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»

Лабораторная работа №1

Тема Построение и программная реализация алгоритма полиномиальной интерполяции табличных функций.

Студент Бугаенко А.П.

Группа ИУ7-35Б

Оценка (баллы) _____

Преподаватель Градов В.М.

Москва.
2021 г

Цель работы. Получение навыков построения алгоритма интерполяции таблично заданных функций полиномами Ньютона и Эрмита.

1 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
3. Значение аргумента, для которого выполняется интерполяция.

2 Код программы

Код программы представлен на листинге ниже.

Листинг main.py

```
import math as m
import pandas as pd

table = [
    {"x": 0.00, "y": 1.000000, "y'": -1.000000},
    {"x": 0.15, "y": 0.838771, "y'": -1.14944},
    {"x": 0.30, "y": 0.655336, "y'": -1.29552},
    {"x": 0.45, "y": 0.450447, "y'": -1.43497},
    {"x": 0.60, "y": 0.225336, "y'": -1.56464},
    {"x": 0.75, "y": -0.018310, "y'": -1.68164},
    {"x": 0.90, "y": -0.278390, "y'": -1.78333},
    {"x": 1.05, "y": -0.552430, "y'": -1.86742}]

def Divided_diff(x, y):
    new_y = []
    n = len(x) - len(y)
    for i in range(0, len(y) - 1):
        new_y.append((y[i] - y[i + 1]) / (x[i] - x[i + n + 1]))
    return new_y

def SortTableNewton(table, x, n):
    table = sorted(table, key=lambda d: abs(d["x"] - x))
    table = sorted(table[:n+1], key=lambda t: t["x"])
    return table
```

```
def FormXYNewton(table):
    X = []
    Y = []
    for row in table:
        X.append(row["x"])
        Y.append(row["y"])
    return X, Y

def CountDiffDivNewton(X, Y):
    Y_arr = [Y]
    while len(Y_arr[-1]) != 1:
        Y = Divided_diff(X, Y)
        Y_arr.append(Y)
    return Y_arr

def CountPolynomNewton(Y_arr):
    polym = []
    for Y in Y_arr:
        polym.append(Y[0])
    return polym

def GetValApproxNewton(polym, X, x):
    x_mult = 1
    result = 0
    for i in range(0, len(polym)):
        result += x_mult * polym[i]
        x_mult = x_mult * (x - X[i])
    return result

def NewtonApprox(x, n, table):

    table = SortTableNewton(table, x, n)
    X, Y = FormXYNewton(table)

    Y_arr = CountDiffDivNewton(X, Y)
    polym = CountPolynomNewton(Y_arr)

    return GetValApproxNewton(polym, X, x)

def SortTableErmit(table, x, n):
    table = sorted(table, key=lambda d: abs(d["x"] - x))
    table = sorted(table[:m.floor(n/2) + 1], key=lambda t:
t["x"])
    return table

def FormXYY_Ermit(table):
    X = []
    Y = []
    Y_ = []
    for row in table:
        X.append(row["x"])
        Y.append(row["y"])
        Y_.append(row["y'"])
    return X, Y, Y_

def Divided_diff_ermit(x, y, n):
    new_y = []
    n = n + 1 - len(y)
    for i in range(0, len(y) - 1):
        new_y.append((y[i] - y[i + 1]) / (x[int(i/2)] -
x[int((i + n + 1)/2)]))
    return new_y
```

3 Результаты работы

1. Представить разностный аналог краевого условия при $x = l$ и его краткий вывод интегро - интерполяционным методом.

Обозначим $F = -k(x)\frac{dT}{dx}$.

4 Вопросы при защите лабораторной работы