



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:** Козлова И. В.

**Группа:** ИУ7-42Б

**Оценка (баллы):** \_\_\_\_\_

**Преподаватель:** Градов В.М.

Москва  
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	1	-1
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.

## Код программы

```
import math
import openpyxl as xls
import numpy as np

def parse_table():
    """
        Загрузка таблицы из исходных данных в
        программу.
    """
    pos = 3
    points = xls.load_workbook("points.xlsx").active
    table = []
    while points.cell(row = pos, column = 1).value is not None:
        table.append([float(points.cell(row = pos, column = 1).value), \
                      float(points.cell(row = pos, column = 2).value), \
                      float(points.cell(row = pos, column = 3).value)])
        pos += 1
    table.sort()
    count = len(table) - 1
```

```

arr_x = []
arr_y = []
for i in range(len(table)):
    arr_x.append(table[i][0])
    arr_y.append(table[i][1])
return table, arr_x, arr_y, count

def input_x():
    """
        Ввод аргумента. (в случае ошибки дается
        еще попытка)
    """
    print("Enter X: ")
    flag = 0
    x = 0
    while flag == 0:
        x = input(float)
        try:
            val = float(x)
            flag = 1
        except ValueError:
            print("Some error! Try again")
    return float(x)

def find_x0_xn(data, power, arg):
    """
        Нахождение начального и конечного
        индекса в таблице (x0 и xn).
    """
    index_x = 0

    while arg > data[index_x][0]:
        index_x += 1
    index_x0 = index_x - power // 2 - 1
    index_xn = index_x + (power // 2) + (power % 2) - 1
    if index_xn > len(data) - 1:
        index_x0 -= index_xn - len(data) + 1
        index_xn = len(data) - 1
    elif index_x0 < 0:
        index_xn += -index_x0
        index_x0 = 0
    return index_x0, index_xn

def div_diff(x, y, node):
    """
        Расчет разделенных разниц для полинома
        Ньютона
    """

```

```

pol = []
for i in range(node):
    pol.append([0] * (node + 1))
for i in range(node):
    pol[i][0], pol[i][1] = x[i], y[i]
i = 2
new_node = node - 1
while i < (node + 1):
    j = 0
    while j < new_node:
        pol[j][i] = round((pol[j + 1][i - 1] - pol[j][i - 1]) \
            / (pol[i - 1][0] - pol[0][0]), 5)
        j += 1
    i += 1
    new_node -= 1
return pol

def polinom_n(x, y, node, arg):
    """
        Расчет значение функции от заданного
    а р г у м е н т а .
        Полином Ньютона.
    """
    pol = div_diff(x, y, node)
    y = pol[0][1]
    i = 2
    while i < node + 1:
        j, p = 0, 1
        while j < i - 1:
            p *= (arg - pol[j][0])
            j += 1
        y += pol[0][i] * p
        i += 1
    return y

def hermite_interpolate(data, node, arg, coords_x):
    """
        Расчет таблицы для полинома Эрмита.
    """
    # Поиск нужных начала и конца отрезка x
    x0, xn = find_x0_xn(data, node // 2, arg)
    data = data[x0 : xn + 1]
    pol = []
    for i in range(2 * len(data)):
        pol.append([0] * (2 * node + 3))
    i = 0
    for j in range(len(data)):
        pol[i][0], pol[i][1], pol[i][2] = data[j][0], data[j][1], data[j][2]

```

```

        i += 1
        pol[i][0], pol[i][1] = data[j][0], data[j][1]
        i += 1
    i = 2
    # Заполнения таблицы, как для полинома
Ньютона
    for j in range(len(pol) - 1):
        if j % 2 == 1:
            pol[j][i] = (pol[j][1] - pol[j + 1][1]) \
                / (pol[j][0] - pol[j + 1][0])
    i = 3
    new_node = node - 2
    while i < len(pol):
        j = 0
        while j < new_node:
            pol[j][i] = round((pol[j + 1][i - 1] - pol[j][i - 1]) \
                / (pol[i - 1][0] - pol[0][0]), 5)
            j += 1
        i += 1
        new_node -= 1
    return pol

def polynom_h(pol, node, arg):
    """
        Расчет значение функции от заданного
аргумента.
        Полином Эрмита.
    """
    y = pol[0][1]
    i = 2
    while i < node + 2:
        j = 0
        p = 1
        while j < i - 1:
            p *= (arg - pol[j][0])
            j += 1
        y += pol[0][i] * p
        i += 1
    return y

def main():
    # Загрузка таблицы и ввод исходных данных
    data, coords_x, coords_y, count = parse_table()
    x = input_x()
    arr_n = [1, 2, 3, 4]

    print("\nИнтерполяция с помощью полинома
Ньютона и Эрмита\n",

```

```

        "| n | x | п. Ньютона | п. Эрмита |")
for n in arr_n:
    print(" | {} | {} |".format(n, x), end="")
    flag = False
    # Проверка на наличие данного аргумента в
таблице
    for i in range(0, len(data)):
        if x == data[i][0]:
            print(" {} |".format(round(data[i][1], 5)), end="")
            flag = True
    # В случае, если аргумента в таблице нет
    if not flag:
        # Полином Ньютона
        x0, xn = find_x0_xn(data, n, x)
        ax = coords_x[x0 : xn + 1]
        ay = coords_y[x0 : xn + 1]
        if len(ax):
            my_root = polinom_n(ax, ay, n + 1, x)
            print(" {} |".format(round(my_root, 5)), end="")

        # Полином Эрмита
        pol = hermite_interpolate(data, n, x, coords_x)
        my_root2 = polynom_h(pol, n, x)
        print(" {} |".format(round(my_root2, 5)), end="\n")

print("\nОбратная интерполяция с помощью
полинома Ньютона\n",
      "| n | x | Корень |")
for n in arr_n:
    print(" | {} | {} |".format(n, 0), end="")
    flag = False
    # Проверка на наличие данного аргумента в
таблице
    for i in range(0, len(data)):
        if 0 == data[i][1]:
            print(" {} |".format(round(data[i][0], 5)), end="")
            flag = True
    # В случае, если аргумента в таблице нет
    if not flag:
        # Обмен столбцами x и y из исходной
таблицы
        n_data = []
        for i in range(len(data)):
            n_data.append([data[i][1], data[i][0], data[i][2]])
        n_data.sort()
        coords_y.clear()
        coords_x.clear()
        for i in range(len(n_data)):

```

```

        coords_x.append(n_data[i][0])
        coords_y.append(n_data[i][1])
# Полином Ньютона для аргумента x = 0
x0, xn = find_x0_xn(n_data, n, 0)
ax = coords_x[x0 : xn + 1]
ay = coords_y[x0 : xn + 1]
if len(ax):
    my_root = polinom_n(ax, ay, n + 1, 0)
    print(" {} |".format(round(my_root, 5)))

if __name__ == "__main__":
    main()

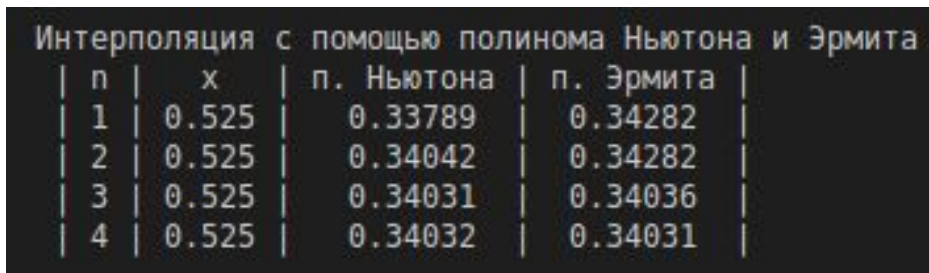
```

## Результаты работы

1. Значения  $y(x)$  при степенях полиномов Ньютона и Эрмита  $n=1, 2, 3$  и  $4$  при фиксированном  $x$ , например,  $x=0.525$  (середина интервала  $0.45-0.60$ ).

Результаты свести в таблицу для сравнения полиномов.

(Вывод программы)



n	x	п. Ньютона	п. Эрмита
1	0.525	0.33789	0.34282
2	0.525	0.34042	0.34282
3	0.525	0.34031	0.34036
4	0.525	0.34032	0.34031

(Для удобства таблица не с фотографии)

n	x	п. Ньютона	п. Эрмита
1	0.525	0.33789	0.34282
2	0.525	0.34042	0.34282
3	0.525	0.34031	0.34036
4	0.525	0.34032	0.34031

(Таблица в исходных данных составлена по функции  $y(x) = \cos(x) - x$ ; Точное значение в точке  $x = 0.525$  равно  $0.3403239416229412$ )

2. Найти корень заданной выше табличной функции с помощью обратной интерполяции, используя полином Ньютона.

(Вывод программы)

Обратная интерполяция с помощью полинома Ньютона			
n	x	Корень	
1	0	0.73873	
2	0	0.73944	
3	0	0.73944	
4	0	0.74762	

(Для удобства таблица не с фотографии)

n	x	Корень
1	0	0.73873
2	0	0.73944
3	0	0.73944
4	0	0.74762

## Вопросы при защите лабораторной работы

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

Да, работать программа с такой степенью будет, как расчет интерполяции с помощью полинома Ньютона, так и с помощью полинома Эрмита. Функция вернет значение из таблицы в точке, которая находится ближе к заданному аргументу. Но точность при такой конфигурации будет низкой.

Интерполяция с помощью полинома Ньютона и Эрмита				
n	x	п. Ньютона	п. Эрмита	
0	0.525	0.45045	0.45045	
1	0.525	0.33789	0.34282	
2	0.525	0.34042	0.34282	
3	0.525	0.34031	0.34036	
4	0.525	0.34032	0.34031	

Расчет корня с помощью обратной интерполяции тоже будет работать - это значение аргумента из данной таблицы, в которой функция принимает значение ближе к нулю.

Обратная интерполяция с помощью полинома Ньютона			
n	x	Корень	
0	0	0.75	
1	0	0.73873	
2	0	0.73944	
3	0	0.73944	
4	0	0.74762	



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

Практически оценить погрешность интерполяции можно при помощи оценки первого отброшенного члена в полиноме Ньютона. При этом в полиноме остаются члены, которые больше заданной погрешности расчетов.

Теоретическую погрешность многочлена Ньютона можно оценить с помощью формулы (где используются производные данной функции):

$$|y(x) - P_n(x)| \leq \frac{M_{n+1}}{(n+1)!} |\varpi_n(x)|, \quad \text{где } M_{n+1} = \max |y^{(n+1)}(\xi)|, \quad - \text{максимальное}$$

значение производной интерполируемой функции, а также  $\varpi_n(x) = \prod_{i=0}^n (x - x_i)$

Именно поэтому теоретическую погрешность сложно оценить.

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

При данном условии можно построить полиномы Эрмита 0, 1, 2 и 3 степени а полиномы Ньютона - 0 и 1 степени.

Минимальная степень равна 0.

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

Информация об упорядоченности аргумента функции существенна при выборе приближенного интервала значений (из (n+1) узлов, которые по возможности расположены симметрично относительно заданного аргумента).

Если аргумент будет неупорядоченным, то значение функции получится с низкой точностью или совсем неверным.

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

Если функция, а точнее ее разделенные разности, значительно меняются на нескольких интервалах, то интерполяция обобщенным многочленом обычно не будет точной для дифференцирования данной функции. Поэтому для таких функций используется квазилинейная интерполяция, которая производится при помощи выравнивающих переменных. То есть выравнивающие переменные используются для того, чтобы повысить точность вычисления производной функции.