Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ “Информатика и системы управления”

КАФЕДРА "Программное обеспечение ЭВМ и информационные технологии”

ОТЧЁТ

К ЛАБОРАТОРНОЙ РАБОТЕ №1

ПО КУРСУ МОДЕЛИРОВАНИЕ НА ТЕМУ:

“Построение и программная реализация алгоритма полиномиальной интерполяции табличных функций”

Студент ИУ7-53 БВ _____ Д.П. Косаревский
(Группа) (Подпись, дата) (И.О.Фамилия)

Преподаватель _____ В.М. Градов
(Подпись, дата) (И.О.Фамилия)

Москва 2021 г.

Цель работы: Получение навыков построения алгоритма интерполяции таблично заданных функций полиномом Ньютона.

Исходные данные:

1. Таблица функции с количеством узлов N.

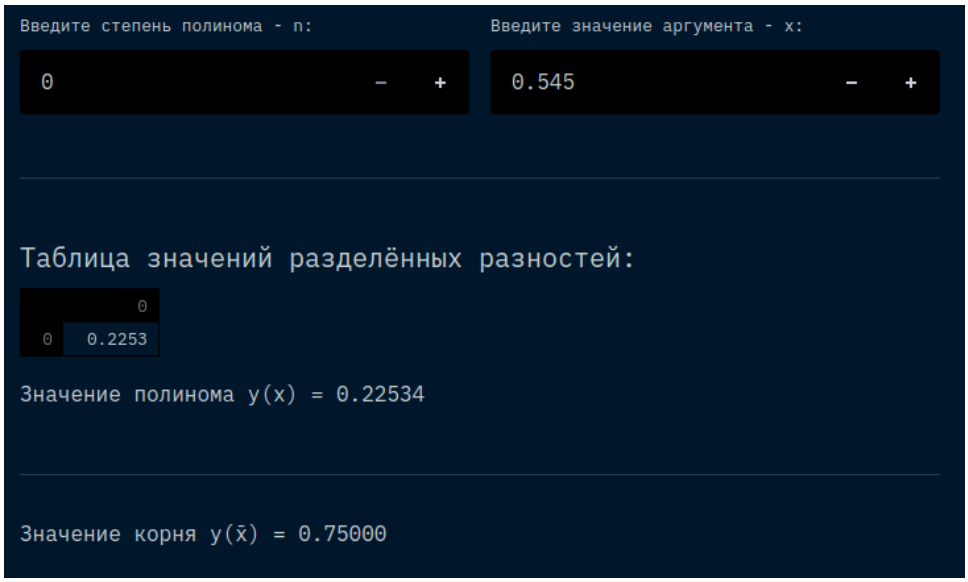
x	**y**
0	1.000000
0.15	0.838771
0.3	0.655336
0.45	0.450447
0.6	0.225336
0.75	-0.01831
0.9	-0.27839
1.05	-0.55243

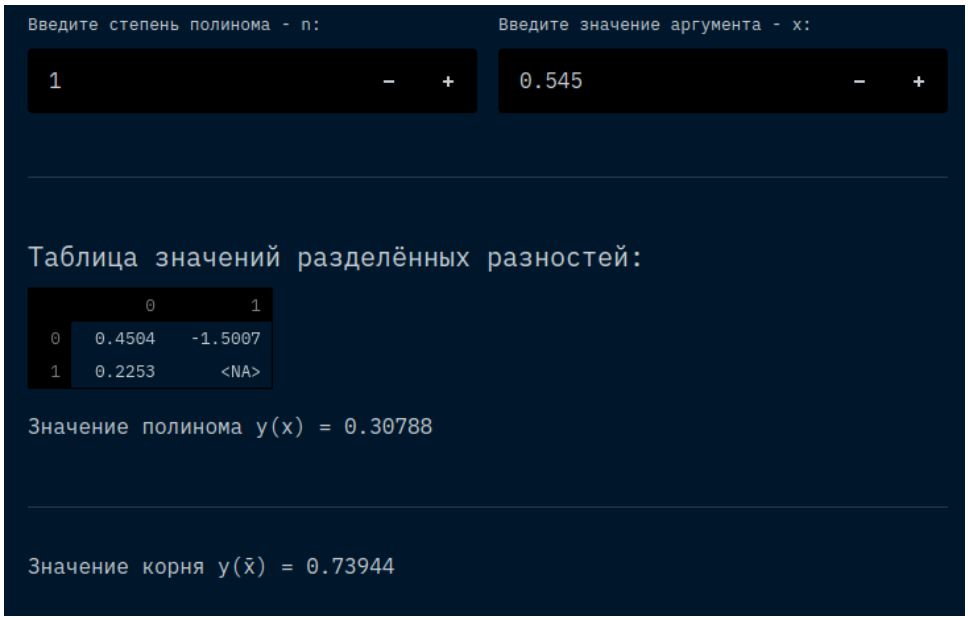
2. Степень аппроксимирующего полинома - n.
3. Значение аргумента, для которого выполняется интерполяция.

Результаты

В результате работы была достигнута поставленная цель и получены следующие результаты:

1. Значения y(x) при степенях полинома n=0, 1, 2, 3 и 4 при фиксированном x.
2. Найти корень заданной выше табличной функции с помощью обратной интерполяци.





Введите степень полинома - n:

2 – +

Введите значение аргумента - x:

0.545 – +

Таблица значений разделённых разностей:

	0	1	2
0	0.4504	-1.5007	-0.4119
1	0.2253	-1.6243	<NA>
2	-0.0183	<NA>	<NA>

Значение полинома y(x) = 0.31003

Значение корня y(x̄) = 0.73905

Введите степень полинома - n:

3 – +

Введите значение аргумента - x:

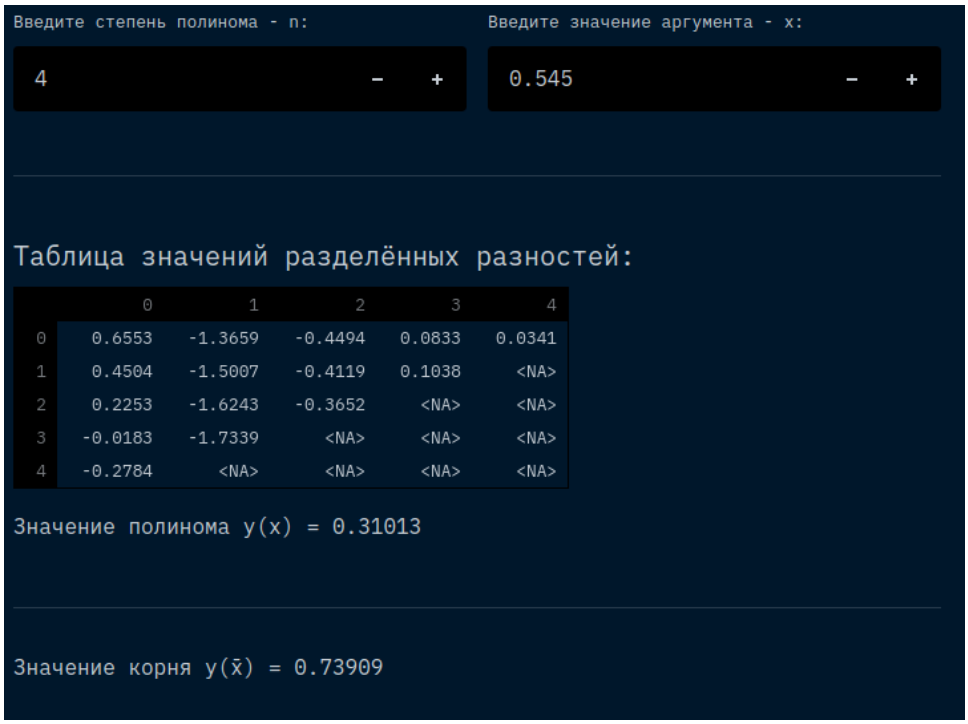
0.545 – +

Таблица значений разделённых разностей:

	0	1	2	3
0	0.6553	-1.3659	-0.4494	0.0833
1	0.4504	-1.5007	-0.4119	<NA>
2	0.2253	-1.6243	<NA>	<NA>
3	-0.0183	<NA>	<NA>	<NA>

Значение полинома y(x) = 0.31012

Значение корня y(x̄) = 0.73908



Работающую программу можно увидеть и протестировать через web-интерфейс по следующей ссылке: https://share.streamlit.io/dkosarevsky/math_modelling/main.py

Программа написана на языке программирования Python 3.9.6 с использованием следующих библиотек:

- streamlit
- pandas
- numpy

Код

```
import matplotlib.pyplot as plt
import streamlit as st
import pandas as pd
import numpy as np

from st_aggrid import AgGrid


class NewtonInterpolationPolynomial(object):
    """
    Class интерполяционный полином Ньютона

    Параметры
    ----------
    nodes : список узлов
    n : степень полинома
    x : значение аргумента
    """

    def __init__(self, nodes, n, x):
        self.nodes = self.sort_nodes(nodes)
        self.range_ = n + 1
        self.n = n
        self.x = x

    @staticmethod
    def sort_nodes(arr):
        """ Сортировка узлов """
        return arr[arr[:, 0].argsort()]

    @staticmethod
    def find_nearest(self):
        """ Поиск индекса ближайшего к аргументу значения узла """
        array = np.asarray(self.nodes[:, 0])
        idx = (np.abs(array - self.x)).argmin()
        return idx

    @staticmethod
    def select_nodes(self):
        """ Выбор новых массивов узлов в диапазоне n+1 от ближайшей к
аргументу точке """
        from_ = self.find_nearest(self) - self.range_ // 2
        if from_ < 0:
            from_ = 0

        to = from_ + self.range_
        if to > self.nodes.shape[0]:
            to = self.nodes.shape[0]
            from_ = to - self.range_

        return self.nodes[np.ix_(range(from_, to), [0, 1])]
```

```
    @staticmethod
    def calc_divided_diffs(self, nodes):
        """ Калькуляция значений разделённых разностей """
        n = self.range_
        divided_diffs = np.zeros([n, n])
        divided_diffs[:, 0] = nodes[:, 1]
        x = nodes[:, 0]

        for j in range(1, n):
            for i in range(n - j):
                divided_diffs[i][j] = (divided_diffs[i + 1][j - 1] -
divided_diffs[i][j - 1]) / (x[i + j] - x[i])

        return divided_diffs

    @staticmethod
    def find_polynomial(self, nodes, diffs):
        """ Поиск полинома Ньютона при фиксированном x """
        x_data = nodes[:, 0]
        coefficients = diffs[0]
        n = self.n
        p = coefficients[n]

        for k in range(1, n + 1):
            p = coefficients[n - k] + (self.x - x_data[n - k]) * p

        return p

    def calc(self):
        """ Вычисление значения полинома """
        selected_nodes = self.select_nodes(self)
        divided_diffs = self.calc_divided_diffs(self, selected_nodes)
        polynomial = self.find_polynomial(self, selected_nodes,
divided_diffs)

        return selected_nodes, divided_diffs, polynomial


def main():
    st.markdown("### Лабораторная работа №1")
    st.markdown("**Тема:** Построение и программная реализация алгоритма
полиномиальной интерполяции табличных функций")
    st.markdown("""
    **Цель работы:** Получение навыков построения алгоритма интерполяции
таблично заданных функций полиномом Ньютона.
    """)

    x_arr = [0.00, 0.15, 0.30, 0.45, 0.60, 0.75, 0.90, 1.05]
    y_arr = [1.000000, 0.838771, 0.655336, 0.450447, 0.225336, -
0.018310, -0.278390, -0.552430]
    df = pd.DataFrame({"x": x_arr, "y": y_arr})

    st.subheader("Таблица функции с количеством узлов N:")
    grid_return = AgGrid(
        df,
```

```
        editable=True,
        height=260,
        reload_data=False,
        theme="dark",
    )
    arr = grid_return["data"].to_numpy()

    c0, c1 = st.columns(2)
    n = c0.number_input("Введите степень полинома - n:", min_value=0,
max_value=7, value=3, step=1)
    x = c1.number_input("Введите значение аргумента - x:", min_value=.0,
max_value=1., value=.525, format="%.3f")

    st.write("---")

    ni = NewtonInterpolationPolynomial(arr, int(n), x)
    new_nodes, diffs, poly = ni.calc()
    st.subheader("Таблица значений разделённых разностей:")
    st.write(pd.DataFrame(diffs).replace(0, np.nan))
    st.write(f"Значение полинома y(x) = {poly:.5f}")

    st.write("---")

    ni_root = NewtonInterpolationPolynomial(np.fliplr(arr), int(n), 0)
    _, _, root = ni_root.calc()
    st.write(f"Значение корня y(x¯) = {root:.5f}")

    st.write("---")

    st.subheader("График функции:")
    plt.figure(figsize=(12, 8))
    plt.plot(arr[:, 0], arr[:, 1], 'bo')
    plt.plot(new_nodes[:, 0], new_nodes[:, 1])
    st.pyplot(plt)


if __name__ == "__main__":
    main()
```

Вопросы

1. Будет ли работать программа при степени полинома n=0?

Да, программа работает. При вычислении y(x) программа возвращает значение функции из таблицы в точке, которая ближе всего к x.

2. Как практически оценить погрешность интерполяции? Почему сложно применить для этих целей теоретическую оценку?

Для практической оценки погрешности интерполяции мы можем воспользоваться оценкой первого отброшенного члена при вычислении многочлена Ньютона. Теоретическую оценку сложно применить, потому что производная функции далеко не всегда известна, а она необходима при вычислении погрешности.

3. Если в двух точках заданы значения функции и ее первых производных, то полином какой степени может быть построен на этих точках?

Мы можем построить полиномы 0 и 1 степени.

4. В каком месте алгоритма построения полинома существенна информация об упорядоченности аргумента функции (возрастает, убывает)?

На этапе формирования конфигурации из (n+1) узлов. Все узлы должны быть максимально близки к значению x.

5. Что такое выравнивающие переменные и как их применить для повышения точности интерполяции?

Если функция (точнее, ее разделенные разности) значительно меняется на протяжении нескольких интервалов сетки, то интерполяция обобщенным многочленом обычно недостаточно точна для дифференцирования этой функции. Для таких функций особенно полезна квазилинейная интерполяция, производимая при помощи выравнивающих переменных.

Выравнивающие переменные - переменные, которые при замене позволяют сглаживать быстропеременную функцию, в некоторых случая вплоть до получения линейной функции.