Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Методы численного анализа

**ОТЧЁТ**

к лабораторной работе

на тему

Интерполяционные многочлены

Выполнил: студент группы 253504

Фроленко Кирилл Юрьевич

Проверил: Анисимов Владимир Яковлевич

Минск 2023

Оглавление

[**Цели выполнения задания** 3](#_30j0zll)

[**Краткие теоретические сведения** 4](https://docs.google.com/document/d/1EWdc2b-UMM5lmh1NDBJQCQ1GzPigp2mZ4cvigqo9fAY/edit#heading=h.1fob9te)

[**Задание**](https://docs.google.com/document/d/1EWdc2b-UMM5lmh1NDBJQCQ1GzPigp2mZ4cvigqo9fAY/edit#heading=h.3znysh7) 8

**Алгоритм решения**8

[**Программная реализация**](https://docs.google.com/document/d/1EWdc2b-UMM5lmh1NDBJQCQ1GzPigp2mZ4cvigqo9fAY/edit#heading=h.2et92p0) 9

[**Полученные результаты**](https://docs.google.com/document/d/1EWdc2b-UMM5lmh1NDBJQCQ1GzPigp2mZ4cvigqo9fAY/edit#heading=h.tyjcwt) 12

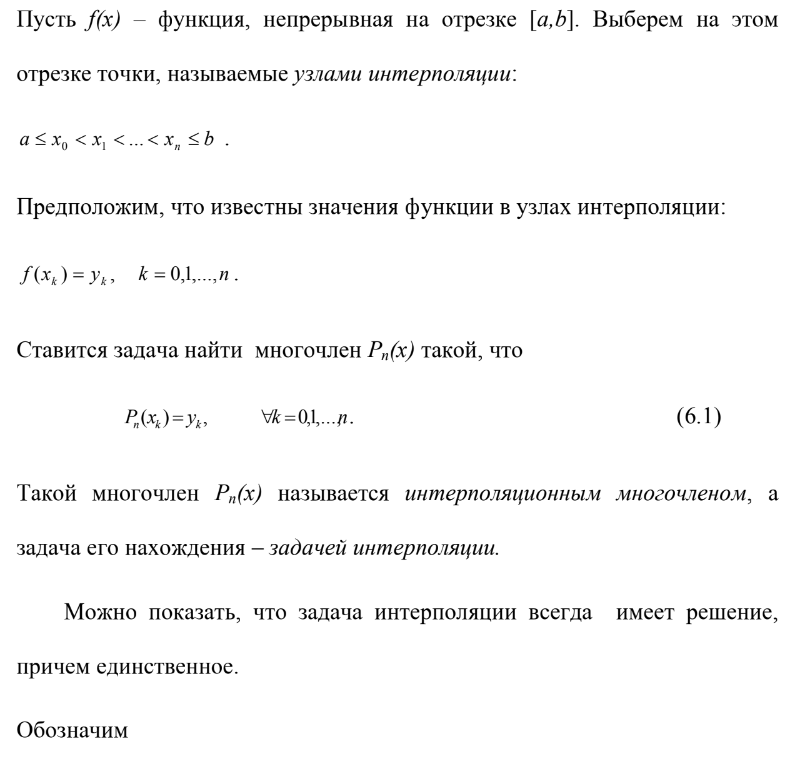
**Тестовые примеры** 13

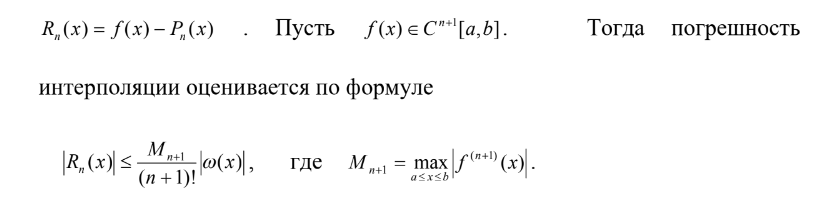
[**Выводы**](https://docs.google.com/document/d/1EWdc2b-UMM5lmh1NDBJQCQ1GzPigp2mZ4cvigqo9fAY/edit#heading=h.1t3h5sf) 20

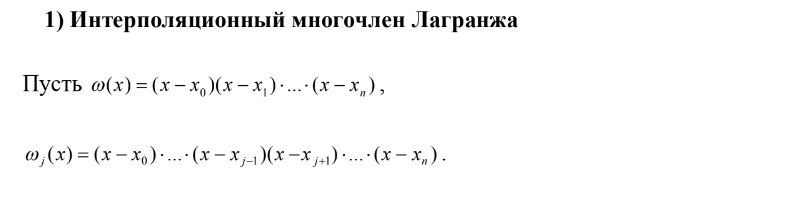
# **Цели выполнения задания**

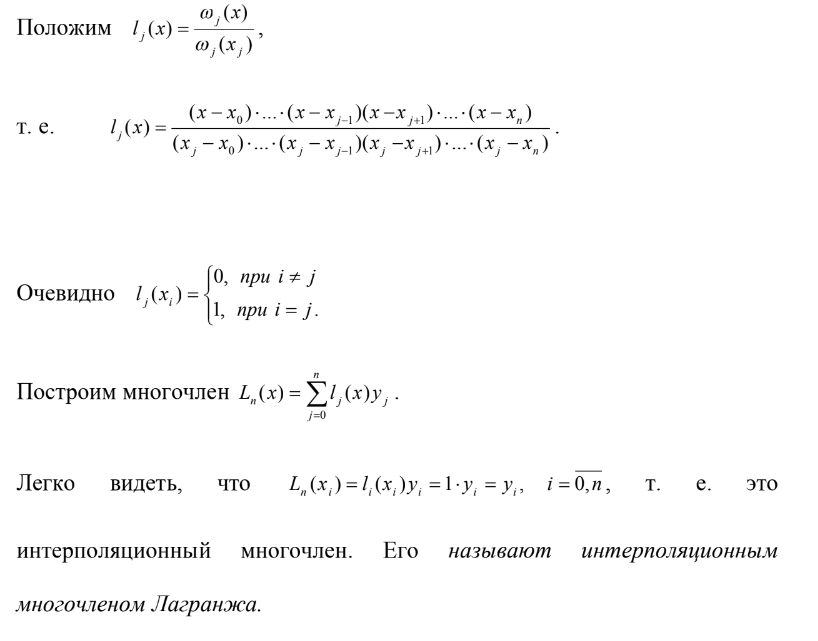
Изучить интерполяцию функций с помощью интерполяционных многочленов Лагранжа и Ньютона.

# **Краткие теоретические сведения**

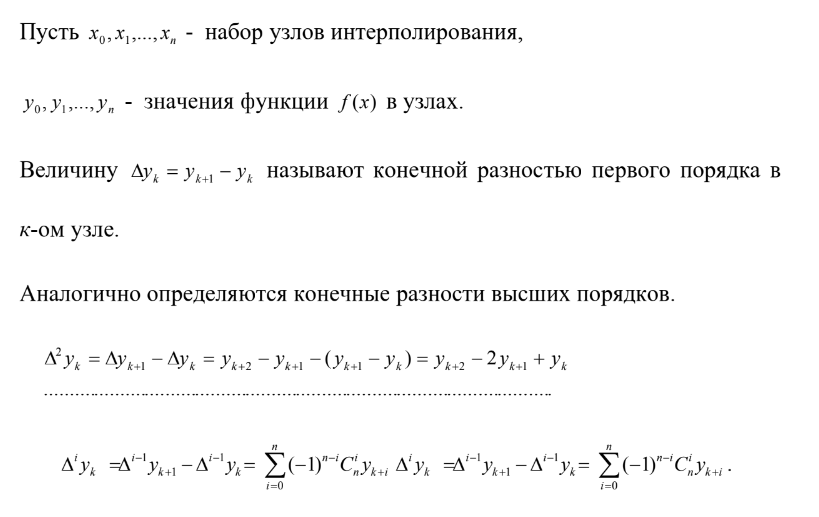


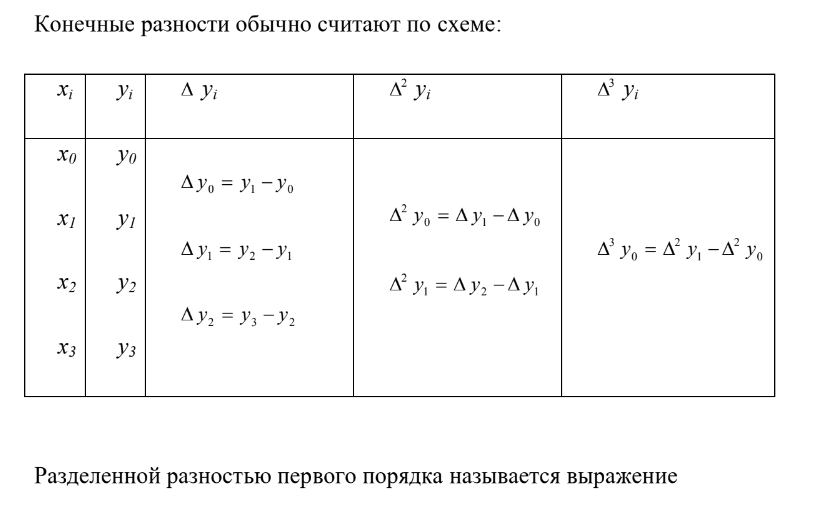




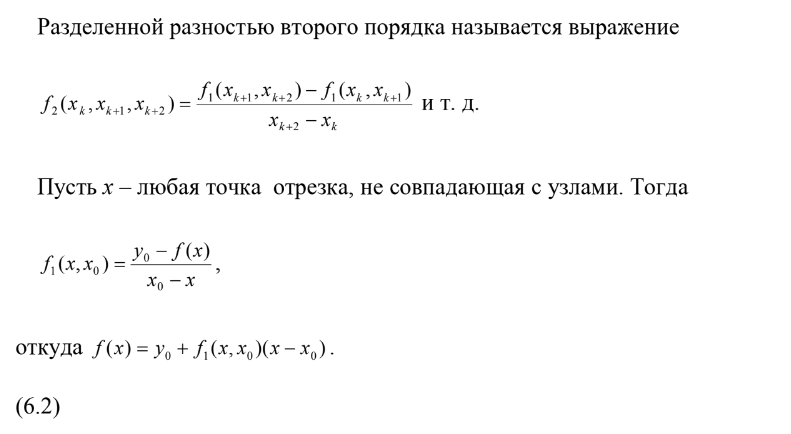


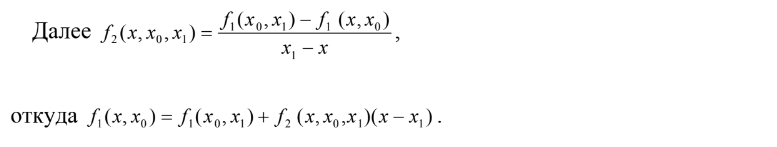


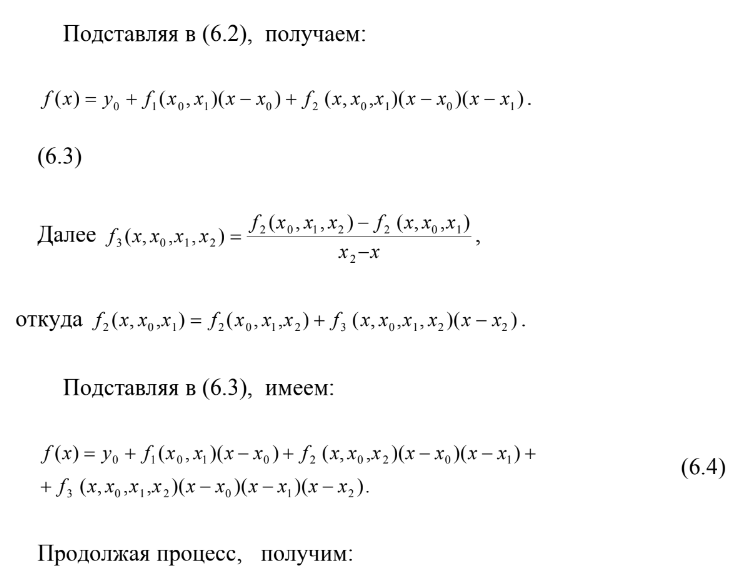




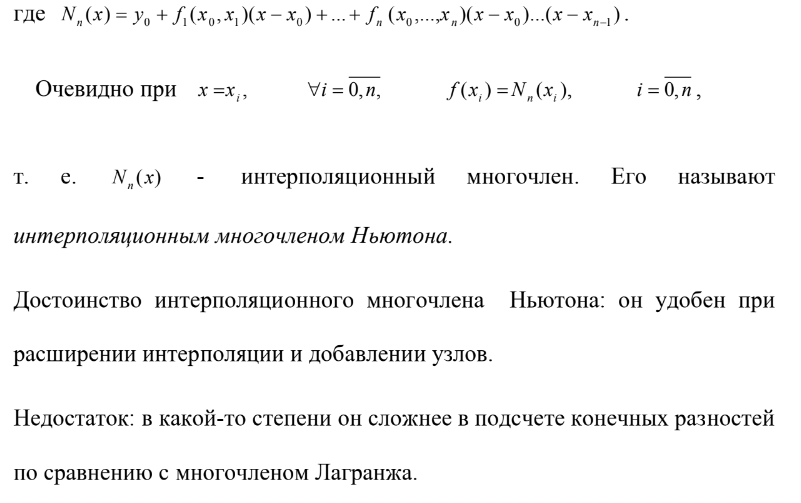






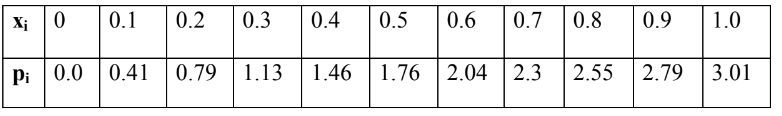




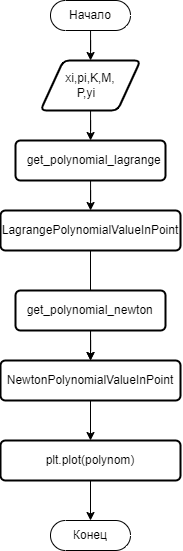


# **Задание**

Вариант 28, k = 14, m = 1.96



**Алгоритм решения**



**Программная реализация**

**Lagrange.py**

import numpy as np

from task import xi, yi

def get\_coefficients(\_pl: int, \_xi: np.ndarray):

n = int(\_xi.shape[0])

coefficients = np.empty((n, 2), dtype=float)

for i in range(n):

if i == \_pl:

coefficients[i][0] = float('inf')

coefficients[i][1] = float('inf')

else:

coefficients[i][0] = 1 / (\_xi[\_pl] - \_xi[i])

coefficients[i][1] = -\_xi[i] / (\_xi[\_pl] - \_xi[i])

filtered\_coefficients = np.empty((n - 1, 2), dtype=float)

j = 0

for i in range(n):

if coefficients[i][0] != float('inf'):

# изменение последовательности, степень увеличивается

filtered\_coefficients[j][0] = coefficients[i][1]

filtered\_coefficients[j][1] = coefficients[i][0]

j += 1

return filtered\_coefficients

def get\_polynomial\_l(\_xi: np.ndarray):

n = int(\_xi.shape[0])

pli = np.zeros((n, n), dtype=float)

for pl in range(n):

coefficients = get\_coefficients(pl, \_xi)

for i in range(1, n - 1): # проходим по массиву coefficients

if i == 1: # на второй итерации занимаются 0-2 степени

pli[pl][0] = coefficients[i - 1][0] \* coefficients[i][0]

pli[pl][1] = coefficients[i - 1][1] \* coefficients[i][0] + coefficients[i][1] \* coefficients[i - 1][0]

pli[pl][2] = coefficients[i - 1][1] \* coefficients[i][1]

else:

clone\_pli = np.zeros(n, dtype=float)

for val in range(n):

clone\_pli[val] = pli[pl][val]

zeros\_pli = np.zeros(n, dtype=float)

for j in range(n-1): # проходим по строке pl массива pli

product\_1 = clone\_pli[j] \* coefficients[i][0]

product\_2 = clone\_pli[j] \* coefficients[i][1]

zeros\_pli[j] += product\_1

zeros\_pli[j+1] += product\_2

for val in range(n):

pli[pl][val] = zeros\_pli[val]

return pli

def get\_polynomial\_lagrange(\_xi: np.ndarray, \_yi: np.ndarray):

n = int(\_xi.shape[0])

polynomial\_l = get\_polynomial\_l(\_xi)

for i in range(n):

for j in range(n):

polynomial\_l[i][j] \*= \_yi[i]

L = np.zeros(n, dtype=float)

for i in range(n):

for j in range(n):

L[i] += polynomial\_l[j][i]

for i in range(L.size):

if L[i] != 0:

if i != 1:

if i == 0:

buff = f"{L[i]}"

ind = i + 1

break

else:

buff = f"{L[i]}x^{i}"

ind = i + 1

break

else:

buff = f"{L[i]}x"

ind = i + 1

break

for i in range(ind, L.size - 1):

if L[i] != 0:

if L[i] > 0:

if i == 1:

buff += f" + {L[i]}x"

else:

buff += f" + {L[i]}x^{i}"

else:

if i == 1:

buff += f" - {abs(L[i + 1])}x"

else:

buff += f" - {abs(L[i + 1])}x^{i}"

return buff

def omega(x\_var, j):

return np.array([x\_var - x\_j for x\_j in xi if x\_j != xi[j]]).prod()

def l\_div\_seq(x\_i, j):

return omega(x\_i, j) / omega(xi[j], j)

def LagrangePolynomialValueInPoint(x\_var):

return sum([l\_div\_seq(x\_var, j) \* yi[j] for j in range(len(xi))])

**Newton.py**

import numpy as np

from task import xi, yi

def get\_coefficients(xi):

n = len(xi)

coefficients = np.zeros((n, n))

for i in range(n):

coefficients[i][0] = yi[i]

for j in range(1, n):

for i in range(j, n):

coefficients[i][j] = (coefficients[i][j - 1] - coefficients[i - 1][j - 1]) / (xi[i] - xi[i - j])

return coefficients

def NewtonPolynomialValueInPoint(coefficients, xi, x):

n = len(xi)

result = coefficients[0][0]

term = 1

for i in range(1, n):

term \*= (x - xi[i - 1])

result += coefficients[i][i] \* term

return result

def get\_polynomial\_newton(coefficients, xi):

n = len(xi)

polynomial = ""

for i in range(n):

term = str(coefficients[i][i])

if i > 0:

for j in range(i):

term += f"\*(x - {xi[j]})"

polynomial += term

if i < n - 1:

polynomial += " + "

return polynomial

**main.py**

from Lagrange import get\_polynomial\_lagrange, LagrangePolynomialValueInPoint

from Newton import get\_polynomial\_newton, get\_coefficients, NewtonPolynomialValueInPoint

from task import xi, yi, P

import matplotlib.pyplot as plt

import numpy as np

print("Интерполяционный многочлен Лагранжа:")

print(get\_polynomial\_lagrange(xi, yi), "\n")

print(f"Значение многочлена в точке {P}:")

print(LagrangePolynomialValueInPoint(P))

coefficients = get\_coefficients(xi)

print("\nИнтерполяционный многочлен Ньютона:")

print(get\_polynomial\_newton(coefficients, xi), "\n")

print(f"Значение многочлена в точке {P}:")

print((NewtonPolynomialValueInPoint(coefficients, xi, P)))

print("\nЗначение в точке, вычисленное с помощью пакета numpy:")

print(np.interp(P, xi, yi))

x\_values = np.linspace(100, 10000, 100)

lagrange\_y = [LagrangePolynomialValueInPoint(x\_val) for x\_val in x\_values]

newton\_y = [NewtonPolynomialValueInPoint(coefficients, xi, x\_val) for x\_val in x\_values]

plt.plot(x\_values, lagrange\_y, label='Polynom', color='blue')

plt.legend()

plt.xlabel('x')

plt.ylabel('y')

plt.title('Polynom')

plt.grid()

plt.show()

**Полученные результаты программы**

Интерполяционный многочлен Лагранжа:

1.96 + 0.628214285714014x + 92.74934523812252x^2 - 5268.126929014015x^3 + 5268.126929014015x^4 - 35322.1643518545x^5 + 35322.1643518545x^6 - 37136.24338624069x^7 + 37136.24338624069x^8 - 3279.3209876544806x^9

Значение многочлена в точке 0.47:

3.632652079708804

Интерполяционный многочлен Ньютона:

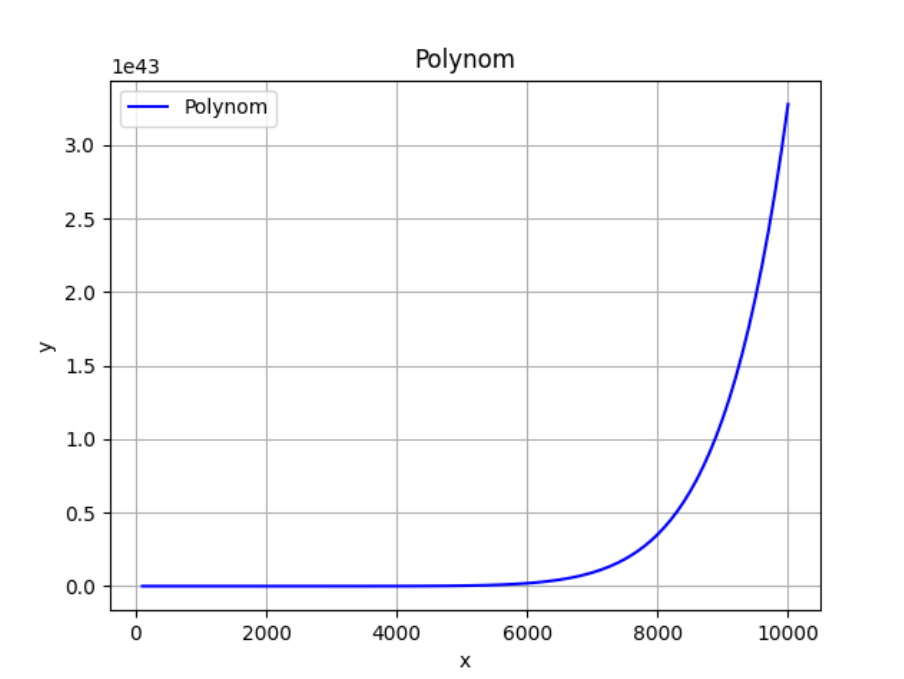
1.96 + 4.100000000000001\*(x - 0.0) + -1.5000000000000124\*(x - 0.0)\*(x - 0.1) + -1.6666666666666172\*(x - 0.0)\*(x - 0.1)\*(x - 0.2) + 16.666666666666536\*(x - 0.0)\*(x - 0.1)\*(x - 0.2)\*(x - 0.3) + -74.99999999999986\*(x - 0.0)\*(x - 0.1)\*(x - 0.2)\*(x - 0.3)\*(x - 0.4) + 236.11111111111205\*(x - 0.0)\*(x - 0.1)\*(x - 0.2)\*(x - 0.3)\*(x - 0.4)\*(x - 0.5) + -575.3968253968327\*(x - 0.0)\*(x - 0.1)\*(x - 0.2)\*(x - 0.3)\*(x - 0.4)\*(x - 0.5)\*(x - 0.6) + 1165.6746031746316\*(x - 0.0)\*(x - 0.1)\*(x - 0.2)\*(x - 0.3)\*(x - 0.4)\*(x - 0.5)\*(x - 0.6)\*(x - 0.7) + -2066.798941799019\*(x - 0.0)\*(x - 0.1)\*(x - 0.2)\*(x - 0.3)\*(x - 0.4)\*(x - 0.5)\*(x - 0.6)\*(x - 0.7)\*(x - 0.8) + 3279.3209876544774\*(x - 0.0)\*(x - 0.1)\*(x - 0.2)\*(x - 0.3)\*(x - 0.4)\*(x - 0.5)\*(x - 0.6)\*(x - 0.7)\*(x - 0.8)\*(x - 0.9)

Значение многочлена в точке 0.47:

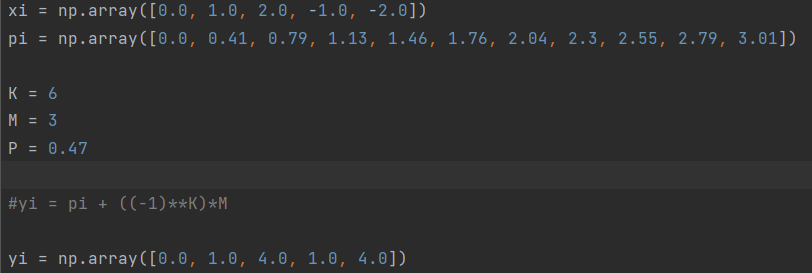
3.6326520797088033

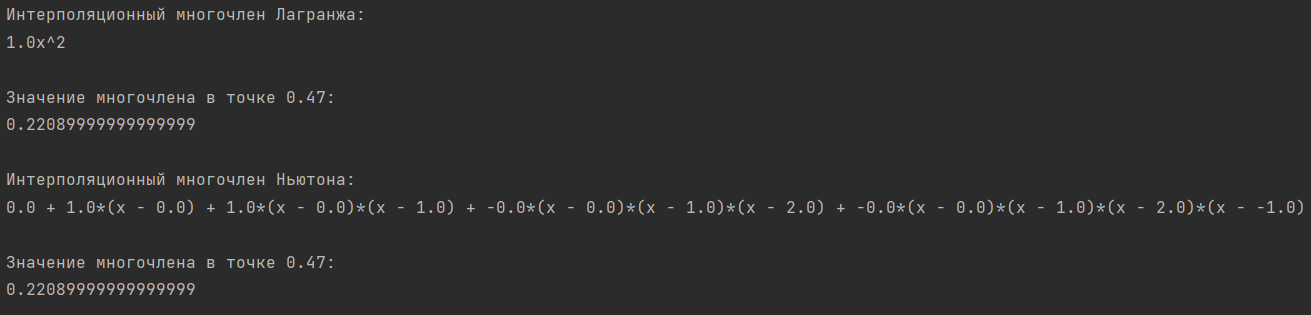
Значение многочлена в точке 0.47(с помощью numpy:

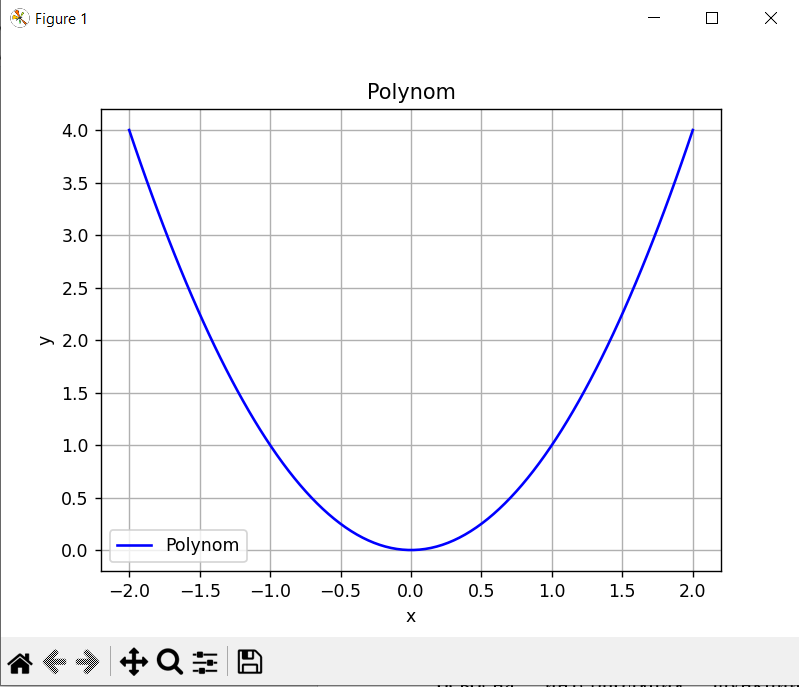
3.63

**Тестовые примеры**

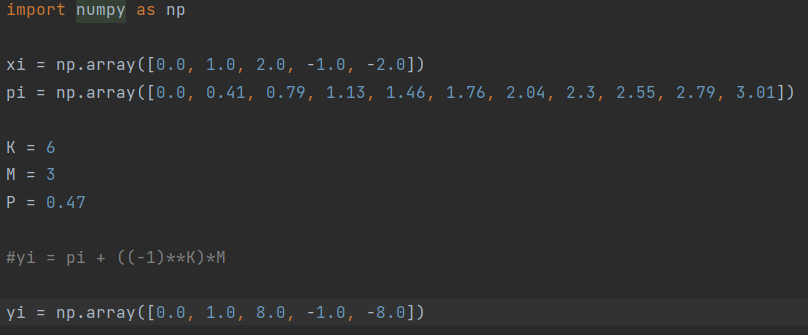
**Тестовый пример 1**

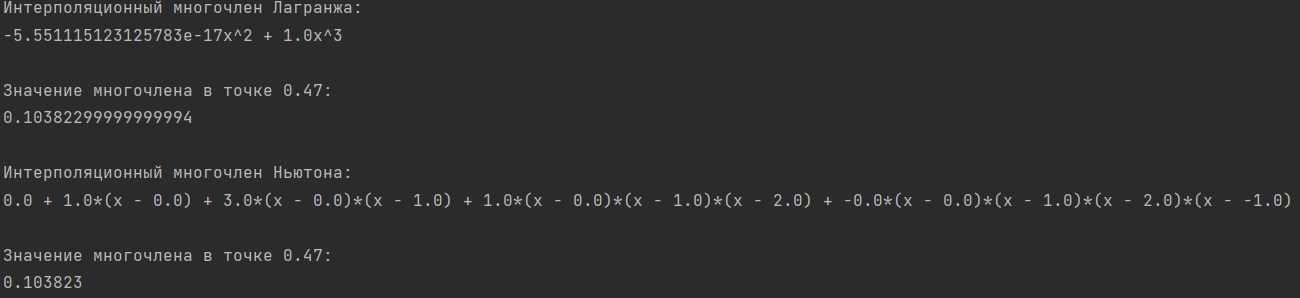
****

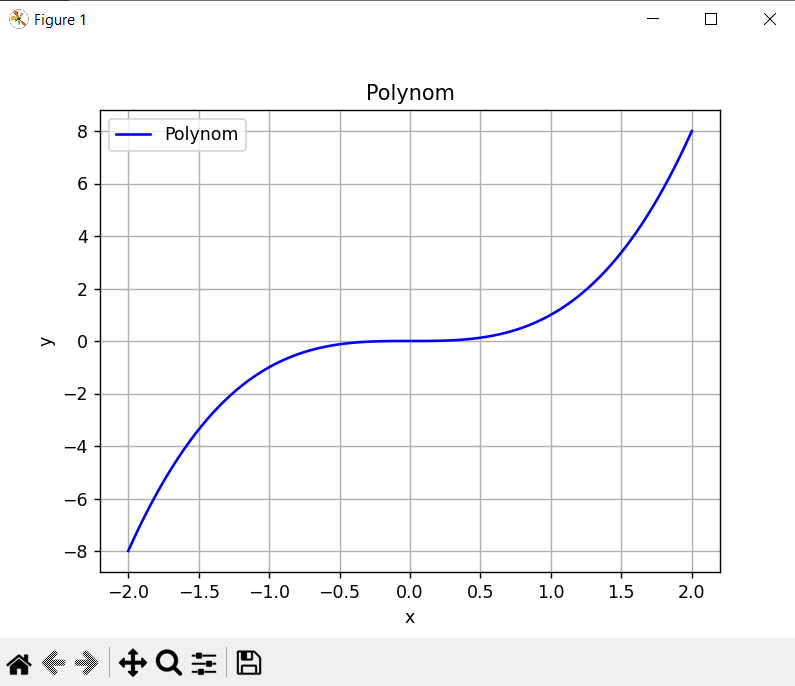
****



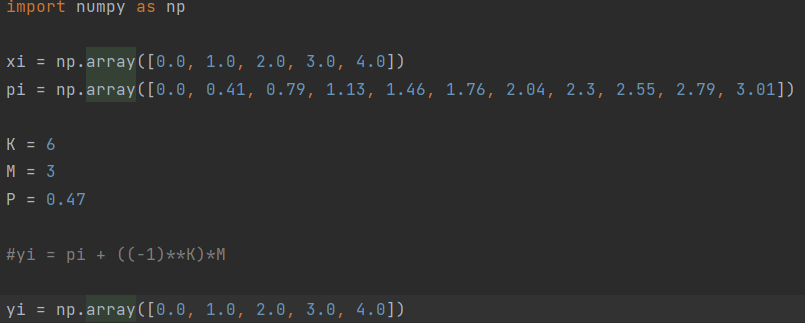
**Тестовый пример 2**

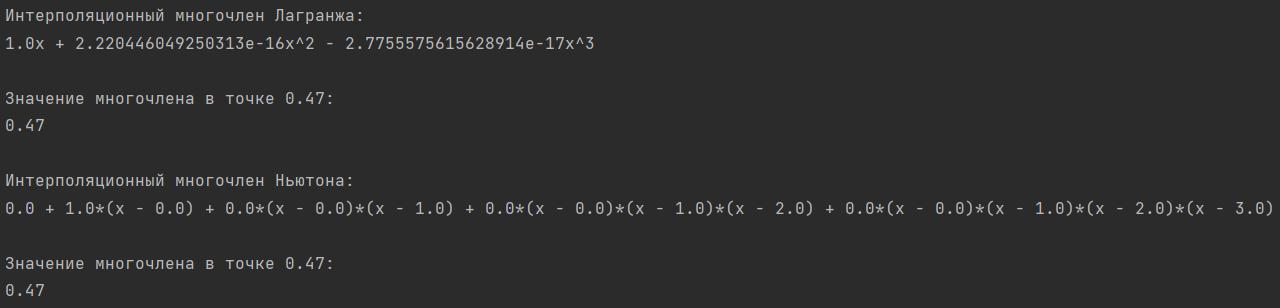
****

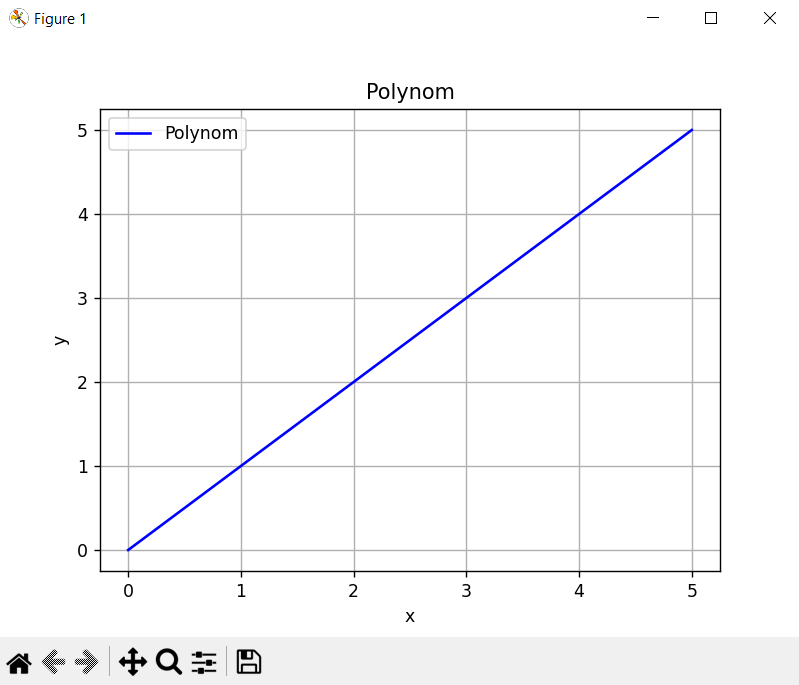




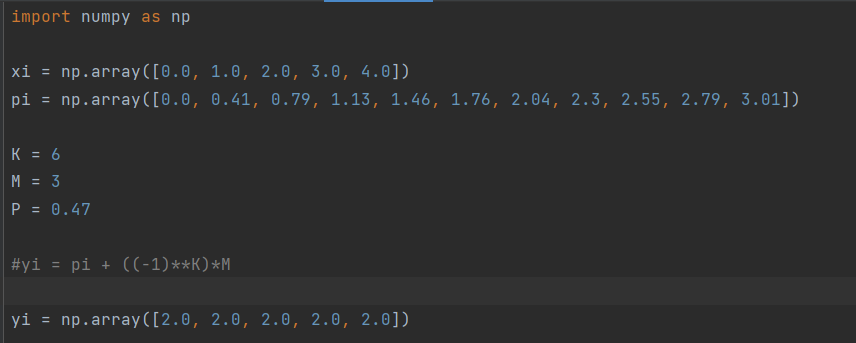
**Тестовый пример 3**

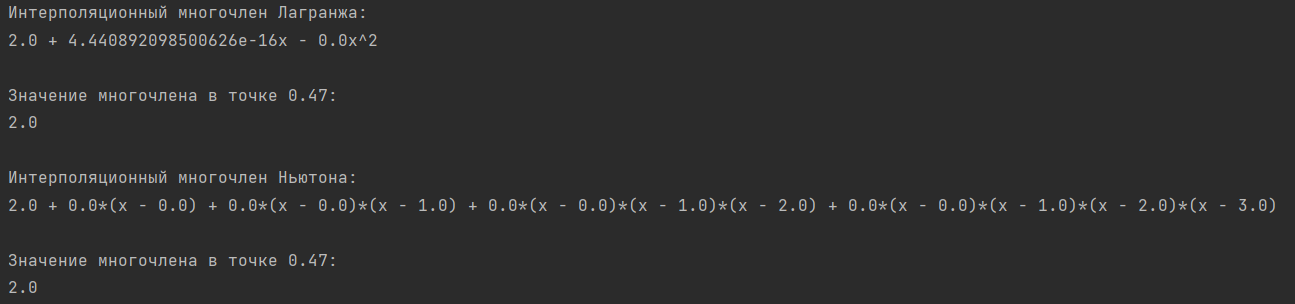
****

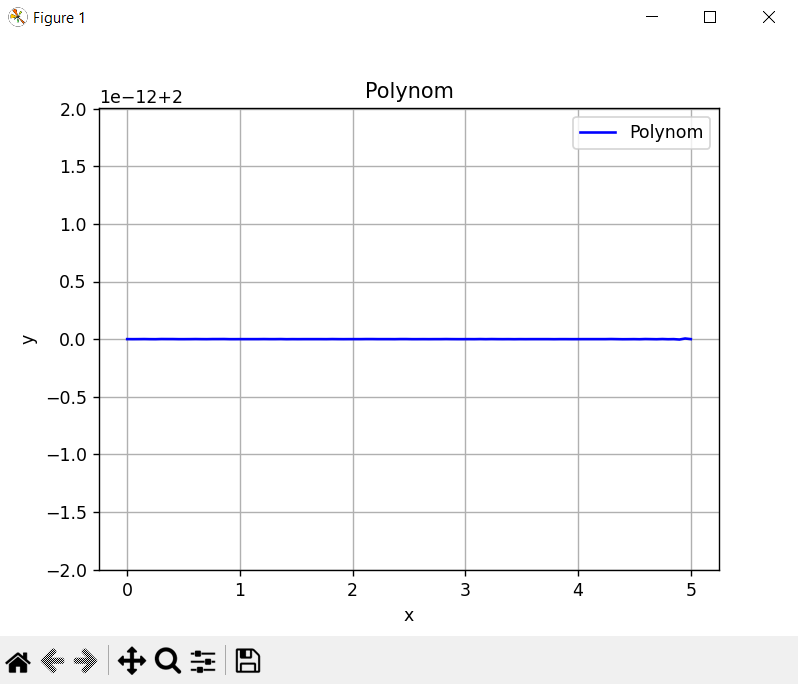
****

****

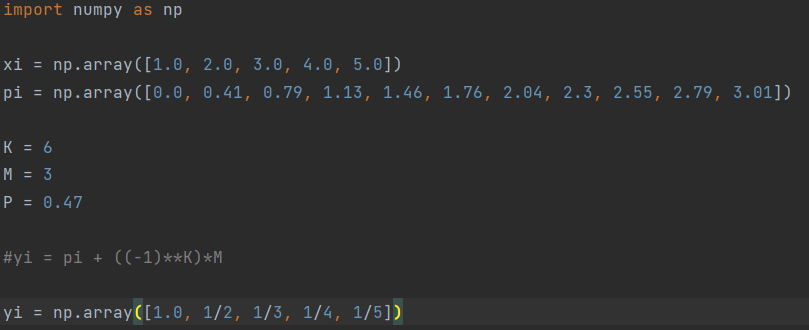
**Тестовый пример 4**

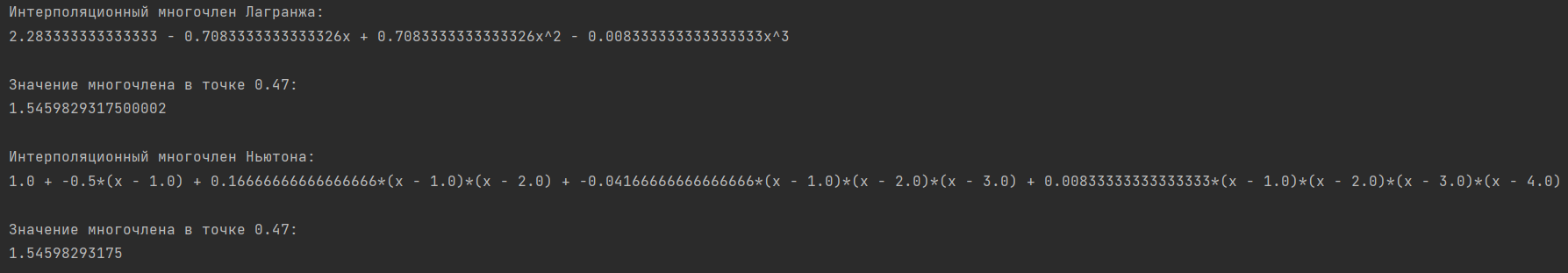
****

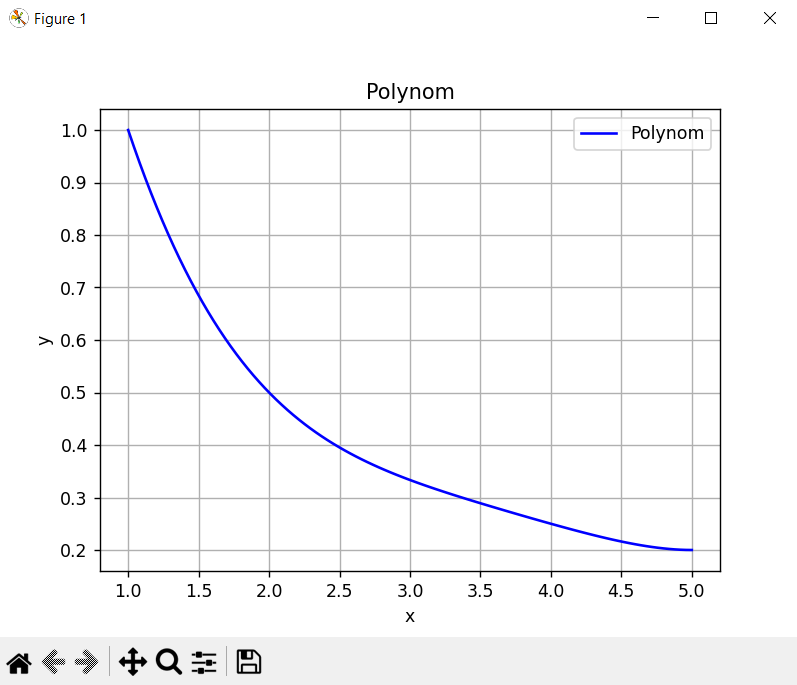
****

****

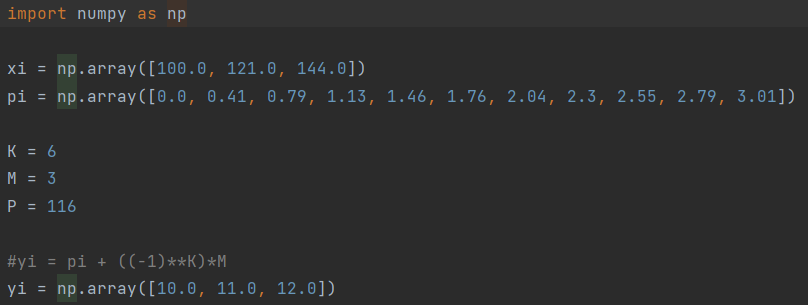
**Тестовый пример 5**

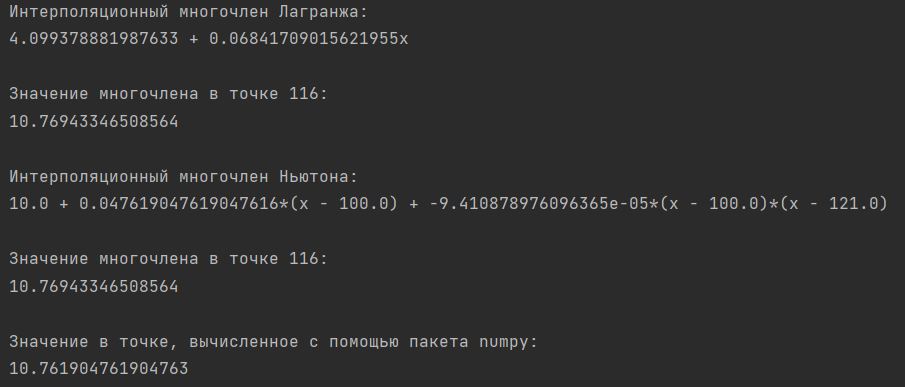
****

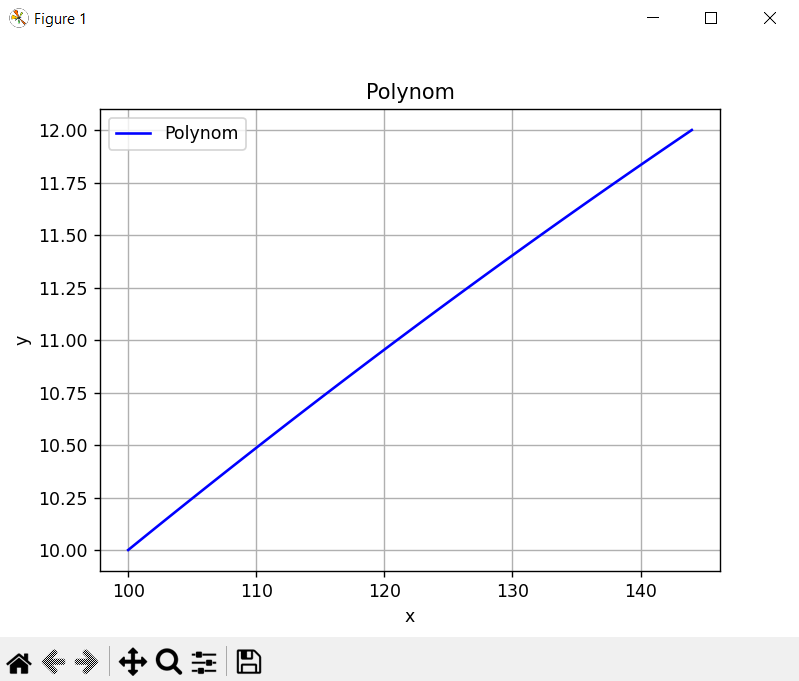
****

****

**Тестовый пример 6**

****

****

****

**Оценка погрешности** Оценим погрешность на примере функции sqrt(x) на узлах x0 = 100, x1 = 121, x2 = 144 в точке x = 116 и на всем отрезке [100,144] с помощью интерполяционного многочлена Лагранжа второй(первой степени) степени

Имеем: , , .

.

Погрешность в точке  составит:

.

А максимальная погрешность на всем отрезке :



Для многочлена первой степени имеем:

**Выводы**

Таким образом, в ходе выполнения лабораторной работы была освоена интерполяция функций с помощью интерполяционных многочленов Лагранжа и Ньютона. Составлена компьютерная программа, на тестовых примерах проверена правильность её работы, построены интерполяционные многочлены в форме Лагранжа и Ньютона, вычислено значение функции в точке согласно заданному варианту.