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 №4

**«Интерполирование сплайнами»**

по дисциплине «Вычислительные алгоритмы»

Студент группы ПИ-02

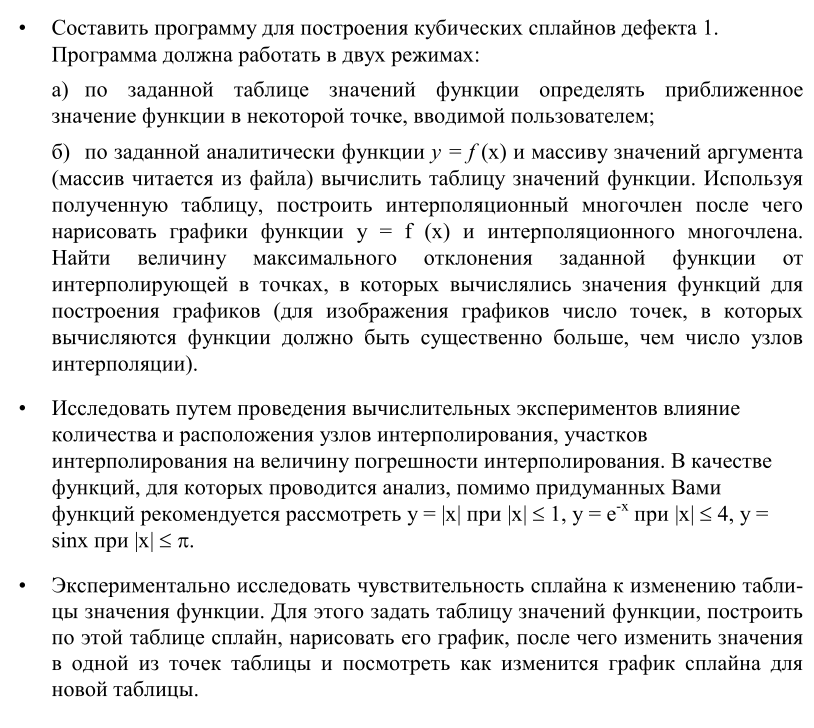
Чередов Р. А.

Преподаватель

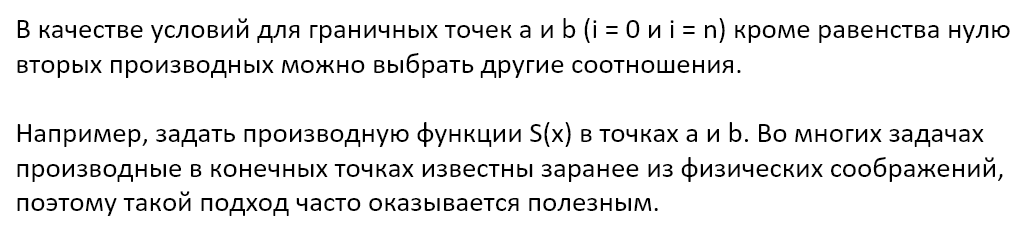
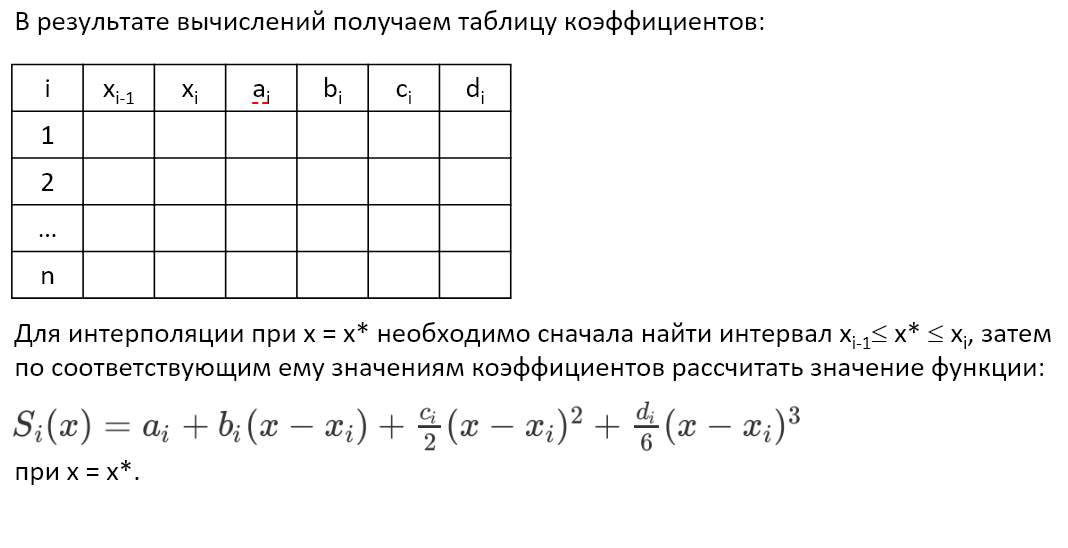
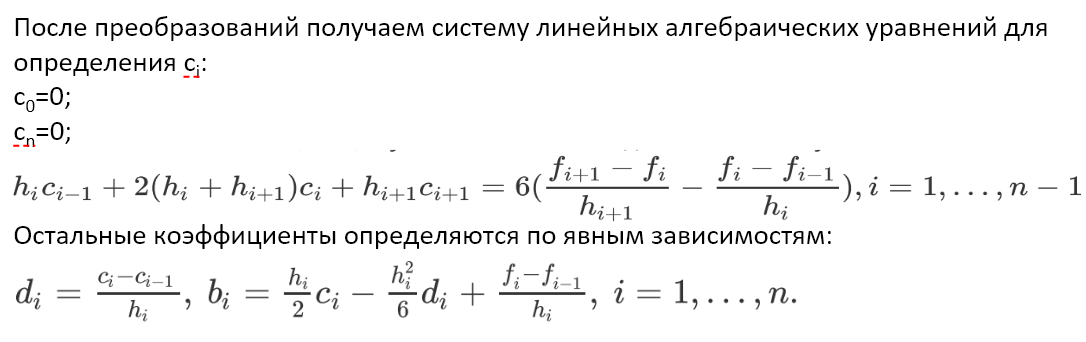
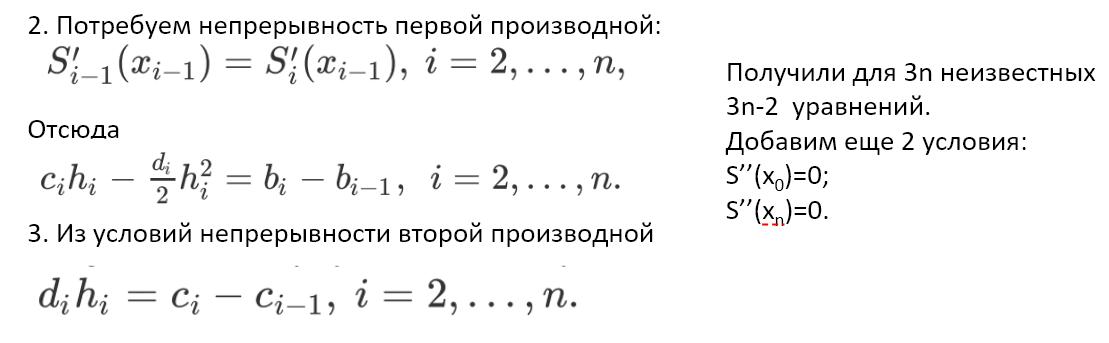
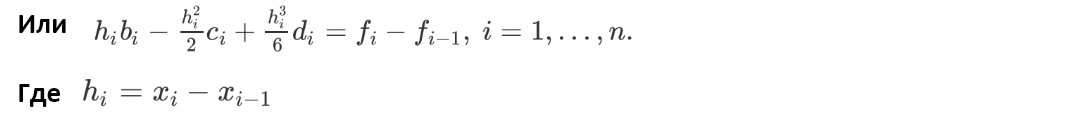
Проскурин А. В.

Барнаул 2023

**Задание к лабораторной работе:**



**Описание метода:**

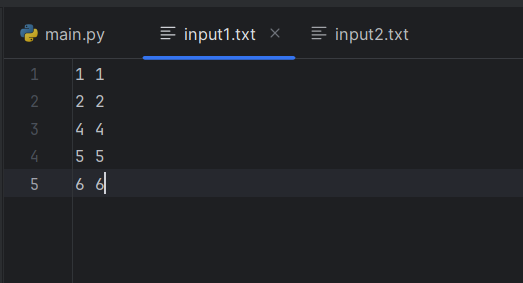
****

**Программа:**

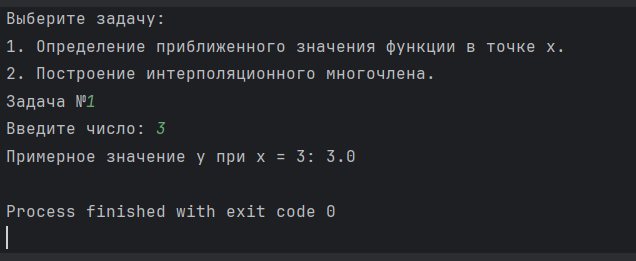
import matplotlib.pyplot as plt  
import numpy as np  
from math import \*  
  
# Сегмент сплайна  
class SplineSegment:  
 def \_\_init\_\_(self, x, a, b, c, d):  
 self.x = x  
 self.a = a  
 self.b = b  
 self.c = c  
 self.d = d  
  
# Построение сплайна  
# x - узлы сетки, y - значения функции в узлах сетки, n - количество узлов сетки  
def CreateSpline(x, y, n):  
 # Инициализация массива сплайнов  
 splines = [SplineSegment(0, 0, 0, 0, 0) for \_ in range(0, n)]  
 for i in range(0, n):  
 splines[i].x = x[i]  
 splines[i].a = y[i]  
 splines[0].c = splines[n - 1].c = 0.0  
 # Решение СЛАУ относительно коэффициентов сплайнов c[i]  
 # Прямой ход метода прогонки  
 # Вычисление прогоночных коэффициентов  
 alpha = [0.0 for \_ in range(0, n - 1)]  
 beta = [0.0 for \_ in range(0, n - 1)]  
 for i in range(1, n - 1):  
 hi = x[i] - x[i - 1]  
 hi1 = x[i + 1] - x[i]  
 k1 = hi  
 k2 = 2.0 \* (hi + hi1)  
 k3 = hi1  
 k4 = 6.0 \* ((y[i + 1] - y[i]) / hi1 - (y[i] - y[i - 1]) / hi)  
 z = (k1 \* alpha[i - 1] + k2)  
 alpha[i] = -k3 / z  
 beta[i] = (k4 - k1 \* beta[i - 1]) / z  
 # Обратный ход метода прогонки  
 for i in range(n - 2, 0, -1):  
 splines[i].c = alpha[i] \* splines[i + 1].c + beta[i]  
 # По известным коэффициентам c[i] находим значения b[i] и d[i]  
 for i in range(n - 1, 0, -1):  
 hi = x[i] - x[i - 1]  
 splines[i].d = (splines[i].c - splines[i - 1].c) / hi  
 splines[i].b = hi \* (2.0 \* splines[i].c + splines[i - 1].c) / 6.0 + (y[i] - y[i - 1]) / hi  
 return splines  
  
# Вычисление значения интерполированной функции в произвольной точке  
def Interpolate(splines, x):  
 # Проверка наличия сплайна  
 if not splines:  
 print("Сплайн еще не построен!")  
 return None  
 n = len(splines)  
 # Нахождение месторасположения x относительно промежутков сплайнов  
 if x <= splines[0].x:  
 spline0 = splines[0]  
 elif x >= splines[n - 1].x:  
 spline0 = splines[n - 1]  
 else:  
 i = 0  
 j = n - 1  
 while i + 1 < j:  
 k = i + (j - i) // 2  
 if x <= splines[k].x:  
 j = k  
 else:  
 i = k  
 spline0 = splines[j]  
 dx = x - spline0.x  
 # Вычисляем значение сплайна в заданной точке  
 return spline0.a + spline0.b \* dx + spline0.c / 2.0 \* dx \*\* 2 + spline0.d / 6.0 \* dx \*\* 3  
  
def ex1():  
 # Чтение таблицы значений  
 x1 = []  
 y1 = []  
 n = 0  
 f = open("input1.txt", 'r')  
 for line in f:  
 x1.append(float(line.split()[0]))  
 y1.append(float(line.split()[1]))  
 n += 1  
 f.close()  
 spline = CreateSpline(x1, y1, n)  
 # Вычисление интерполяции сплайнами  
 x = int(input("Введите число: "))  
 print("Примерное значение y при x = " + str(x) + ": " + str(Interpolate(spline, x)))  
  
def ex2():  
 # Задание функции  
 func = input("Введите функцию: ")  
 n = 0  
 # Чтение аргументов из файла  
 x1 = []  
 y1 = []  
 f = open("input2.txt", 'r')  
 for line in f:  
 x = float(line)  
 x1.append(x)  
 y1.append(eval(func))  
 n += 1  
 f.close()  
 y0 = []  
 x0 = np.arange(-5, 5, 0.001)  
 for x in x0:  
 y0.append(eval(func))  
 # Создание сплайна по точкам  
 spline = CreateSpline(x1, y1, len(x1))  
 polX = []  
 polY = []  
 # Интерполяция сплайнами в заданных точках  
 for i in range(n):  
 polX.append(x1[i])  
 polY.append(Interpolate(spline, i))  
 allPolX = x0  
 allPolY = []  
 # Вычислим значения во всех точках через сплайн  
 for i in allPolX:  
 allPolY.append(Interpolate(spline, i))  
 error = []  
 c = 0  
 # Отклонения  
 i = -5  
 while i < 5:  
 if i >= x1[0] and i <= x1[len(x1) - 1]:  
 error.append(abs(y0[c] - allPolY[c]))  
 c += 1  
 i += 0.001  
 plt.plot(x0, y0)  
 plt.plot(x1, y1, 'ro')  
 plt.plot(allPolX, allPolY, 'r')  
 plt.ylim([-5, 55])  
 plt.grid(True)  
 plt.show()  
 print("Максимальное отклонение:", '%.3f' % max(error))  
  
  
print("Выберите задачу: \n1. Определение приближенного значения функции в точке x.\n2. Построение интерполяционного многочлена.")  
ex = input("Задача №")  
if ex == '1':  
 ex1()  
else:  
 if ex == '2':  
 ex2()  
 else:  
 print("Введите № задачи!")

Тесты:

1. Проверка задачи №1.
   1. Привожу некоторые значения функции y = x и нахожу значение этой функции в точке 3.

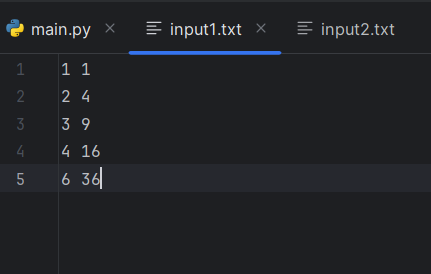


*Рис. 1.1.1. Таблица значений.*

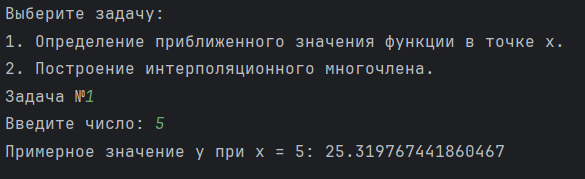


*Рис. 1.1.2. Результат вычислений.*

* 1. Привожу некоторые значения функции y = x2 и нахожу значение этой функции в точке 5.

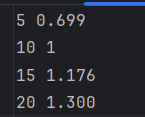


*Рис. 1.2.1. Таблица значений.*

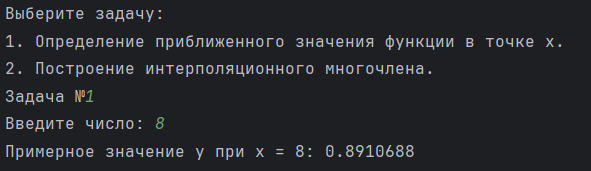


*Рис. 1.2.2. Результат вычислений.*

* 1. Привожу некоторые значения функции y = lg(x) и нахожу значение этой функции в точке 8.



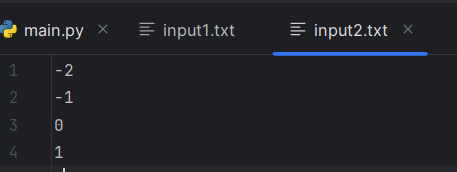
*Рис. 1.3.1. Таблица значений.*



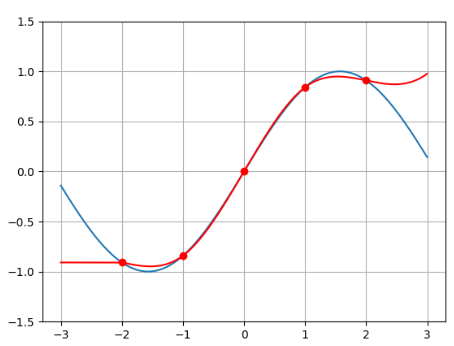
*Рис. 1.3.2. Результат вычислений.*

1. Проверка задачи №2.

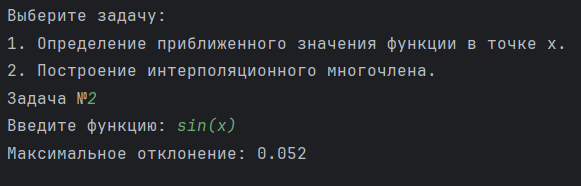
2.1. Введена функция sin(x) и 5 точек.



*Рис. 2.1.1. Значения аргумента.*

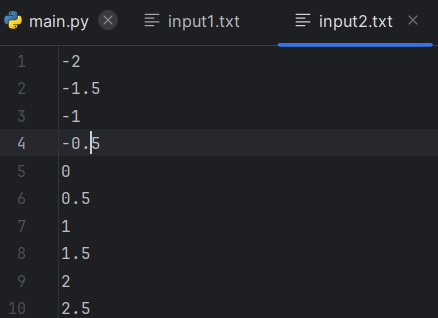
**

*Рис. 2.1.2.1. Результат вычислений.*

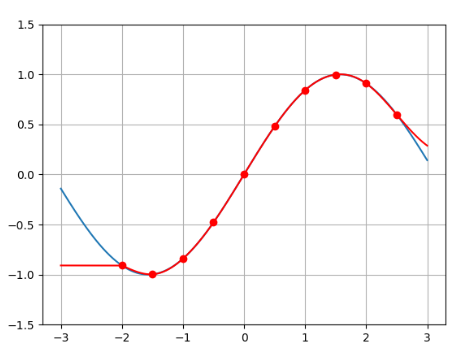


*Рис. 2.1.2.2. Результат вычислений.*

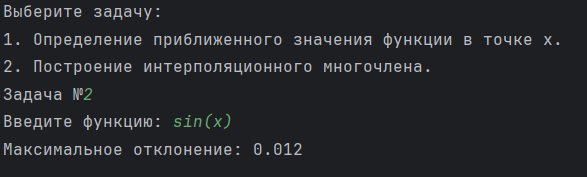
2.2. Введена функция sin(x) и 10 точек.



*Рис. 2.2.1. Значения аргумента.*

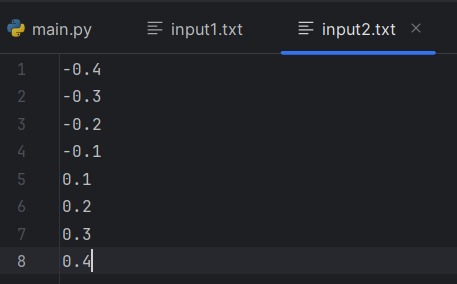
**

*Рис. 2.2.2.1. Результат вычислений.*

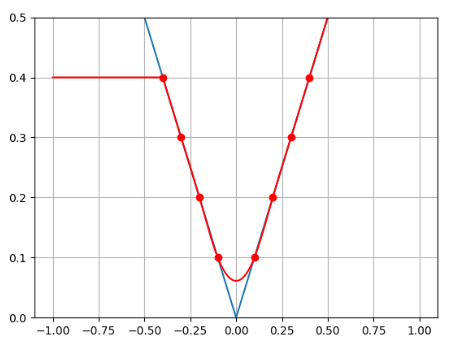


*Рис. 2.2.2.2. Результат вычислений.*

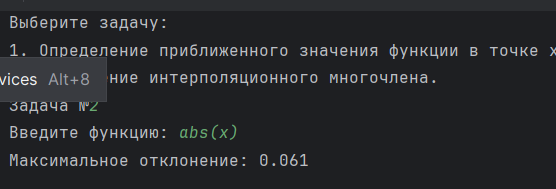
2.3. Введена функция |x| и 8 точек.



*Рис. 2.3.1. Значения аргумента.*

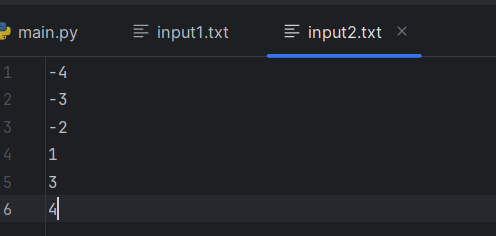
**

*Рис. 2.3.2.1. Результат вычислений.*

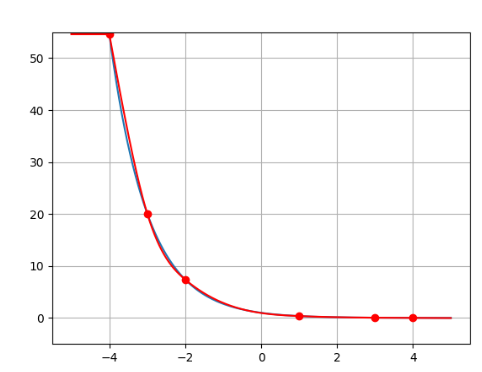


*Рис. 2.3.2.2. Результат вычислений.*

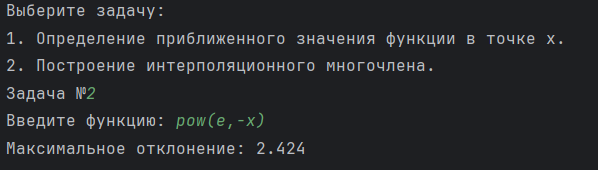
2.4. Введена функция e -x и 6 точек.



*Рис. 2.4.1. Значения аргумента.*

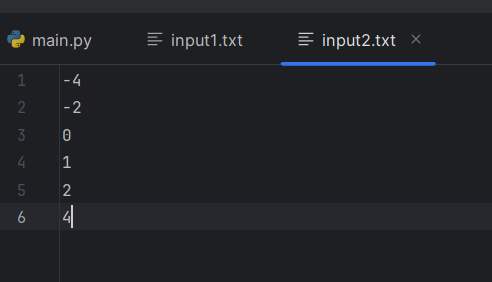
**

*Рис. 2.4.2.1. Результат вычислений.*

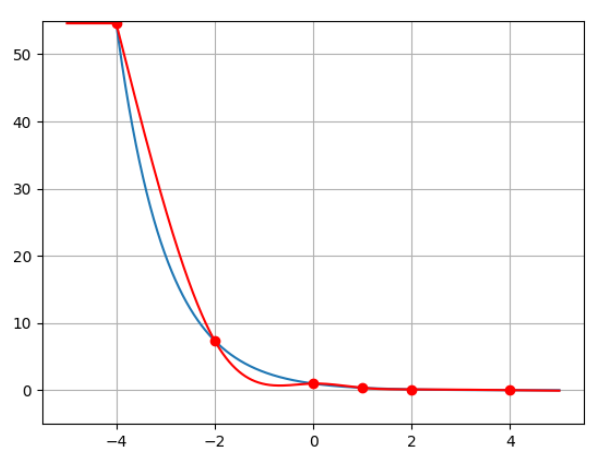


*Рис. 2.4.2.2. Результат вычислений.*

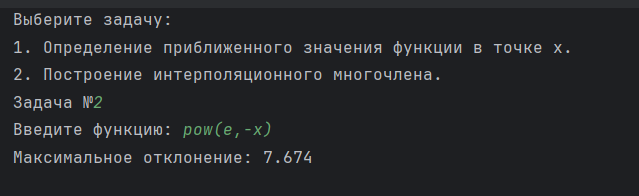
2.5. Введена функция e -x и 6 точек отличных от примера 2.5.



*Рис. 2.5.1. Значения аргумента.*

**

*Рис. 2.5.2.1. Результат вычислений.*



*Рис. 2.5.2.2. Результат вычислений.*

**Вывод:** После проведения исследований, стало очевидно, что количество и расположение узлов влияет на величину погрешности интерполирования. Исходя из тестов 2.1 и 2.2 при увеличении узлов интерполирования увеличивается точность, так же как из тестов 2.4 и 2.5 ясно, что от месторасположения точек, меняется точность.