МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Вятский государственный университет»
(ФГБОУ ВО «ВятГУ»)
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

ЧИСЛЕННОЕ ИНТЕГРИРОВАНИЕ И РЕШЕНИЕ ОБЫКНОВЕННЫХ ДИФФЕРЕНЦИАЛЬНЫХ УРАВНЕНИЙ

Отчет по лабораторной работе №4 дисциплины
«Вычислительная математика»

Выполнил студент группы ИВТ-21 ______________________________/Рзаев А.Э./
Проверил преподаватель _________________________________/Архангельский В.В./

2016 г.

1 Постановка задачи

1.1 Вычислить определённый интеграл с точностью до 0,0001.
Выбрать значение n, обеспечивающее заданную точность, из формулы остаточного члена.

Задание:
Определённый интеграл от функции: 1/sqrt(0,5*x^2+1,5)
Пределы интегрирования: [1,2;2,0]
Использовать формулу трапеций.

1.2 Вычислить определённый интеграл с точностью до 0,0001 по другой квадратурной формуле:

Задание:
Определённый интеграл от функции: (x+1)*cos(x^2)
Пределы интегрирования: [0,2;1,0]
Использовать формулу Симпсона.
В качестве начального шага взять число, близкое к E^(1/m), где m=4.
Для приближённой оценки погрешности применить принцип Рунге.

1.3 Вычислить определённый интеграл по квадратурной формуле Гаусса.
Для оценки погрешности взять различное количество узлов: n1=4; n2=7.

Квадратурная формула Гаусса с 4 узлами:
x1=-x4=-0,86114 A1=A4=0,34785
x2=-x3=-0,33998 A2=A3=0,65215

Квадратурная формула Гаусса с 7 узлами:
x1=-x7=-0,949107912 A1=A7=0,129484966
x2=-x6=-0,741531186 A2=A6=0,279705391
x3=-x5=-0,405845151 A3=A5=0,381830051
x4=0 A4=0,417959184

Задание:
Определённый интеграл от функции: (x^2+2)/sqrt(x^2+1)
Пределы интегрирования: [-0,4;1,8]

1.4 Определить значения всех интегралов, используя Wolphram Alpha.

1.5 Решить обыкновенное дифференциальное уравнение.
Решение представить в табличной и графической формах.
Для оценки погрешности выполнить расчёт с шагом h и с шагом h/2.

Задание:

По формуле 4-го порядка точности решить дифференциальное уравнение методом Адамса:

y'=2*x+y

y(0)=1; h=0,1; 0<=x<=1; нач. отр. [1;0,9145;0,8562;0,8225]

2 Ход выполнения

2.1 Метод трапеций

Метод трапеций — метод численного интегрирования функции одной переменной, заключающийся в замене на каждом элементарном отрезке подынтегральной функции на многочлен первой степени, то есть линейную функцию. Площадь под графиком функции аппроксимируется прямоугольными трапециями.

Если отрезок $[a, b]$ является элементарным и не подвергается дальнейшему разбиению, приближенное значение интеграла можно найти по формуле:

$$\int_a^b f(x)dx \approx \frac{f(a)+f(b)}{2}(b-a).$$

Погрешность результатов можно оценить через максимум второй производной:

$$|E(f)| \leq \frac{(b-a)^3}{12} \max_{x\in[a,b]} |f''(x)|.$$

Если отрезок $[a, b]$ разбивается узлами интегрирования и на каждом из элементарных отрезков применяется формула трапеций, то суммирование даст составную формулу трапеций:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n-1} \frac{f(x_i)+f(x_{i+1})}{2}(x_{i+1}-x_i).$$

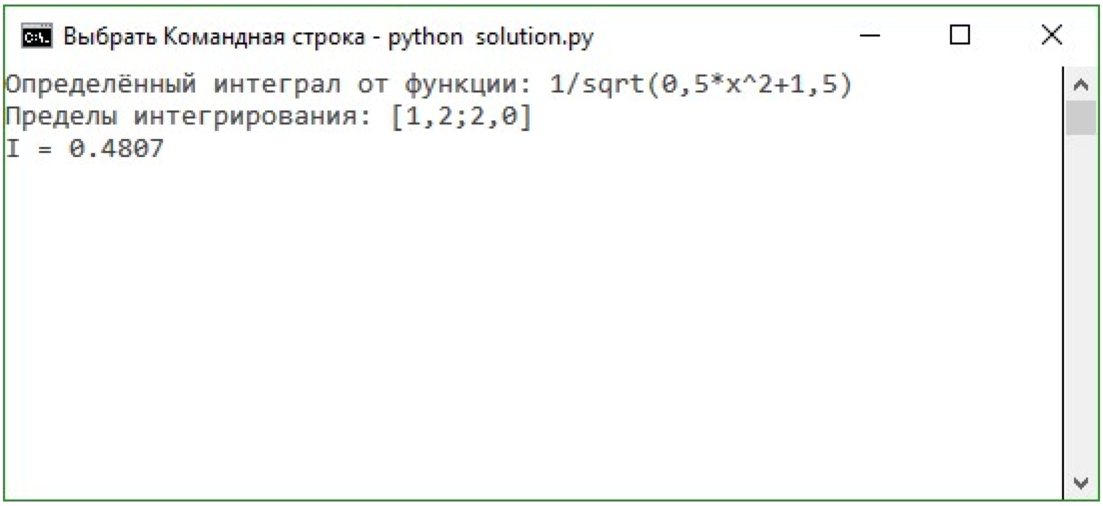


Рисунок 1 – Результат выполнения 1-го задания в программе

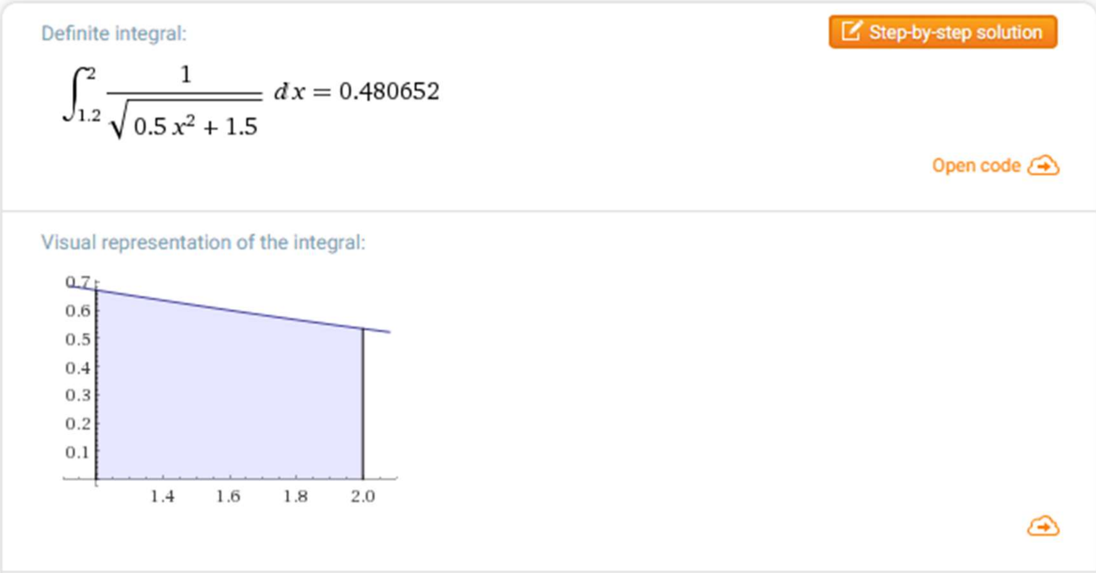


Рисунок 2 – Проверка 1-го задания в Wolphram Alpha

2.2 Метод Симпсона

Суть метода заключается в приближении подынтегральной функции на отрезке $[a,b]$ интерполяционным многочленом второй степени $p_2(x)$, то есть приближение графика функции на отрезке параболой.

Формулой Симпсона называется интеграл от интерполяционного многочлена второй степени на отрезке $[a,b]$:

$$\int_a^b f(x)dx \approx \int_a^b p_2(x)dx = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right).$$

Для более точного вычисления интеграла, интервал $[a,b]$ разбивают на $N = 2n$ отрезков одинаковой длины и применяют формулу Симпсона на каждой соседней паре из них. Значение исходного интеграла является суммой результатов интегрирования на всех отрезках:

$$\int_a^b f(x)dx \approx \frac{h}{3}\left[f(x_0) + 2\sum_{j=1}^{n-1} f(x_{2j}) + 4\sum_{j=1}^{n} f(x_{2j-1}) + f(x_N)\right],$$

где $h = \frac{b-a}{N}$ – величина шага, $x_j = a + jh$ – узлы интегрирования.

Общая погрешность $E(f)$ определяется по формуле:

$$|E(f)| \le \frac{b-a}{2880} h^4 \max_{x\in[a,b]} \left|f^{(4)}(x)\right|.$$

При невозможности оценить погрешность с помощью максимума четвёртой производной (например, на заданном отрезке она не существует, либо стремится к бесконечности), можно использовать более грубую оценку:

$$|E(f)| \le \frac{b-a}{288} h^3 \max_{x\in[a,b]} \left|f^{(3)}(x)\right|.$$

Принцип Рунге: интеграл вычисляется для последовательных значений числа шагов $N = n_0, 2n_0, 4n_0, \ldots$, где n_0 — начальное число шагов. Процесс вычислений заканчивается, когда для очередного значения N будет выполнено условие $\Delta_{2n} < \varepsilon$, где ε — заданная точность.

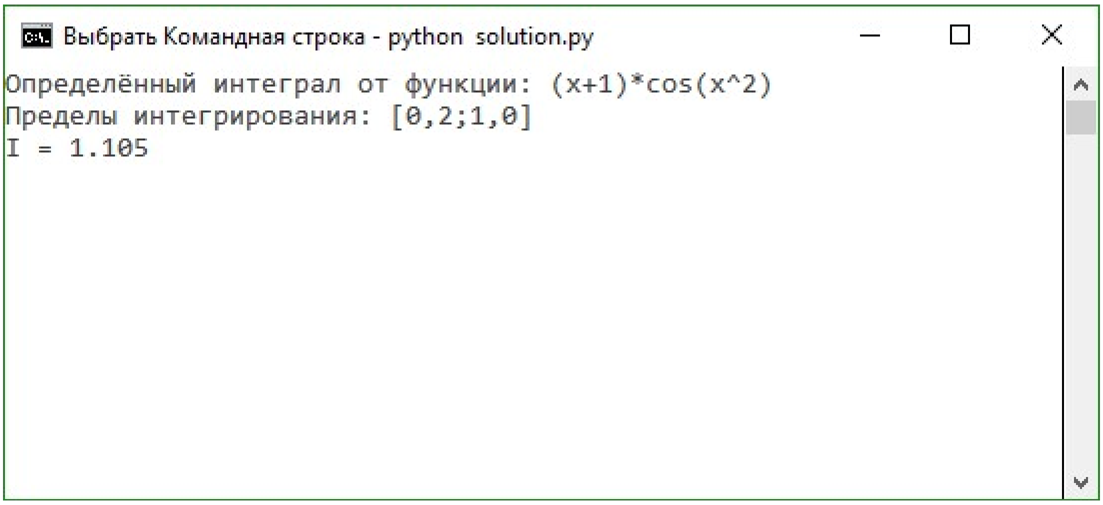


Рисунок 3 – Результат выполнения 2-го задания в программе

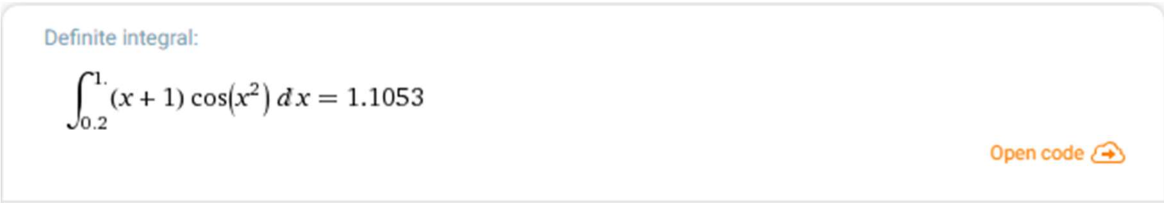


Рисунок 4 – Проверка 2-го задания в Wolphram Alpha

2.3 Метод Гаусса

Если по условиям задачи имеется право выбора узлов квадратурной формулы, то для вычисления интеграла применяют квадратурные формулы типа Гаусса:

$$\int_a^b p(x) f(x) dx \approx \sum_{i=0}^{n} A_i f(x_i) \qquad (1)$$

Коэффициенты A_i и узлы x_i квадратурных формул выбираются так, чтобы приближенное равенство (1) было бы точным для всех многочленов наивысшей возможной степени. Квадратурные формулы типа Гаусса степени n будут точными для всех полиномов степени не выше $2n - 1$, тогда как формулы Ньютона-Котеса точны только для полиномов степени не выше n.

Квадратурная формула Гаусса

$$\int_{-1}^{1} f(x) dx \approx \sum_{i=0}^{n} A_i f(x_i)$$

получена с весовой функцией $p(x) \in 1$ и узлами x_i, являющимися корнями полиномов Лежандра:

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n(x^2-1)^n}{dx^n}.$$

Коэффициенты A_i вычисляются по формулам:

$$A_i = \frac{2}{(1-x^2)\left[P_n'(x_i)\right]^2}.$$

При необходимости вычислить интеграл с другими пределами

$$\int_a^b f(x)dx$$

следует сделать замену переменных: $t = \frac{b-a}{2}x + \frac{a+b}{2}$, тогда формула примет вид:

$$\int_a^b f(x)dx \approx \frac{b-a}{2}\sum_{i=0}^{n} A_i f\left(\frac{b-a}{2}x_i + \frac{a+b}{2}\right).$$

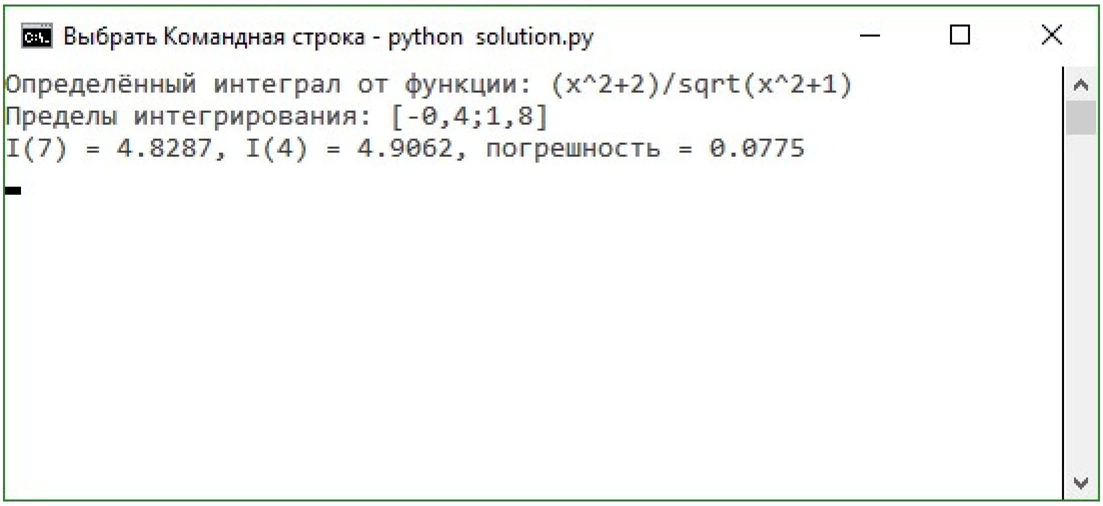


Рисунок 5 – Результат выполнения 3-го задания в программе

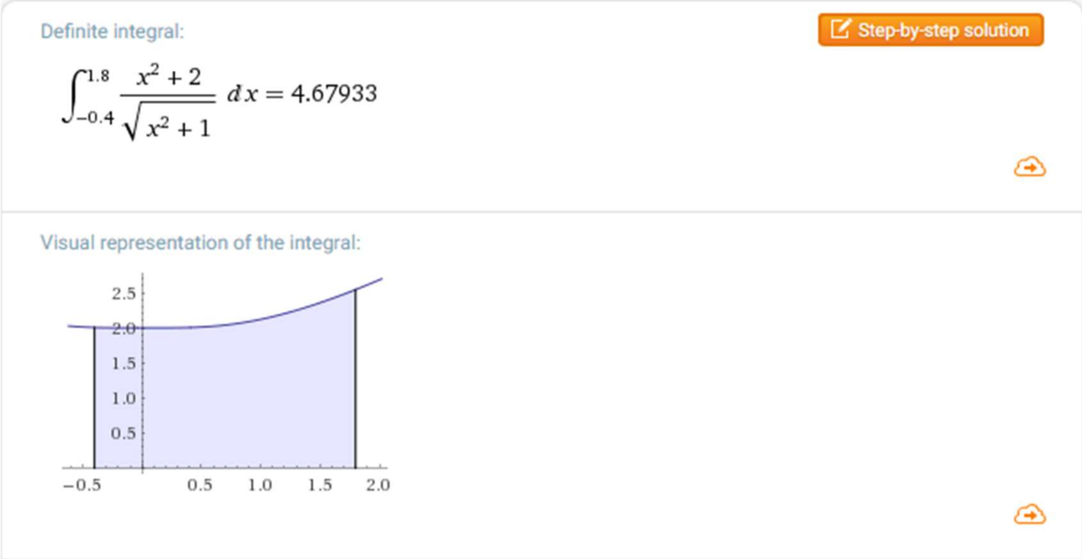


Рисунок 6 – Проверка 3-го задания в Wolphram Alpha

2.4 Метод Адамса для решения дифференциальных уравнений

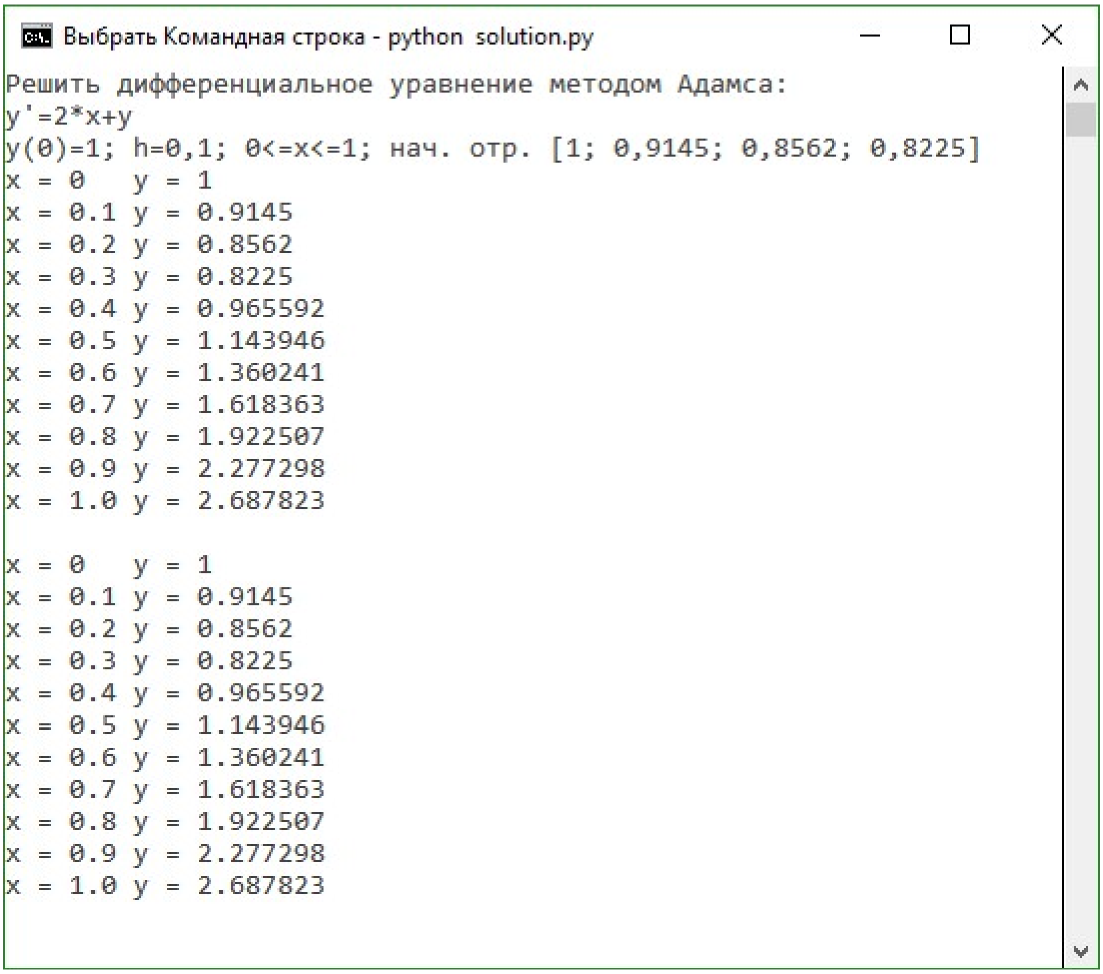
Широко распространенным семейством многошаговых методов являются методы Адамса. Простейший из них, получающийся при $k = 1$, совпадает с методом Эйлера первого порядка точности. В практических расчетах чаще всего применяют вариант метода Адамса, имеющий четвертый порядок точности и использующий на каждом шаге результаты предыдущих четырех. Именно его и называют обычно методом Адамса. Рассмотрим данный метод для уравнений вида $y' = f(x, y)$.

Пусть найдены значения $y_{i-3}, y_{i-2}, y_{i-1}, y_i$ в четырех последовательных узлах (k = 4). При этом имеются также вычисленные ранее значения правой части $f_{i-3}, f_{i-2}, f_{i-1}, f_i$ где $f_j = f(x_j, y_j)$. В случае постоянного шага h конечные разности для правой части в узле x_i имеют вид

$$\Delta f_i = f_i - f_{i-1},$$
$$\Delta^2 f_i = f_i - 2f_{i-1} + f_{i-2},$$
$$\Delta^3 f_i = f_i - 3f_{i-1} + 3f_{i-2} - f_{i-3}.$$

Тогда разностную схему четвертого порядка метода Адамса можно записать после необходимых преобразований в виде

$$y_{i+1} = y_i + hf_i + \frac{h^2}{2}\Delta f_i + \frac{5h^3}{12}\Delta^2 f_i + \frac{3h^4}{8}\Delta^3 f_i.$$



Выбрать Командная строка - python solution.py

```
Решить дифференциальное уравнение методом Адамса:
y'=2*x+y
y(0)=1; h=0,1; 0<=x<=1; нач. отр. [1; 0,9145; 0,8562; 0,8225]
x = 0   y = 1
x = 0.1 y = 0.9145
x = 0.2 y = 0.8562
x = 0.3 y = 0.8225
x = 0.4 y = 0.965592
x = 0.5 y = 1.143946
x = 0.6 y = 1.360241
x = 0.7 y = 1.618363
x = 0.8 y = 1.922507
x = 0.9 y = 2.277298
x = 1.0 y = 2.687823

x = 0   y = 1
x = 0.1 y = 0.9145
x = 0.2 y = 0.8562
x = 0.3 y = 0.8225
x = 0.4 y = 0.965592
x = 0.5 y = 1.143946
x = 0.6 y = 1.360241
x = 0.7 y = 1.618363
x = 0.8 y = 1.922507
x = 0.9 y = 2.277298
x = 1.0 y = 2.687823
```

Рисунок 7 – Результат выполнения 4-го задания в программе (таблица)

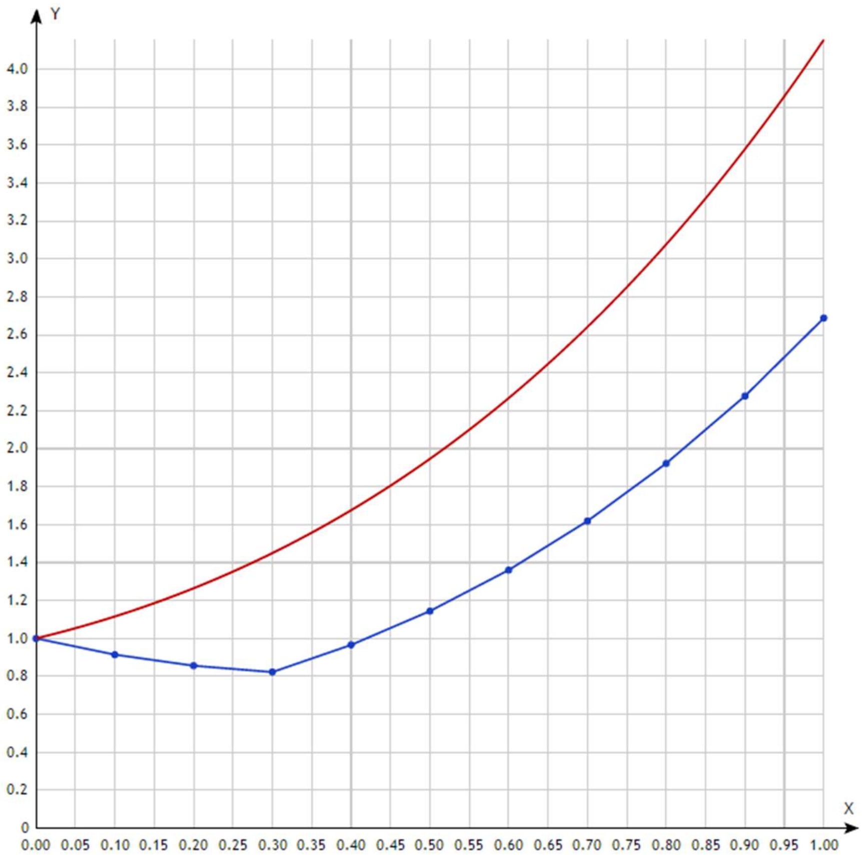


Рисунок 8 – Результат выполнения 4-го задания в программе (график)

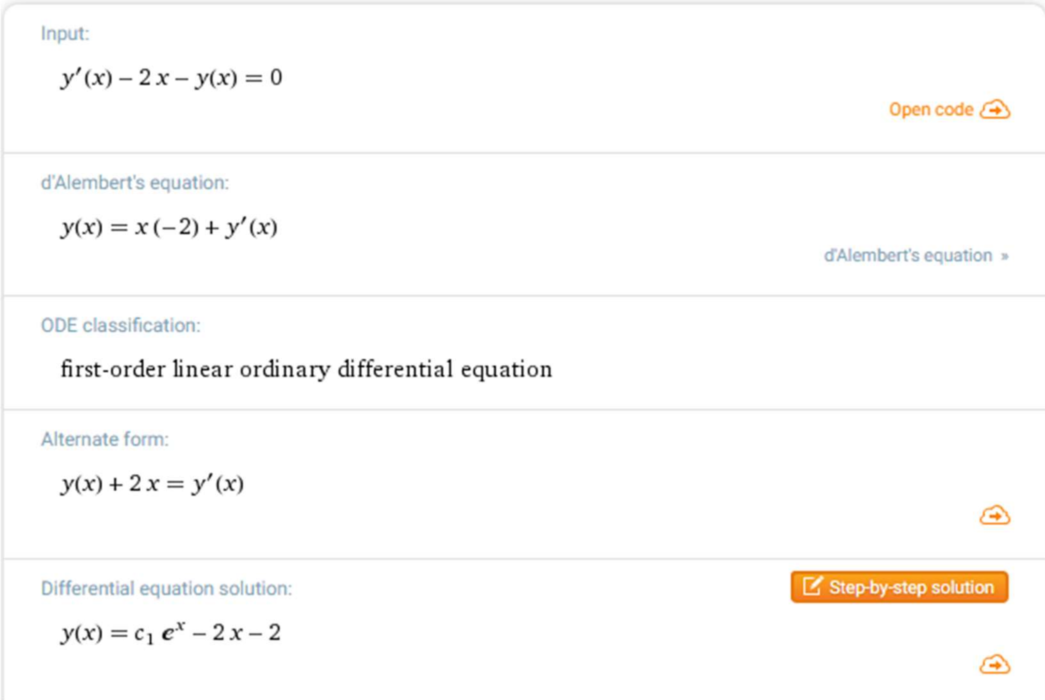


Рисунок 9 – Решение уравнения в Wolphram Alpha

3 Листинг

Листинг программы представлен в приложении А.

4 Выводы

В ходе выполнения лабораторной работы, для достижения конечного результата был изучен материал по темам численного интегрирования и методам решения обыкновенных дифференциальных уравнений. По теме численного интегрирования были изучены такие методы, как: метод трапеций, метод Симпсона (парабол) и метод Гаусса. По теме дифференциальных уравнений был изучен метод Адамса.

После изучения материала, была реализована программа, в которой выполняется решение заданий для данной лабораторной работы указанными методами. Для проверки решений интегралов использовался сервис Wolfram Apha. В трех из четырех заданий результаты сошлись, в одном – небольшое расхождение с правильным ответом.

Приложение А
(обязательное)
Листинг кода программы

```
from math import *
from functools import reduce


def trapezoid(func, derivation, rng, eps):
    a, b = rng
    n = abs(round((b - a) ** 3 * derivation(a) / (12 * eps)))
    h = (b - a) / n

    integral = sum(h * (func(a + h * i) + func(a + h * (i + 1))) / 2 for i in
range(n))

    return round(integral, int(-log10(eps)))


func_trapezoid = lambda x: (0.5 * x ** 2 + 1.5) ** -0.5

deriv_trapezoid = lambda x: -0.5 * x * (0.5 * x ** 2 + 1.5) ** -1.5

rng_trapezoid = (1.2, 2.0)

eps_trapezoid = 0.0001

print('''Определённый интеграл от функции: 1/sqrt(0,5*x^2+1,5)
Пределы интегрирования: [1,2;2,0]''')
print('I = {}'.format(trapezoid(func_trapezoid, deriv_trapezoid,
rng_trapezoid, eps_trapezoid)))


def simpson(func, rng, eps):
    n = 8
    a, b = rng

    while True:
        h1 = (b - a) / n
        h2 = (b - a) / (2 * n)

        i1 = h1 / 3 * (func(a) + func(b)
                       + 4 * sum(func(a + i * h1) for i in range(1, n - 1,
2))
                       + 2 * sum(func(a + i * h1) for i in range(2, n - 1,
2)))

        i2 = h2 / 3 * (func(a) + func(b)
                       + 4 * sum(func(a + i * h2) for i in range(1, 2 * n -
1, 2))
                       + 2 * sum(func(a + i * h2) for i in range(2, 2 * n -
1, 2)))

        if abs(i2 - i1) > eps:
            n += 1
        else:
            return round(i1, int(-log10(eps))), n
```

```
func_simpson = lambda x: (x + 1) * cos(x ** 2)

rng_simpson = (0.2, 1.0)

eps_simpson = 0.0001

print('''Определённый интеграл от функции: (x+1)*cos(x^2)
Пределы интегрирования: [0,2;1,0]''')
print('I = {}'.format(simpson(func_simpson, rng_simpson, eps_simpson)[0]))


def gauss(table1, table2, func, rng):
    a, b = rng
    res = [0.5 * (b - a) * sum(a * func((b - a) * x / 2 + (a + b) / 2) for a,
x in zip(*table)) for table in (table1, table2)]
    return [round(v, 4) for v in (*res, abs(res[0] - res[1]))]

table1_gauss = [
    (0.129484966, 0.279705391, 0.381830051, 0.417959184, 0.381830051,
0.279705391, 0.129484966),
    (-0.949107912, -0.741531186, -0.405845151, 0, 0.405845151, 0.741531186,
0.949107912)
]

table2_gauss = [
    (0.34785, 0.65215, 0.65215, 0.34785),
    (-0.86114, -0.33998, 0.33998, 0.86114)
]

func_gauss = lambda x: (x ** 2 + 2) / (x ** 2 + 1) ** 0.5

rng_gauss = (-0.4, 1.8)

print('''Определённый интеграл от функции: (x^2+2)/sqrt(x^2+1)
Пределы интегрирования: [-0,4;1,8]''')
print('I(7) = {}, I(4) = {}, погрешность = {}'.format(*gauss(table1_gauss,
table2_gauss, func_gauss, rng_gauss)))


def adams(func, begs, h, rng):
    a, b = rng
    res = []
    for d in (h, h / 2):
        ys = list(begs)
        xs = []
        fs = []
        x = a
        i = 0
        while i < len(ys):
            fs.append(func(x, ys[i]))
            xs.append(x)
            x += d
            i += 1

        while x <= b:
            d_f = fs[-1] - fs[-2]
            d2_f = fs[-1] - 2 * fs[-2] + fs[-3]
            d3_f = fs[-1] - 3 * fs[-2] + 3 * fs[-3] - fs[-4]

            y = ys[-1] + h * fs[-1] + \
```

```
                0.5 * (h ** 2) * d_f + \
                5 / 12 * (h ** 3) * d2_f + \
                3 / 8 * (h ** 4) * d3_f

            ys.append(y)
            fs.append(func(x, y))
            xs.append(x)

            x += d

        res.append((xs, ys))

    return res


func_adams = lambda x, y: 2 * x + y

h_adams = 0.1

rng_adams = (0, 1)

begs_adams = (1, 0.9145, 0.8562, 0.8225)

res = adams(func_adams, begs_adams, h_adams, rng_adams)

print('''Решить дифференциальное уравнение методом Адамса:
y'=2*x+y
y(0)=1; h=0,1; 0<=x<=1; нач. отр. [1; 0,9145; 0,8562; 0,8225]''')
for r in res:
    for x, y in zip(*res[0]):
        print('x = {}\ty = {}'.format(round(x, 6), round(y, 6)))
    print()
input()
```