Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение  
высшего образования   
«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчет защищен с оценкой \_\_\_\_\_

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Отчет

По лабораторной работе №9

**«Метод Рунге-Кутты»**

по дисциплине «Вычислительные алгоритмы»

Студент группы ПИ-02

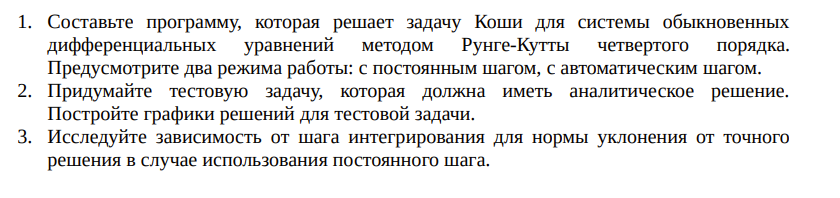
Чередов Р. А.

Преподаватель, к.ф-м.н., доцент,

Проскурин А. В.

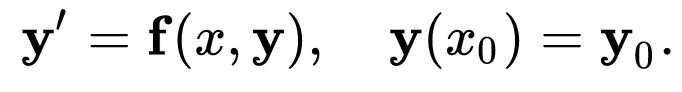
Барнаул 2023

**Задание к лабораторной работе:**

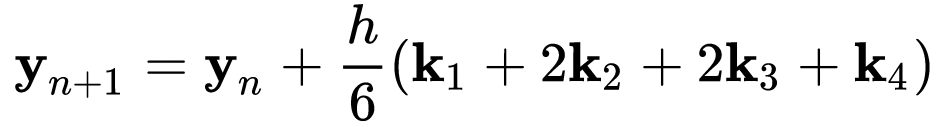


**Описание метода:**

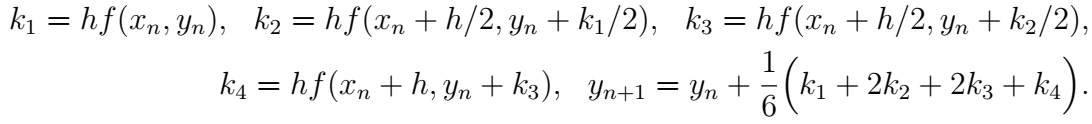
**Метод Рунге-Кутты** используют для решения [задачи Коши](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B4%D0%B0%D1%87%D0%B0_%D0%9A%D0%BE%D1%88%D0%B8) для [обыкновенных дифференциальных уравнений](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8B%D0%BA%D0%BD%D0%BE%D0%B2%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5_%D0%B4%D0%B8%D1%84%D1%84%D0%B5%D1%80%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D1%83%D1%80%D0%B0%D0%B2%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F) и их систем, он обладает точностью метода *Ο*4(*h*).

Задача Коши для системы обыкновенных дифференциальных уравнений первого порядка

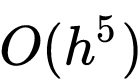
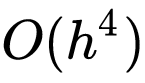
Приближенное значение в последующих точках вычисляется по итерационной формуле:



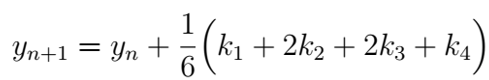
Вычисление нового значения проходит в четыре стадии:



где **h h** — величина шага сетки по **x**xxx.

Этот метод имеет четвёртый порядок точности. Это значит, что ошибка на одном шаге имеет порядок O ( h 5 ) , а суммарная ошибка на конечном интервале интегрирования имеет порядок O ( h 4 ) .

Ход решения:

1. Задаем промежуток [a, b]
2. Задаем начальное вхождение f(t), где t = 1, 2, …, j.
3. Задаем соответствующие начальным вхождениям функции.
4. Задаем точность или количество шагов.
5. При вводе количества шагов используем постоянный шаг, а при вводе точности автоматический шаг.
6. Вычисляем шаг, как , где h – шаг, n – количество шагов (итераций).
7. Находим для всех функций коэффициенты k1, k2, k3 и k4.
8. Находим dy для всех функций, берем максимальный dy.
9. Находим текущее приближение с использованием предыдущего: 

И используем его для вычисления следующего приближения.

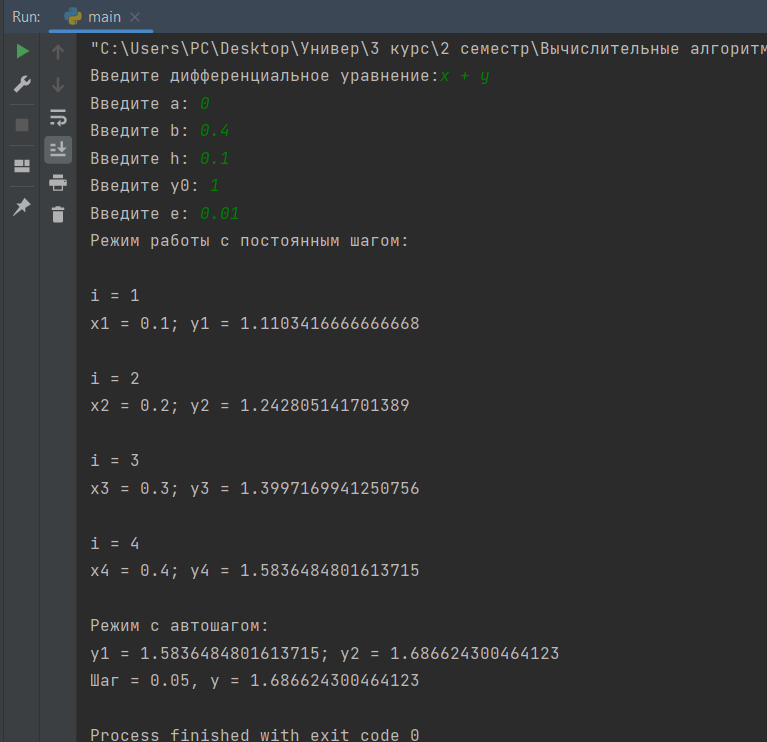
1. Повторяем, пока не дойдем до введенного количества шагов.
2. При использовании автоматического шага производим вычисления методом Рунге-Кутты (каждый раз передаем количество шагов и действуем, как описано выше для постоянного шага).
3. Если максимальное последнее вычисленное отклонение dy > ε, то считаем, что алгоритм завершил работу на текущем шаге, иначе умножаем шаг на 2 и повторяем пункт 10.
4. Получаем суммарную погрешность и изменения dy для соответствующих функций.

**Программа:**

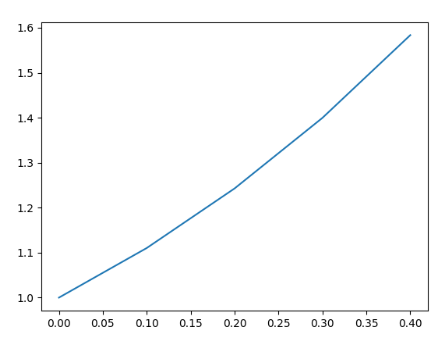
import numpy as np  
import matplotlib.pyplot as plt  
# Метод Рунге-Кутты  
def rk4\_meth(x, y, h1):  
 k1 = h1 \* eval(f, {"x": x}, {"y": y})  
 k2 = h1 \* eval(f, {"x": x + h1 / 2}, {"y": y + k1 / 2})  
 k3 = h1 \* eval(f, {"x": x + h1 / 2}, {"y": y + k2 / 2})  
 k4 = h1 \* eval(f, {"x": x + h1}, {"y": y + k3})  
 return y + (k1 + 2 \* k2 + 2 \* k3 + k4) / 6  
  
  
f = input("Введите дифференциальное уравнение:")  
a = float(input("Введите a: "))  
b = float(input("Введите b: "))  
h = float(input("Введите h: "))  
y = float(input("Введите y0: "))  
e = float(input("Введите e: "))  
  
print("Режим работы с постоянным шагом:")  
xArr = []  
res = []  
xArr.append(a)  
res.append(y)  
i = 0  
n = np.ceil((b - a) / h)  
a0 = a  
while i < n:  
 print("\ni = " + str(i + 1))  
 res.append(rk4\_meth(a0, res[i], h))  
 a0 += h  
 xArr.append(a0)  
 print("x" + str(i + 1) + " = " + str(np.around(a0, 1)) + "; y" + str(i + 1) + " = " + str(res[i + 1]))  
 i += 1  
  
print("\nРежим с автошагом:")  
  
while True:  
 y1 = []  
 y1.append(y)  
 y2 = []  
 y2.append(y)  
 i = 0  
 a0 = a  
 while a0 < b:  
 y1.append(rk4\_meth(a0, y1[i], h))  
 a0 += h  
 i += 1  
 i = 0  
 a0 = a  
 while a0 < b:  
 y2.append(rk4\_meth(a0, y2[i], h / 2))  
 a0 += h / 2  
 i += 1  
 print("y1 = " + str(y1[-1]) + "; y2 = " + str(y2[-1]))  
 if (abs(y1[-1] - y2[-1]) / 15 < e):  
 break  
 else:  
 h /= 2  
  
print("Шаг = {0}, y = {1}".format(h / 2, y2[-1]))  
# График  
plt.plot(xArr, res)  
plt.show()

**Тесты:**

1. Вычисление дифференциальное уравнение y’ = x + y при a = 0, b = 0.4, h = 0.1, y0 = 1, eps = 0.01.



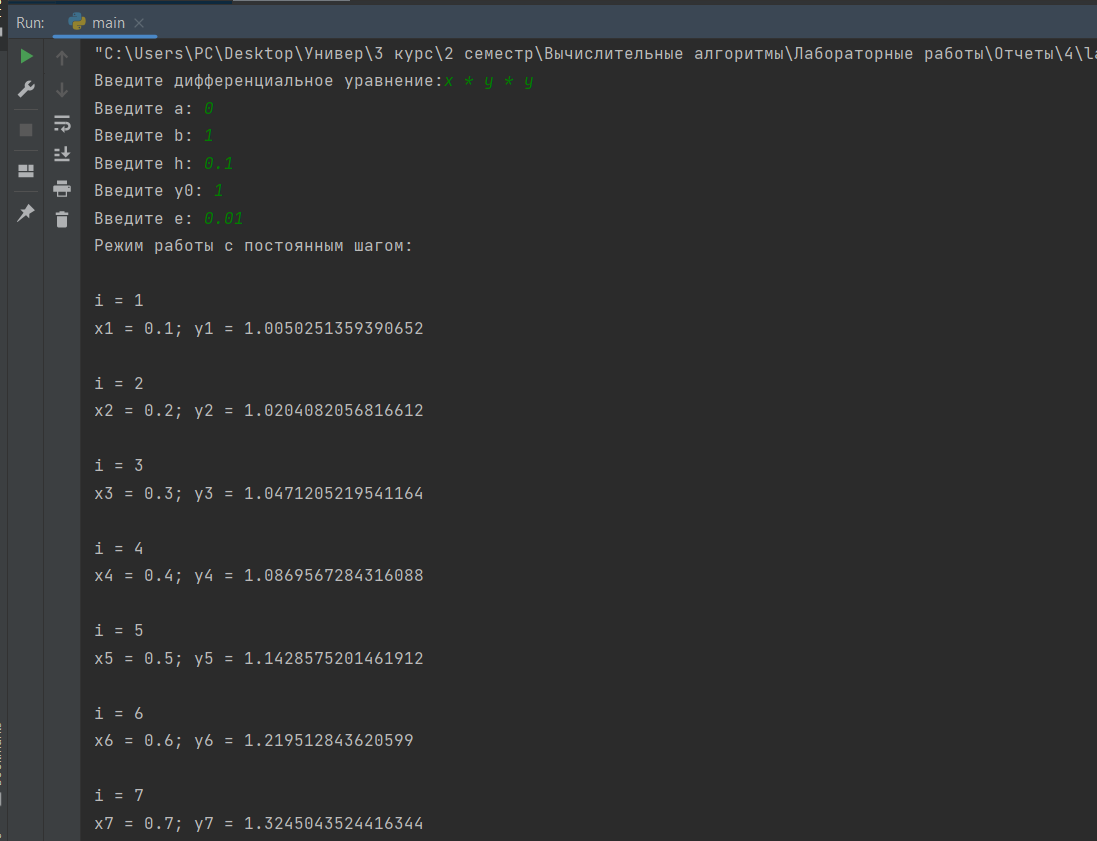
*Рис. 1.1. Результат работы программы.*

****

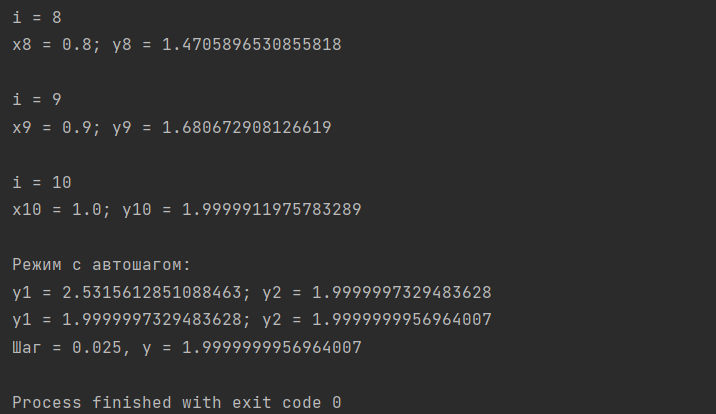
*Рис. 1.2. График функции.*

1. Вычисление дифференциальное уравнение y’ = xy2 при

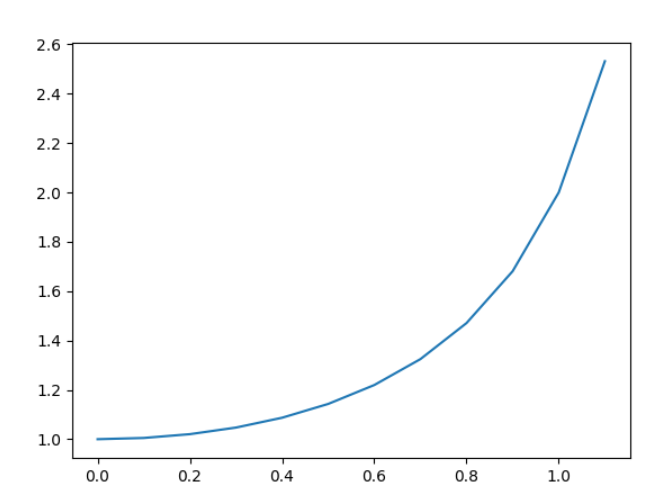
a = 0, b = 1, h = 0.1, y0 = 1, eps = 0.01.



*Рис. 2.1. Результат работы программы.*

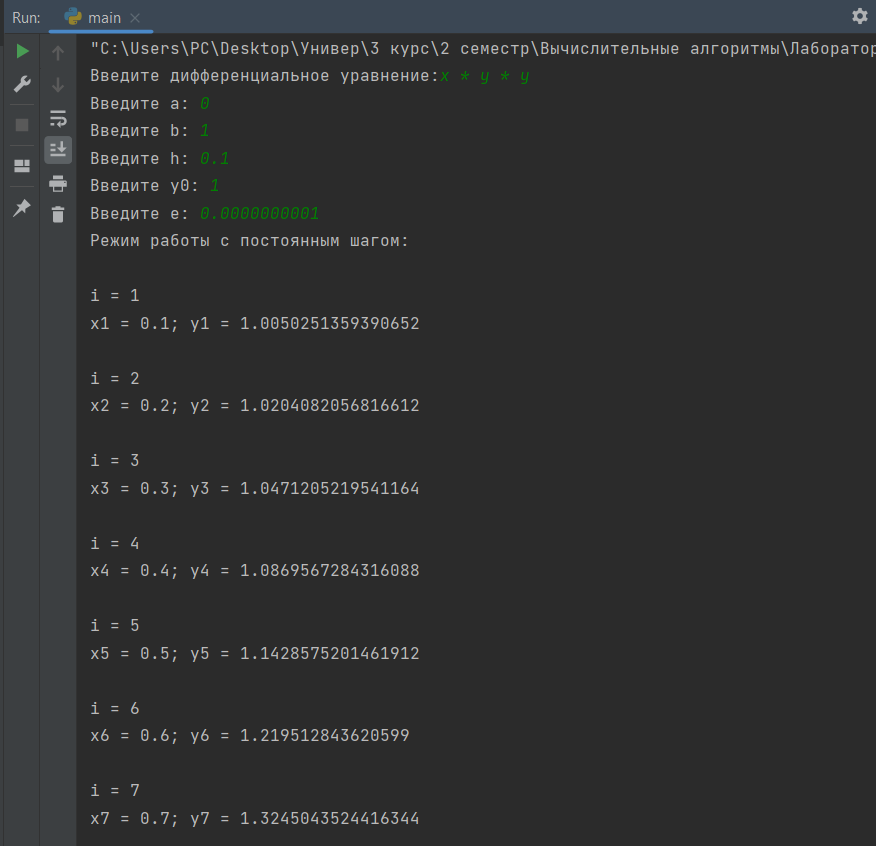
**

*Рис. 2.2. Результат работы программы.*

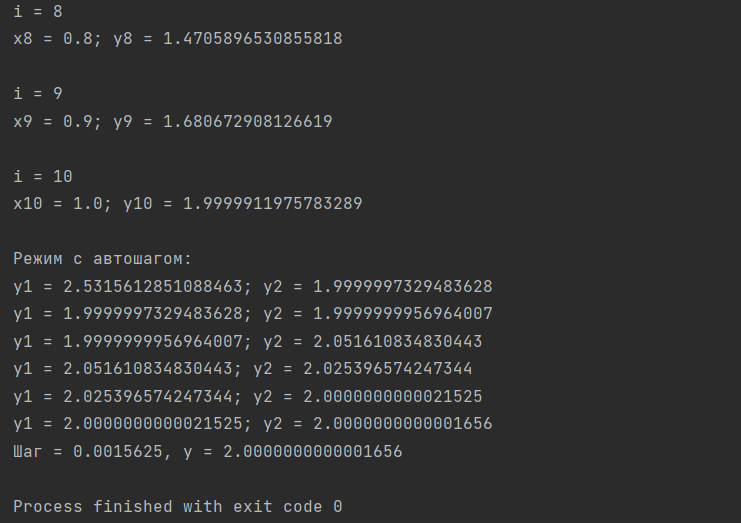
**

*Рис. 2.2. График функции.*

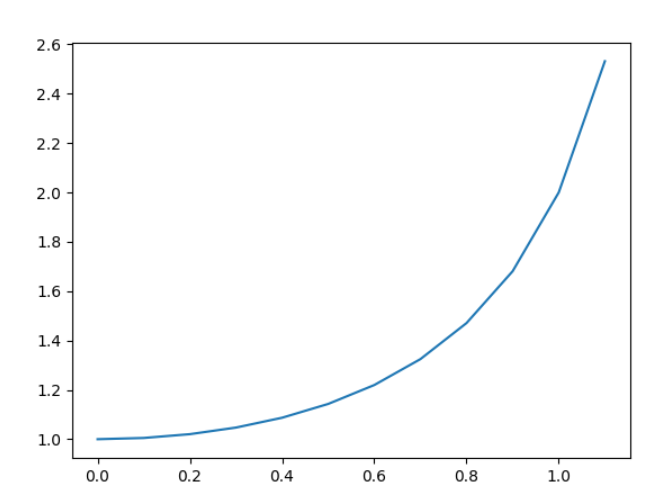
1. Вычисление дифференциальное уравнение y’ = xy2 при

a = 0, b = 1, h = 0.1, y0 = 1, eps = 0.0000000001.

*Рис. 3.1.1. Результат работы программы.*

**

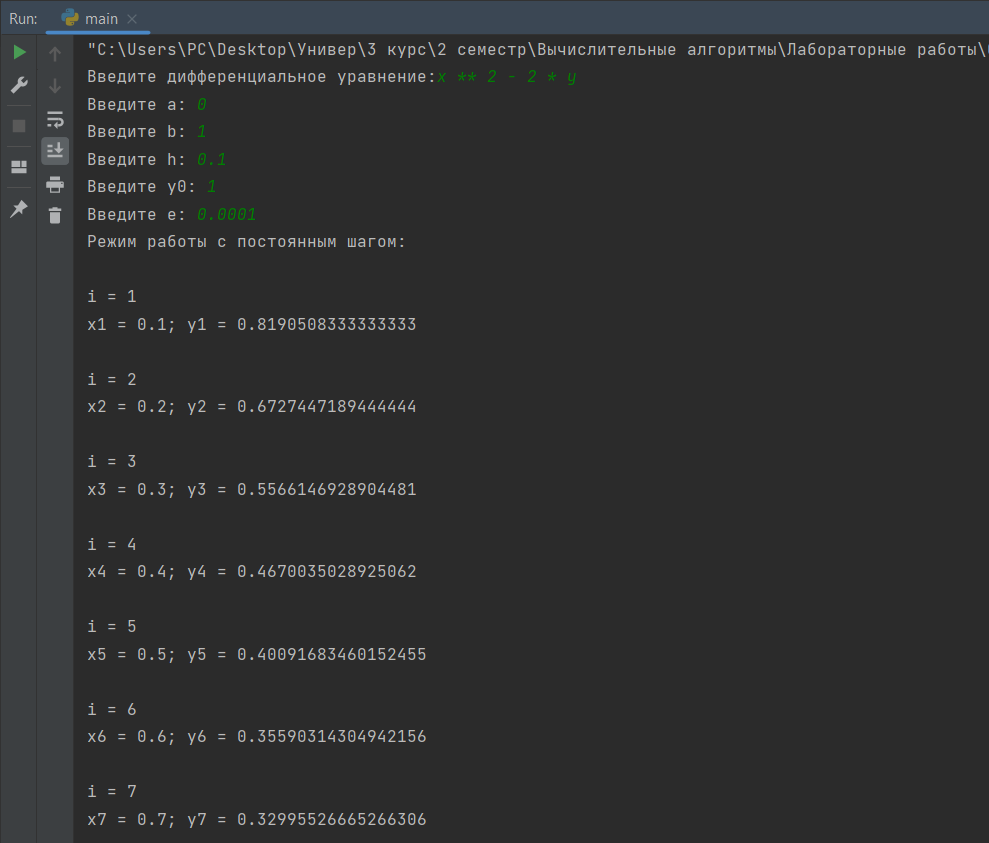
*Рис. 3.1.2. Результат работы программы.*

**

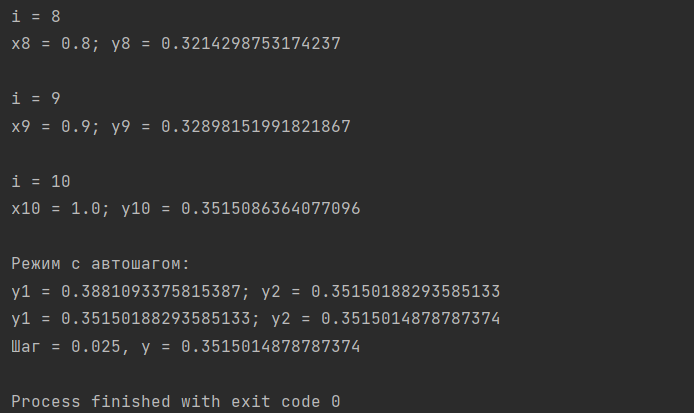
*Рис. 3.2. График функции*

1. Вычисление дифференциальное уравнение y’ = x2 - 2y при

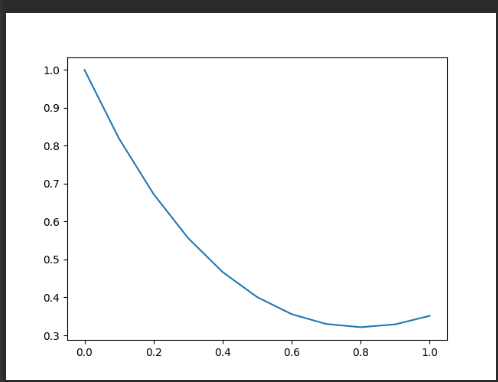
a = 0, b = 1, h = 0.1, y0 = 1, eps = 0.0001.



*Рис. 4.1.1. Результат работы программы.*

**

*Рис. 4.1.2. Результат работы программы.*



*Рис. 4.2. График функции.*

**Вывод:** согласно вычислительным экспериментам при уменьшении точности eps увеличивается количество итераций при вычислении с автошагом и уменьшается шаг. Кроме того, надо отметить, что и при статичном шаге метод довольно точный и не сильно отличается от результата при вычислении с автошагом.