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 053506

Слуцкий Никита Сергеевич

Проверил: Анисимов Владимир Яковлевич

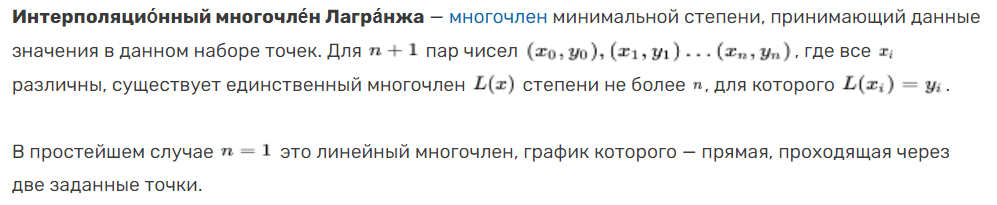
Минск 2022

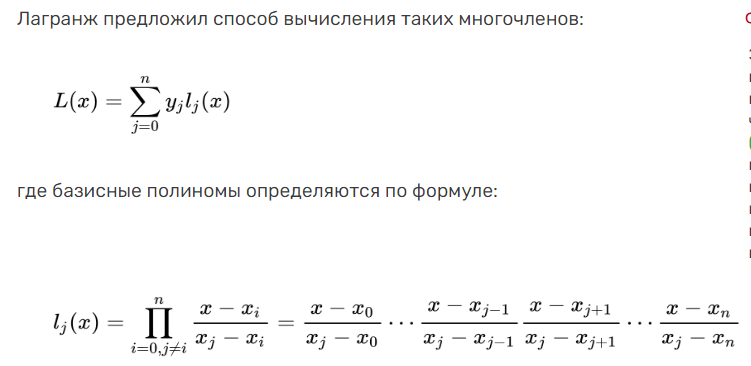
**Вариант 7 (Номер в журнале – 21)**

**Цели выполнения задания:**

Изучить интерполяцию функций с помощью интерполяционных многочленов.

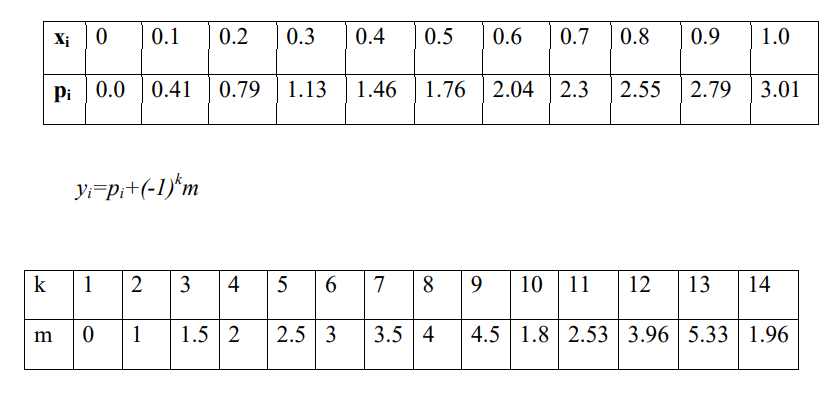
**Краткие теоретические сведения:**





**Задание**

Построить интерполяционные многочлены в форме Лагранжа и Ньютона, используя номер варианта и данные:



**Программная реализация:**

Ниже можно ознакомиться с программной реализацией главных функций по триангуляции, обратному ходу, а также вызов этого всего из главного файла. Вспомогательные функции и перегруженные операторы (реализации) опущены. Реализация на языке программирования Python с использованием библиотеки NumPy.

import sympy  
from sympy**.**abc import x**,** y  
  
  
def get\_basis\_polynom**(**current: int**,** x\_array: list**[**float**])** -> sympy**.**poly:  
 response = sympy**.**poly**(**1**,** x**)** for counter in range**(**len**(**x\_array**))**:  
 if counter != current:  
 response \*= sympy**.**poly**((**x - x\_array**[**counter**])** / **(**x\_array**[**current**]** - x\_array**[**counter**]))** return response  
  
  
def get\_lagrange\_polynom**(**x\_array: list**[**float**],** y\_array: list**[**float**])** -> sympy**.**poly:  
 print**(**sympy**.**poly**(**x**))** response = sympy**.**poly**(**0**,** x**)** for counter in range**(**len**(**x\_array**))**:  
 response += y\_array**[**counter**]** \* get\_basis\_polynom**(**counter**,** x\_array**)** return response

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

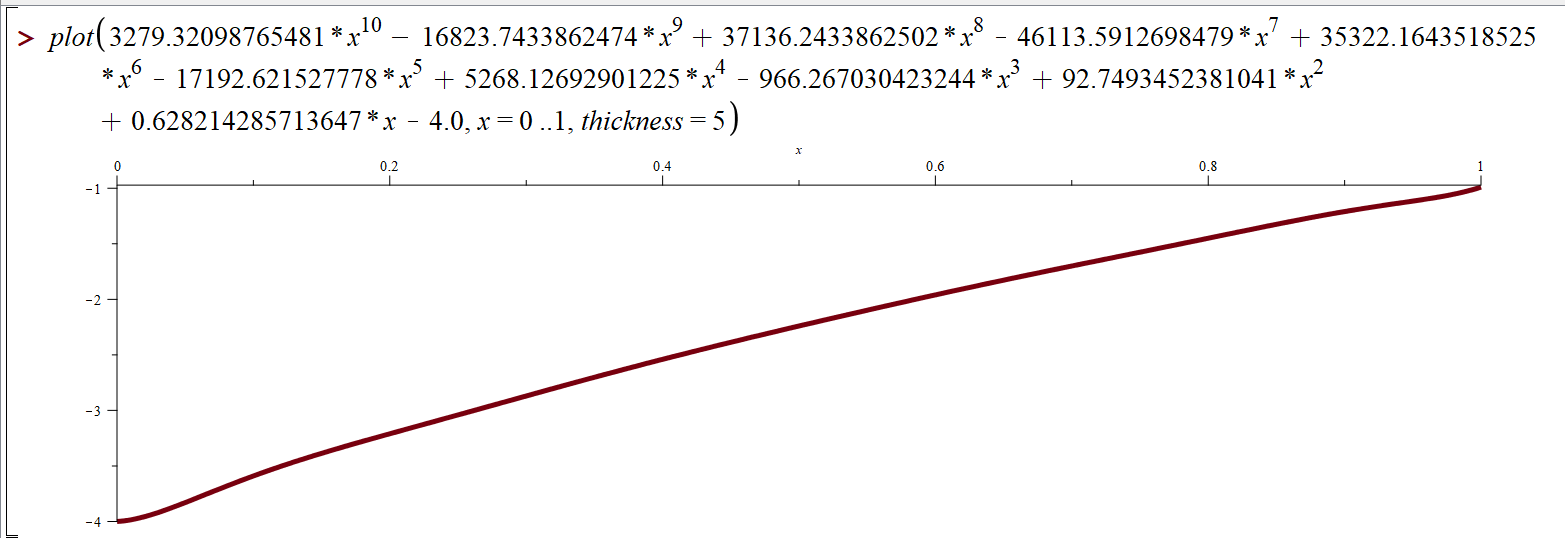
from data import get\_y**,** OPTION**,** X  
from lagrange\_interpolating\_polynom import get\_lagrange\_polynom  
  
  
def main**()** -> None:  
 y\_array: list = get\_y**(**OPTION**)** x\_array: list = list**(**X**)** print**(**y\_array**)** print**(**get\_lagrange\_polynom**(**x\_array**,** y\_array**))**if \_\_name\_\_ == '\_\_main\_\_':  
 main**()**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

OPTION: int = 7  
  
M: list**[**float**]** = **[**0**,** 1**,** 1.5**,** 2**,** 2.5**,** 3**,** 3.5**,** 4**,** 4.5**,** 1.8**,** 2.53**,** 3.96**,** 5.33**,** 1.96**]**X: list**[**float**]** = **[**0.0**,** 0.1**,** 0.2**,** 0.3**,** 0.4**,** 0.5**,** 0.6**,** 0.7**,** 0.8**,** 0.9**,** 1.0**]**P: list**[**float**]** = **[**0.0**,** 0.41**,** 0.79**,** 1.13**,** 1.46**,** 1.76**,** 2.04**,** 2.3**,** 2.55**,** 2.79**,** 3.01**]**def get\_y**(**option: int**)** -> list**[**float**]**:  
 response: list = list**()** for counter in range**(**len**(**P**))**:  
 response**.**append**(**P**[**counter**]** + **((**-1**)** \*\* option**)** \* M**[**option**])** return response

**Полученные результаты:**

Для уравнения из варианта 7 был найден полином Лагранжа и Ньютона. В силу того, что при заданном множестве точек, он единственный, то при раскрытии скобок (а только так пакет Python SymPY позволяет получить ответ при перемножении многочленов) получается один и тот же результат. Проверяю в Maple правильность ( и по таблице с точками)



# **Тестовые задания**

Были решены уравнения с другими коэффициентами из всех вариантов.

**Выводы**

Полагаюсь на библиотеку SymPy. Например, в Лагранже нужно много перемножать многочлены (хоть и первой степени), складывать их, конкретика зависит от реализации в SymPy той или иной операции. Точность не измерял. Многочлен наилучшего приближения не строил по причине сложности в реализации.