Министерство образования и науки 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

ЛАБОРАТОРНАЯ РАБОТА №1.

Тема: Построение и программная реализация алгоритма полиномиальной интерполяции табличных функций.

Студент Мицевич Максим Дмитриевич
фамилия, имя, отчество

Группа ИУ7-41Б

Студент _____ Мицевич М. Д.
подпись, дата *фамилия, и.о.*

Проверяющий _____ Градов В. М.
подпись, дата *фамилия, и.о.*

Оценка _____

Москва.
2021г

Цель работы. Получение навыков построения алгоритмаинтерполяции таблично заданных функций полиномамиНьютонаи Эрмита.

I. Исходные данные

1) Таблица функции и её производных

x	y	y`
0.00	1.000000	--1.000000
0.15	0.838771	-1.14944
0.30	0.655336	-1.29552
0.45	0.450447	-1.43497
0.60	0.225336	-1.56464
0.75	-0.018310	-1.68164
0.90	-0.278390	-1.78333
1.05	-0.552430	-1.86742

2) Степень аппроксимирующего полинома — n.
Зная степень полинома, мы выбираем n+1 узлов из таблицы для интерполяции.

3) Значение аргумента, для которого выполняется интерполяция.
Из таблицы нужно выбирать узлы, которые удалены от значения аргумента на наибольшее расстояние.

II. Код программы

Код программы представлен на листингах 1-3

Листинг 1. Код newton.py

```
from math import fabs

class Newton_polynom():
    def __init__(self, table, power, arg):
        self.power = power
        self.arg = arg
        self.table = table.copy()
        self.diff = []

    def find_nearest(self):
```

```
        self.table.sort(key = lambda x: fabs(x[0] -
self.arg))

    def find_diff(self):
        self.diff.append([x[1] for x in self.table])
        for i in range(self.power - 1, -1, -1):
            diff = []
            for j in range(0, i + 1):
                x = (self.diff[-1][j] - self.diff[-1][j +
1]) / (self.table[j][0] - self.table[j + self.power - i]
[0])
                diff.append(x)
            self.diff.append(diff)

    def find_polynom(self):
        self.find_nearest()
        self.find_diff()

    def res(self, x):
        y = self.diff[0][0]
        for k in range(1, self.power + 1):
            xmlt = 1
            for j in range(0, k):
                xmlt *= (x - self.table[j][0])
            y += xmlt * (self.diff[k][0])

        return y
```

Листинг 2. Код hermit.py

```
from math import fabs, ceil


class Hermit_polynom:
    def __init__(self, table, power, arg):
        self.power = power
        self.arg = arg
        self.table = table.copy()
        self.diff = []

    def find_interval(self):
```

```
        self.table.sort(key=lambda x: fabs(x[0] -
self.arg))
        for i in range(len(self.table) - 1, -1, -1):
            self.table.insert(i + 1, self.table[i])

    def find_diff(self):
        eps = 1e-6

        self.diff.append([x[1] for x in self.table])
        for i in range(self.power - 1, -1, -1):
            diff = []
            for j in range(0, i + 1):
                if fabs(self.table[j][0] - self.table[j +
self.power - i][0]) < eps:
                    x = self.table[j + 1][2]
                else:
                    x = (self.diff[-1][j] - self.diff[-1][j
+ 1]) / \
                        (self.table[j][0] - self.table[j +
self.power - i][0])
                diff.append(x)
            self.diff.append(diff)

    def find_polynom(self):
        self.find_interval()
        self.find_diff()

    def res(self, x):
        y = self.diff[0][0]
        for k in range(1, self.power + 1):
            xmlt = 1
            for j in range(0, k):
                xmlt *= (x - self.table[j][0])
            y += xmlt * (self.diff[k][0])

        return y
```

Листинг 3. Код программы main.py

```
from newton import Newton_polynom
from hermit import Hermit_polynom
```

```
table = [[0.00, 1.000000, -1.000000],
         [0.15, 0.838771, -1.14944],
         [0.30, 0.655336, -1.29552],
         [0.45, 0.450447, -1.43497],
         [0.60, 0.225336, -1.56464],
         [0.75, -0.018310, -1.68164],
         [0.90, -0.278390, -1.78333],
         [1.05, -0.552430, -1.86742]]

def find_root(n):
    table_revert = [(x[1], x[0]) for x in table]
    newton = Newton_polynom(table_revert, n, 0)
    newton.find_polynom()

    return newton.res(0)


def main():
    print("x = 0.525")
    print("| power | " + " " * 4 + "Newton | " + " " * 4 +
"Hermit |")
    for n in range(1, 5):
        newton = Newton_polynom(table, n, 0.525)
        newton.find_polynom()
        yn = newton.res(0.525)

        hermit = Hermit_polynom(table, n, 0.525)
        hermit.find_polynom()
        yh = hermit.res(0.525)

        print(f"| {n:5} | {yn:10.8} | {yh:10.8} |")
        print()

    print("| power | " + " " * 6 + "root |")
    for n in range(1, 5):
        root = find_root(n)
        print(f"| {n:5} | {root:10.8} |")

    n = int(input("Input polynom power: "))
    arg = float(input("Input argument: "))
    x = float(input("Input argument: "))
```

```
    newton = Newton_polynom(table, n, arg)
    newton.find_polynom()
    yn = newton.res(x)

    hermit = Hermit_polynom(table, n, arg)
    hermit.find_polynom()
    yh = hermit.res(x)

    print(f"Newton res - {yn:.8}")
    print(f"Hermit res - {yh:.8}")

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

III. Результат работы

Значения полиномов Ньютона и Эрмита степеней от 1 до 4 в точке 0.525. Значение аргумента, для которого выполняется интерполяция также равно 0.525

```
x = 0.525
| power |     Newton |     Hermit |
|     1 |  0.3378915 |   0.342684 |
|     2 |  0.34020837| 0.34028775 |
|     3 | 0.34031381 | 0.34032281 |
|     4 | 0.34032448 | 0.34032397 |
```

Найдем корень, заданной функции при помощи обратной интерполяции и полинома Ньютона разных степеней:

```
| power |       root |
|     1 |  0.7387275 |
|     2 | 0.73904613 |
|     3 | 0.73909485 |
|     4 | 0.73908771 |
```

IV. Вопросы при защите лабораторной работы

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

Да, P(x)=y(x0), где y(x0) — значение узла, ближайшего к аргументу, для которого выполняется интерполяция.

2. Как практически оценить погрешность интерполяции? Почему сложно применить для этих целей теоретическую оценку?
Можно оценить погрешность интерполяции, если знать как убывают члены ряда. В случае, когда они убывают сильно можно взять все начиная с определенного члена. Его значение будет являться погрешностью.

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

Так как заданы значения функции в точках и ее первые производные, можно построить полином Эрмита. Всего у нас есть 2 * 2 = 4 узла, значит степень полинома равна 4 — 1 = 3.

4. В каком месте алгоритма построения полинома существенна информация об упорядоченности аргумента функции (возрастает, убывает)?
Данная информация важна при выборе узлов, наиболее близких к аргументу, для которого выполняется интерполяция. Если узлы как-то отсортированны, то поиск узлов становится проще

5. Что такое выравнивающие переменные и как их применить для повышения точности интерполяции?
Для повышение точности может использоваться метод выравнивающих переменных. Метод заключается в том, что функцию на протяжении нескольких шагов таблицы приближают к некой элементарной функции. После строится таблица на новых переменных, полученных из элементарной функции, их интерполяция и нахождение обратным преобразованием. Такие переменные называются выравнивающими.