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:

Интерполяцио́нный многочле́н Лагра́нжа — многочлен минимальной степени, принимающий данные значения в данном наборе точек. Для $n+1$ пар чисел $(x_0, y_0), (x_1, y_1) \dots (x_n, y_n)$, где все x_i различны, существует единственный многочлен $L(x)$ степени не более n, для которого $L(x_i) = y_i$.

В простейшем случае $n = 1$ это линейный многочлен, график которого — прямая, проходящая через две заданные точки.

Лагранж предложил способ вычисления таких многочленов:

$$L(x) = \sum_{j=0}^{n} y_j l_j(x)$$

где базисные полиномы определяются по формуле:

$$l_j(x) = \prod_{i=0, j\neq i}^{n} \frac{x - x_i}{x_j - x_i} = \frac{x - x_0}{x_j - x_0} \cdots \frac{x - x_{j-1}}{x_j - x_{j-1}} \frac{x - x_{j+1}}{x_j - x_{j+1}} \cdots \frac{x - x_n}{x_j - x_n}$$

Задание

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

x_i	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
p_i	0.0	0.41	0.79	1.13	1.46	1.76	2.04	2.3	2.55	2.79	3.01

$$y_i = p_i + (-1)^k m$$

k	1	2	3	4	5	6	7	8	9	10	11	12	13	14
m	0	1	1.5	2	2.5	3	3.5	4	4.5	1.8	2.53	3.96	5.33	1.96

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.

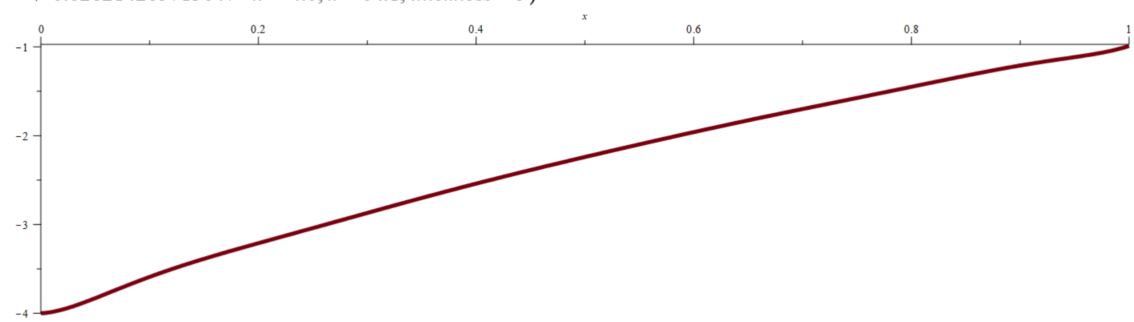
```
import sympy
from sympy.abc import x, y


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)

> $plot(3279.32098765481*x^{10} - 16823.7433862474*x^9 + 37136.2433862502*x^8 - 46113.5912698479*x^7 + 35322.1643518525*x^6 - 17192.621527778*x^5 + 5268.12692901225*x^4 - 966.267030423244*x^3 + 92.7493452381041*x^2 + 0.628214285713647*x - 4.0, x = 0..1, thickness = 5)$



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.