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»

Выполнил студент: __*Клименко Алексей Константинович*_____

фамилия, имя, отчество

Группа: _____*ИУ7-45Б*_____

Проверил, к.п.н.: _____

подпись, дата

Оценка _____ Дата _____

2021 г.

Цель работы

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

Исходные данные

1. Таблица функции и её производных

x	y	y'
0.00	1.000000	-1.00000
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 аппроксимироующего полинома - n,
3. Значение аргумента, для которого выполняется интерполяция.

Код программы

```
import sys


# загрузка данных из файла
filename = sys.argv[1]
X, Y, Y_P = [], dict(), dict()
for line in open(filename, "rt").readlines():
     nums = [float(num) for num in line.split()]
     X.append(nums[0])
     Y[nums[0]], Y_P[nums[0]] = nums[1:]

def fill_high_order_diffs(diffs, X, n):
     for order in range(1, n):
```

```
        for i in range(n - order):
            xx = tuple(X[i:i + 2 + order])
            diffs[xx] = (diffs[xx[:-1]] - diffs[xx[1:]]) / (xx[0] - xx[-1])

def make_y(X, Y, diffs, n, x):
    y, x_prod = Y[X[0]], x - X[0]
    for i in range(n):
        xx = tuple(X[:i+2])
        y += x_prod * diffs[xx]
        x_prod *= x - X[i+1]
    return y

def newton(X, Y, n, x):
    X = sorted(X, key=lambda xi: abs(x - xi))
    X = sorted(X[:n+1])

    diffs = dict()
    for i in range(n):
        xx = tuple(X[i:i+2])
        diffs[xx] = (Y[xx[0]] - Y[xx[1]]) / (xx[0] - xx[1])

    fill_high_order_diffs(diffs, X, n)
    return make_y(X, Y, diffs, n, x)

def ermit(X, Y, Y_P, n, x):
    X = sorted(X, key=lambda xi: abs(x - xi))
    X = sorted(X[:(n+2)//2] * 2)

    diffs = dict()
    for i in range(n):
        xx = tuple(X[i:i+2])
        if X[i] == X[i+1]:
            diffs[xx] = Y_P[xx[0]]
        else:
            diffs[xx] = (Y[xx[0]] - Y[xx[1]]) / (xx[0] - xx[1])

    fill_high_order_diffs(diffs, X, n)
    return make_y(X, Y, diffs, n, x)

x = 0.525
print(f"x = {x}")
print("  n  |   y (newton)   |  y (ermit)  ")
print("=====|================|=============")
for n in range(0, 5):
    y_n = newton(X, Y, n, x)
    y_e = ermit(X, Y, Y_P, n, x)
    print(f"  {n}  |    {y_n:.5g}     |   {y_e:.5g}")
```

Результаты работы

1. Значения y(x) при степенях полиномов Ньютона и Эрмита n = 1, 2, 3 и 4 при фиксированных x:

x = 0.525

```
$ make
py main.py data.txt
x = 0.525
  n   |   y (newton)   |  y (ermit)
======|================|============
  0   |    0.22534     |   0.22534
  1   |    0.33789     |   0.34268
  2   |    0.34021     |   0.34036
  3   |    0.34031     |   0.34032
  4   |    0.34032     |   0.34032
```

x = 0.55

```
$ make
py main.py data.txt
x = 0.55
  n   |   y (newton)   |  y (ermit)
======|================|============
  0   |    0.22534     |   0.22534
  1   |    0.30037     |   0.30357
  2   |    0.30243     |   0.30257
  3   |    0.30252     |   0.30252
  4   |    0.30252     |   0.30252
```

x = 0.575

```
$ make
py main.py data.txt
x = 0.575
  n   |   y (newton)   |  y (ermit)
======|================|============
  0   |    0.22534     |   0.22534
  1   |    0.26285     |   0.26445
  2   |    0.26414     |   0.26422
  3   |    0.26419     |   0.26419
  4   |    0.26419     |   0.26419
```

2. Результат нахождения корня заданной функции:

```
$ make
py main.py data.txt
y = 0.0
  n   |   x (newton)
======|===============
  0   |    0.75000
  1   |    0.73873
  2   |    0.73905
  3   |    0.73909
  4   |    0.73909
```

Получили значение x = 0.73909

Контрольные вопросы

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

Да, будет. В этом случае результат будет просто ближайшим табличным значением функции для заданной точки x.

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

Теоритическая оценка зачастую не подходит для практических рассчётов вследствие того, что производная заданной функции может быть не известна. Поэтому для практической оценки погрешности интерполяции следует использовать первый отброшенный член ряда.

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

Исходные данные представляют собой 4 независимых параметра (2 - значения функции в двух точках, 2 - значения производных в точках), поэтому степень полинома, который может быть построен на заданных точках, будет равна **трём** (на единицу меньше числа параметров).

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

На шаге построения конфигурации из (n + 1) узлов. Все эти узлы должны быть настолько близко к целевому значению x, на сколько это возможно. Если при этом исходная таблица упорядочена по аргументу функции, то тогда выбор узлов равносилен взятию диапазона, содержащего целевое значение x.

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

Выравнивающие переменные - это переменные (u и v, например), которые при замене обычных (x и y) позволяют "сгладить" быстропеременную функцию (получить на некоторых её участках график близкий к прямой). Например, для функции $y(x) = e^x$, замена $u = \ln(x)$, $v = y$ позволяет получить линейную функцию.

При этом важно, чтобы преобразования переменных были несложными (как в прямое, так и обратное). Зачастую применяются операции логарифмирования, экспоненцирования, а также тригонометрические функции.