**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ «МИСиС»**

ИНСТИТУТ Информационных технологий и автоматизированных систем управления

КАФЕДРА Инженерной кибернетики

**ЛАБОРАТОРНАЯ РАБОТА №4**

**по дисциплине «ЧИСЛЕННЫЕ МЕТОДЫ»**

**Интерполяционные многочлены Лагранжа и Ньютона.**

Вариант 10

Кущ А. А.

Группа БПМ-18-1

Москва 2020

Оглавление

[Теоретическая часть 3](#_Toc55069350)

[Цель: 3](#_Toc55069351)

[Многочлен Лагранжа 3](#_Toc55069352)

[Многочлен Ньютона 3](#_Toc55069353)

[Практическая часть: 4](#_Toc55069354)

# Теоретическая часть

### Цель:

Построить интерполяционные многочлены Лагранжа и Ньютона.

Найти их значения в точках -1.3 и 1.75.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x | -2 | -1 | 0 | 1 | 2 | 3 | 4 |
| f(x) | 1.23 | -1.66 | -3.25 | -3.11 | 6.66 | 7.8 | -0.04 |

### Многочлен Лагранжа

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| f(x) | X1 | X2 | X3 | X4 | X5 | X6 | X7 | multi |
| 1.23 | x + 2 | -1 | -2 | -3 | -4 | -5 | -6 | (x+2)\*720 |
| -1.66 | 1 | x + 1 | -1 | -2 | -3 | -4 | -5 | (x+1)\*(-120) |
| -3.25 | 2 | 1 | x | -1 | -2 | -3 | -4 | x \* 48 |
| -3.11 | 3 | 2 | 1 | x - 1 | -1 | -2 | -3 | (x-1)\*(-36) |
| -6.66 | 4 | 3 | 2 | 1 | x - 2 | -1 | -2 | (x-2)\*48 |
| 7.8 | 5 | 4 | 3 | 2 | 1 | x - 3 | -1 | (x-3)\*(-120) |
| -0.04 | 6 | 5 | 4 | 3 | 2 | 1 | x - 4 | (x-4)\*720 |
|  |  | | | | | | |  |

Формула многочлена Лагранжа:

### Многочлен Ньютона

# Практическая часть:

import numpy as np

y = [1.23,-1.66,-3.25,-3.11,6.66,7.8,-0.04]

def sum\_(y, x):

x\_ = [x - i for i in range(-2, 5)]

k = [720, -120, 48, -36, 48, -120, 720]

return sum([y[i]/(x\_[i] \* k[i]) for i in range(len(k))])

def L(y, x):

return (x+2)\*(x+1)\*x\*(x-1)\*(x-2)\*(x-3)\*(x-4)\*sum\_(y, x)

print('Lagrange | Method 1: f(-1.3)={:.4f}, f(1.75)={:.4f}'.format(L(y, -1.3), L(y, 1.75)))

def newton(x):

newton\_array = np.zeros((7, 7), dtype=np.float16)

newton\_array[:, 0] = y

# print(newton\_array)

for col in range(1, 7):

for line in range(1, 7):

if not newton\_array[line, col-1]:

break

newton\_array[line-1, col] = (newton\_array[line, col-1] - newton\_array[line-1, col-1]) / col

# print(newton\_array)

x\_ = [x - i for i in range(-2, 5)]

res = newton\_array[0, 0]

for col in range(1, 7):

tmp = 1

for i in range(col):

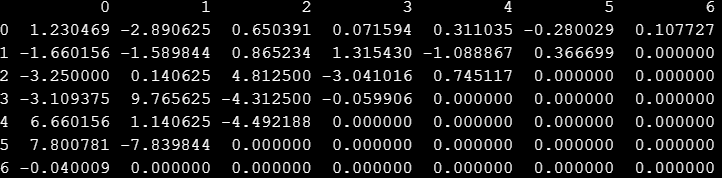
tmp \*= x\_[i]

res += tmp \* newton\_array[0, col]

return res

print('Newton | Method 2: f(-1.3)={:.4f}, f(1.75)={:.4f}'.format(newton(-1.3), newton(1.75)))

Матрица Ньютона:



Итоговый вывод скрипта:

Lagrange | Method 1: f(-1.3)=-2.6476, f(1.75)=4.0066

Newton | Method 2: f(-1.3)=-2.6454, f(1.75)=4.0030