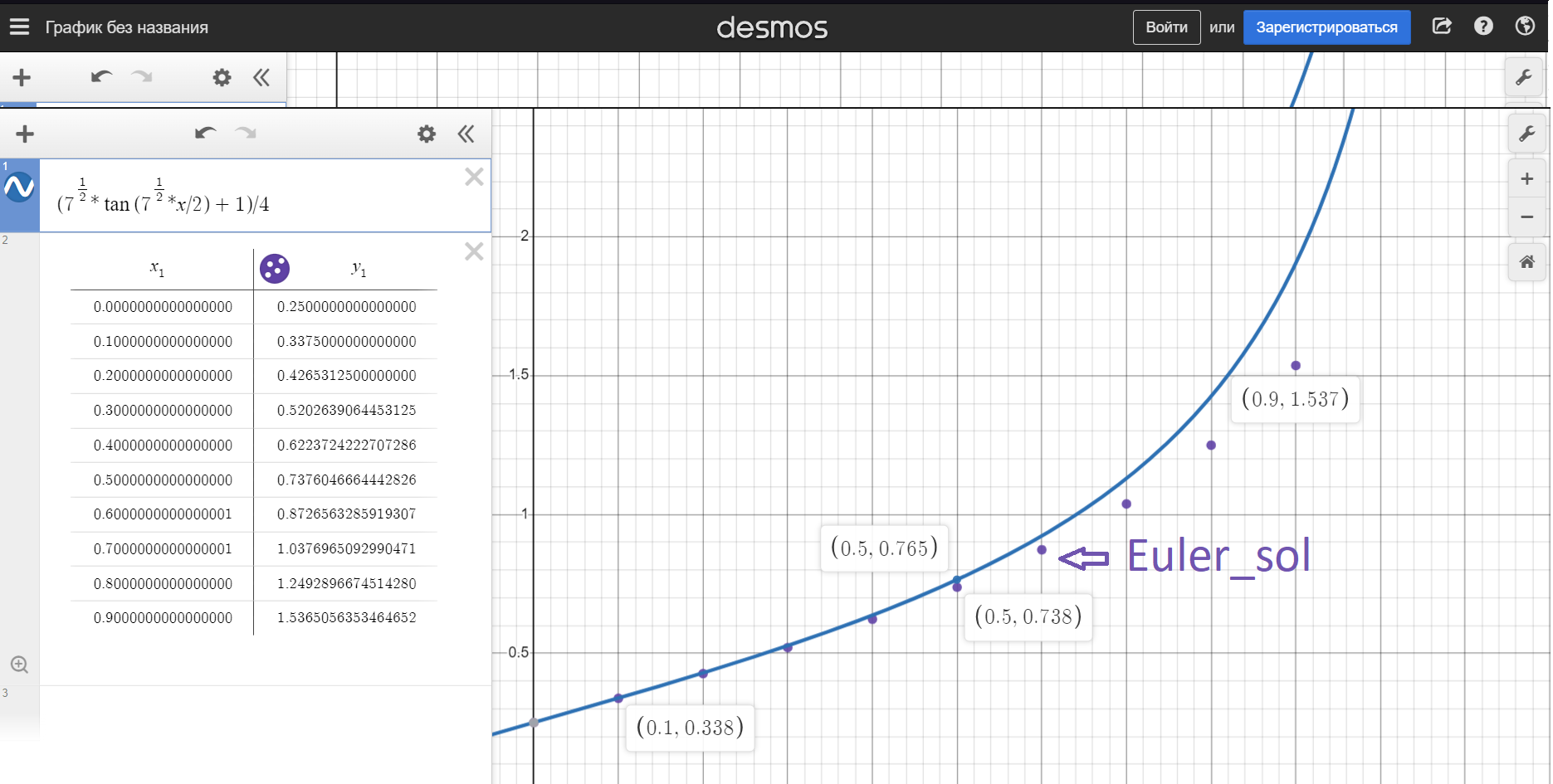
1. Найти точное решение тестовой задачи. Вывести на печать таблицу значений точного решения в равноотстоящих с шагом h точках xk = x0+k∙h, где k=-2, -1, 0,..., N; здесь N и h−параметры задачи.
2. Методом разложения в ряд Тейлора (можно ограничиться пятью-шестью ненулевыми слагаемыми) найти и вывести на печать значения приближённого решения в точках xk=x0+k∙h, где k=-2, -1, 0,..., N.
3. Во всех точках xk вывести на печать значения абсолютной погрешности метода разложения в ряд Тейлора.
4. Используя начало таблицы, построенное в п. 2) (первые пять найденных Тейлором значений), вывести на печать значения приближённого решения, полученного экстраполяционным методом Адамса 4-го порядка в точках xk=x0+k∙h, где k=3,4,...,N.
5. Методом Рунге-Кутта 4-го порядка найти и вывести на печать значения приближённого решения в точках xk=x0+k∙h, где k=1,2,...,N.
6. Найти и вывести на печать значения приближенных решений исходной задачи, полученных методом Эйлера, методом ЭйлераI и методом ЭйлераII в точках xk=x0+k∙h, где k=1,2,...,N.
7. Для всех методов определить абсолютную погрешность для последнего значения yN≈y(xN).
8. **Ход работы**
9. Представленная ниже программа позволяет аппроксимировано вычислять решение задачи Коши дифференциального уравнения первого порядка.
10. Для приближенного вычисления решений используются метод Тейлора, метод Адамса 4-ого порядка, метод Рунге-Кутта 4-ого порядка, а также методы Эйлера, ЭйлераI и ЭйлераII.
11. Исходным параметром задачи является уравнение . В программе можно задать а) h – длину шага аргумента в таблице, б) N – количество шагов или табличных значений. Ввод осуществляется при помощи клавиатуры.
12. Программа написана на языке С++, в расчетах используются переменные типа double.
13. **Приложения**

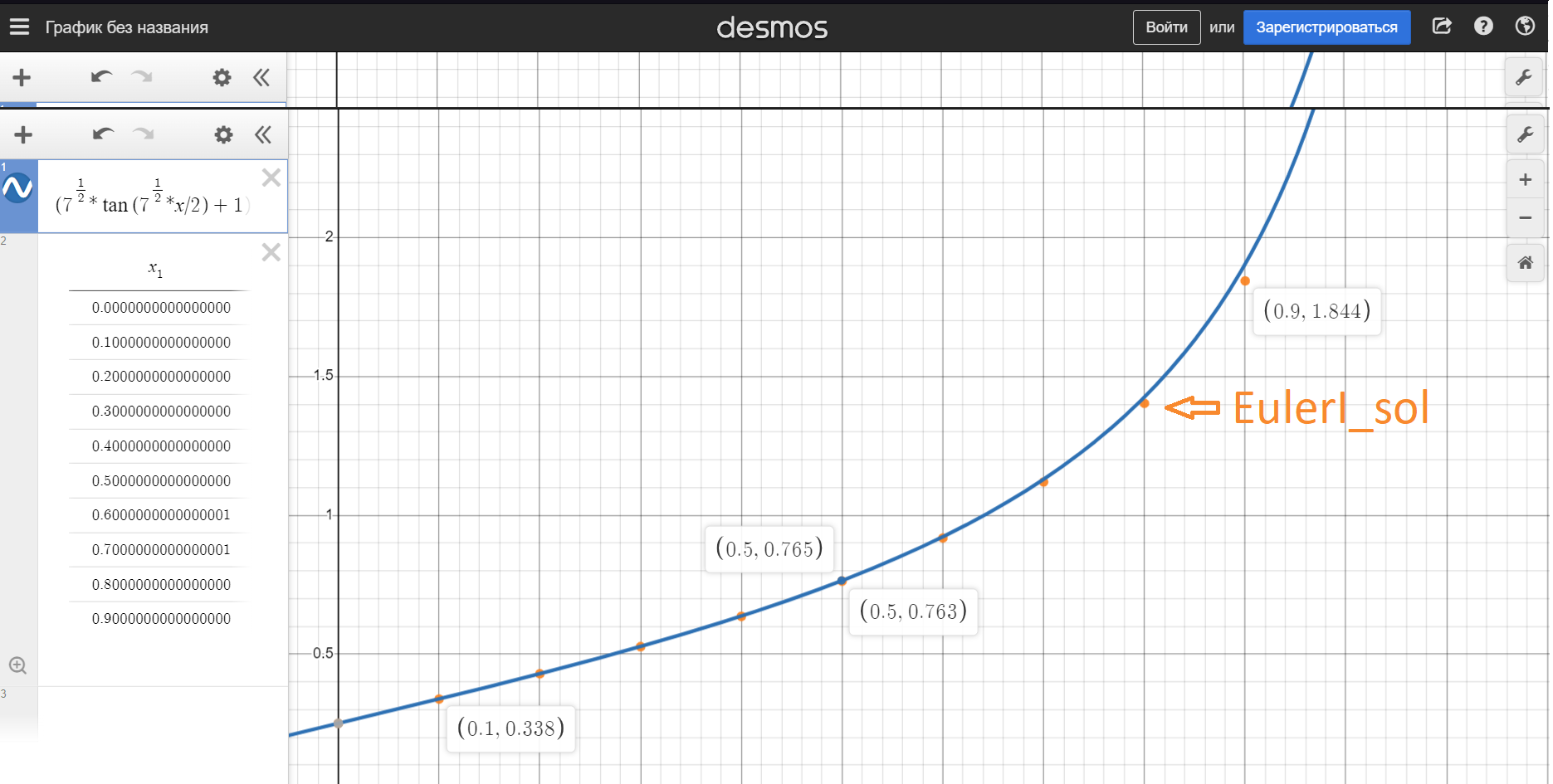
А) Графики

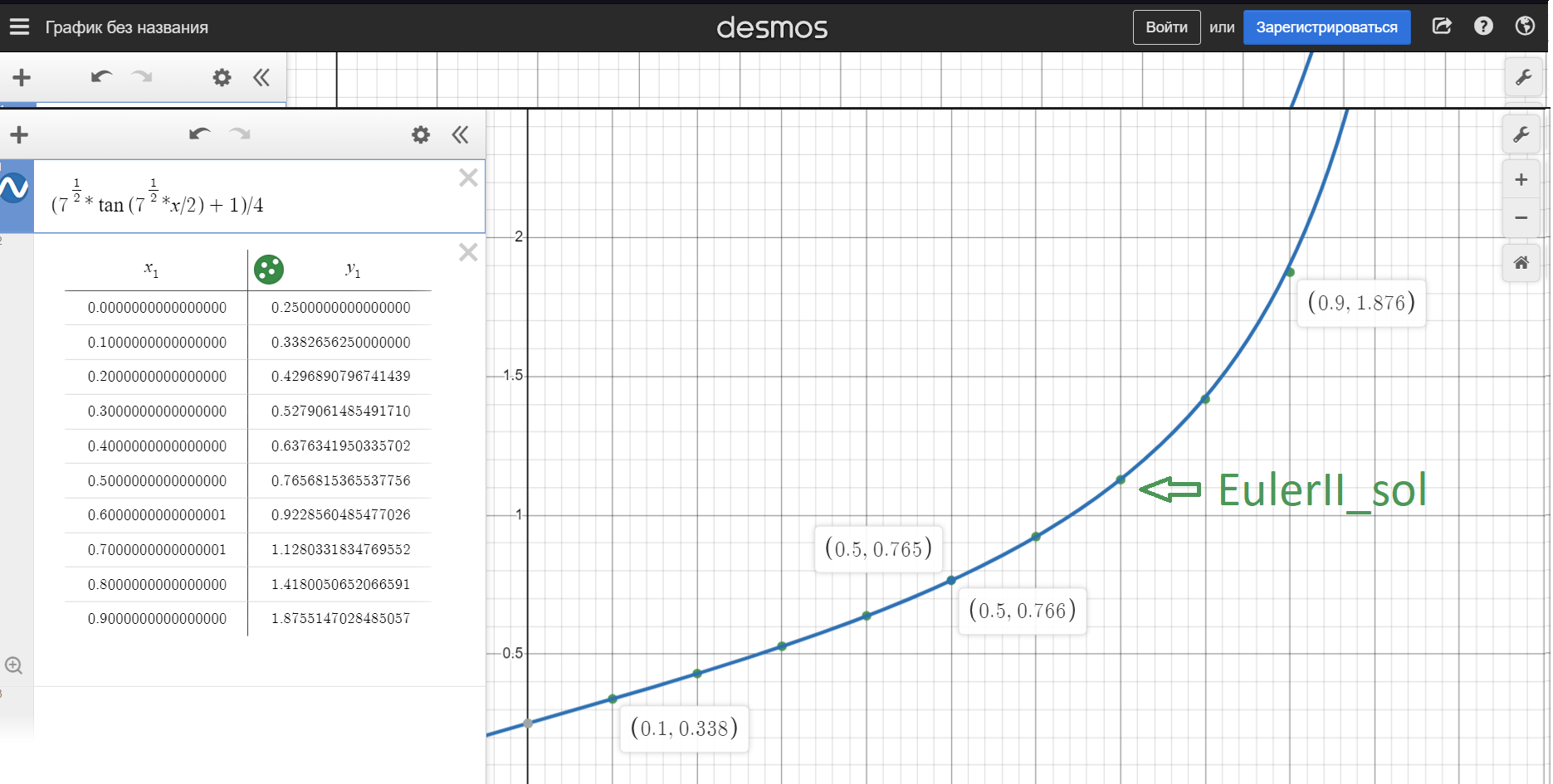












Б) Программа

#include <iostream>

#include <iomanip>

#include <vector>

double f(double y) {

return 2 \* y \* y - y + 1;

}

double solution(double x) { //Посчитано "руками"

return (sqrt(7) \* tan(sqrt(7) \* x / 2) + 1) / 4;

}

double Taylor(double x) { //Посчитано "руками"

return 0.25 + 0.875 \* x + pow(x, 3) \* 49.0 / 96.0 + pow(x, 5) \* 343.0 / 960.0 + pow(x, 7) \* 40817.0 / 161280.0;

}

void Adams(const std::vector <std::pair<double, double>> &Taylor, std::vector <std::pair<double, double>> &Adams, int N) {

for (int i = 0; i < 5; ++i)

{

Adams[i] = Taylor[i];

}

for (int i = 5; i < N; ++i) {

Adams[i].first = Taylor[i].first;

}

for (int k = 5; k < N; ++k)

{

std::vector<double> l(5, 1.0);

for (int i = 0; i < 5; ++i)

{

for (int j = 0; j < 5; ++j)

{

if (i != j) {

l[i] \*= (Adams[k].first - Adams[j].first) / (Adams[i].first - Adams[j].first);

}

}

}

for (int i = 0; i < 5; ++i)

{

Adams[k].second += Adams[i].second \* l[i];

}

}

}

void RungeKutt(std::vector <std::pair<double, double>>& RungeKutt, double h) {

double y = 0.25;

int N = RungeKutt.size();

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

RungeKutt[i].first = i \* h;

}

RungeKutt[0].second = y;

for (int i = 1; i < N; ++i)

{

double y\_prev = RungeKutt[i - 1].second;

double k1 = h \* f(y\_prev);

double k2 = h \* f(y\_prev + k1 / 2);

double k3 = h \* f(y\_prev + k2 / 2);

double k4 = h \* f(y\_prev + k3);

RungeKutt[i].second = y\_prev + 1.0 / 6.0 \* (k1 + 2 \* k2 + 2 \* k3 + k4);

}

}

void Euler(std::vector<std::pair<double, double>> &Euler, double h) {

const int N = Euler.size();

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

Euler[i].first = i \* h;

}

Euler[0].second = 0.25;

for (int i = 1; i < N; ++i)

{

double y\_prev = Euler[i - 1].second;

Euler[i].second = y\_prev + h \* f(y\_prev);

}

}

void EulerI(std::vector<std::pair<double, double>> &EulerI, double h)

{

const int N = EulerI.size();

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

EulerI[i].first = i \* h;

}

EulerI[0].second = 0.25;

for (int i = 1; i < N; ++i) {

double y\_prev = EulerI[i - 1].second;

double y\_half = y\_prev + h / 2 \* f(y\_prev);

EulerI[i].second = y\_prev + h \* f(y\_half);

}

}

void EulerII(std::vector<std::pair<double, double>> &EulerII, double h)

{

const int N = EulerII.size();

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

EulerII[i].first = i \* h;

}

EulerII[0].second = 0.25;

for (int i = 1; i < N; ++i)

{

double y\_prev = EulerII[i - 1].second;

double Y = y\_prev + h \* f(y\_prev);

EulerII[i].second = y\_prev + h / 2 \* (f(y\_prev) + f(Y));

}

}

int main() {

std::cout << std::fixed << std::setprecision(16);

std::cout << "Input length of step: ";

double h = 0.1; //h = 0.1

//std::cin >> h;

std::cout << "Input N - number of steps + 2: ";

int N = 10; //N = 10

//std::cin >> N;

std::cout << "Table of values: \n" << std::setw(15) << "X\_i" << std::setw(20) << "Solution(X\_i)\n";

std::vector <std::pair<double, double>> exact\_sol(N + 2);

for (int i = -2; i < N; ++i) {

exact\_sol[i+2].first = i \* h;

exact\_sol[i+2].second = solution(exact\_sol[i+2].first);

std::cout << std::setw(19) << exact\_sol[i + 2].first << "|" << exact\_sol[i + 2].second << "\n";

}

std::vector <std::pair<double, double>> Taylor\_sol(N + 2);

std::cout << "\nTable of Taylor values: \n" << std::setw(15) << "X\_i" << std::setw(20) << "Taylor(X\_i)" << std::setw(34) << "|Taylor(X\_i) - Solution(X\_i)|\n";

for (int i = -2; i < N; ++i) {

Taylor\_sol[i + 2].first = i \* h;

Taylor\_sol[i + 2].second = Taylor(Taylor\_sol[i+2].first);

std::cout << std::setw(19) << Taylor\_sol[i + 2].first << "|" << std::setw(19) << Taylor\_sol[i + 2].second << "|" << abs(Taylor\_sol[i + 2].second - exact\_sol[i + 2].second) << "\n";

}

std::vector <std::pair<double, double>> Adams\_sol(N + 2);

std::cout << "\nTable of Adams values: \n" << std::setw(15) << "X\_i" << std::setw(20) << "Adams(X\_i)" << std::setw(34) << "|Adams(X\_i) - Solution(X\_i)|\n";

Adams(Taylor\_sol, Adams\_sol, N + 2);

for (int i = -2; i < N; ++i) {

std::cout << std::setw(19) << Adams\_sol[i + 2].first << "|" << std::setw(19) << Adams\_sol[i + 2].second << "|" << abs(Adams\_sol[i + 2].second - exact\_sol[i + 2].second) << "\n";

}

std::vector <std::pair<double, double>> RungeKutt\_sol(N);

std::cout << "\nTable of RungeKutt values: \n" << std::setw(15) << "X\_i" << std::setw(20) << "RungeKutt(X\_i)" << std::setw(34) << "|RungeKutt(X\_i) - Solution(X\_i)|\n";

RungeKutt(RungeKutt\_sol,h);

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

std::cout << std::setw(19) << RungeKutt\_sol[i].first << "|" << std::setw(19) << RungeKutt\_sol[i].second << "|" << abs(RungeKutt\_sol[i].second - exact\_sol[i+2].second) << "\n";

}

std::vector <std::pair<double, double>> Euler\_sol(N);

std::cout << "\nTable of Euler values: \n" << std::setw(15) << "X\_i" << std::setw(20) << "Euler(X\_i)" << std::setw(34) << "|Euler(X\_i) - Solution(X\_i)|\n";

Euler(Euler\_sol, h);

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

std::cout << std::setw(19) << Euler\_sol[i].first << "|" << std::setw(19) << Euler\_sol[i].second << "|" << abs(Euler\_sol[i].second - exact\_sol[i + 2].second) << "\n";

}

std::vector <std::pair<double, double>> EulerI\_sol(N);

std::cout << "\nTable of EulerI values: \n" << std::setw(15) << "X\_i" << std::setw(20) << "EulerI(X\_i)" << std::setw(34) << "|EulerI(X\_i) - Solution(X\_i)|\n";

EulerI(EulerI\_sol, h);

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

std::cout << std::setw(19) << EulerI\_sol[i].first << "|" << std::setw(19) << EulerI\_sol[i].second << "|" << abs(EulerI\_sol[i].second - exact\_sol[i + 2].second) << "\n";

}

std::vector <std::pair<double, double>> EulerII\_sol(N);

std::cout << "\nTable of EulerII values: \n" << std::setw(15) << "X\_i" << std::setw(20) << "EulerII(X\_i)" << std::setw(34) << "|EulerII(X\_i) - Solution(X\_i)|\n";

EulerII(EulerII\_sol, h);

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

std::cout << std::setw(19) << EulerII\_sol[i].first << "|" << std::setw(19) << EulerII\_sol[i].second << "|" << abs(EulerII\_sol[i].second - exact\_sol[i + 2].second) << "\n";

}

return 0;

}

https://github.com/Chernovuk/Chislaki/tree/master/Sixth\_task