**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** Пронин А.С.

**Группа** ИУ7-42Б

**Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Преподаватель** Градов В.М.

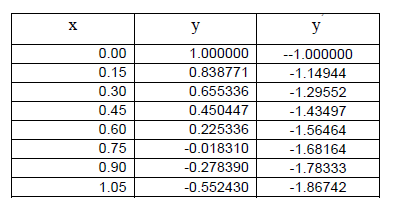
Москва.

2021 г

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

# ****1. Исходные данные****

1. Таблица функции и её производных



2. Степень аппроксимирующего полинома - n. (От 1 до 4)

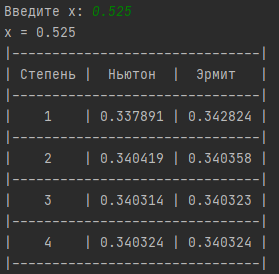
3. Значение аргумента, для которого выполняется интерполяция. (Вводится пользователем)

# 2. Код программы

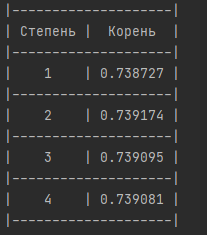
table1 = [(0.00, 1.000000, -1.00000),  
 (0.15, 0.838771, -1.14944),  
 (0.30, 0.655336, -1.29552),  
 (0.45, 0.450447, -1.43497),  
 (0.60, 0.225336, -1.56464),  
 (0.75,-0.018310, -1.68164),  
 (0.90,-0.278390, -1.78333),  
 (1.05,-0.552430, -1.86742)]  
  
def search\_value(x, table):  
 rez = []  
 rez.append(list(table[len(table) - 1]))  
 index = len(table) - 1  
  
 for i in range(1, len(table)):  
 if (x - table[i - 1][0]) \* (x - table[i][0]) <= 0:  
 rez[0] = list(table[i - 1])  
 index = i - 1  
 return rez, index  
  
def make\_data\_Newton(x, n, table):  
 n += 1  
 rez, index = search\_value(x, table)  
  
 tmp = 1  
  
 for i in range(0, n - 1):  
 if (i % 2 == 0):  
 if (index + tmp < len(table)):  
 rez.append(list(table[index + tmp]))  
 elif (index - tmp >= 0):  
 rez.append(list(table[index - tmp]))  
 tmp += 1  
 else:  
 print("Not enough data")  
 rez = []  
 break  
 else:  
 if (index - tmp >= 0):  
 rez.append(list(table[index - tmp]))  
 tmp += 1  
 elif (index + tmp + 1 < len(table)):  
 tmp += 1  
 rez.append(list(table[index + tmp]))  
 tmp += 1  
 else:  
 print("Not enough data")  
 rez = []  
 break  
  
 return rez  
  
def make\_data\_Ermit(x, n, table):  
 n += 1  
 rez, index = search\_value(x, table)  
  
 tmp = 1  
  
 counter = 1  
 down = 1  
  
 while (counter < n):  
 if (counter % 2 == 1):  
 rez.append(list(rez[counter - 1]))  
 else:  
 if (down):  
 if (index + tmp < len(table)):  
 rez.append(list(table[index + tmp]))  
 elif (index - tmp >= 0):  
 rez.append(list(table[index - tmp]))  
 tmp += 1  
 else:  
 print("Not enough data")  
 rez = []  
 break  
 else:  
 if (index - tmp >= 0):  
 rez.append(list(table[index - tmp]))  
 tmp += 1  
 elif (index + tmp + 1 < len(table)):  
 tmp += 1  
 rez.append(list(table[index + tmp]))  
 tmp += 1  
 else:  
 print("Not enough data")  
 rez = []  
 break  
 down = (down + 1) % 2  
 counter += 1  
  
 return rez  
  
  
def get\_dif(data, yi, yj, xi, xj):  
 if (data[xi][0] == data[xj][0]):  
 dif = data[xi][2]  
 else:  
 dif = (data[yi][1] - data[yj][1]) / (data[xi][0] - data[xj][0])  
 return dif  
  
def calculate\_polinom(data, n, x):  
 n += 1  
 rez = data[0][1]  
 difs = list(data)  
 for i in range(n - 1):  
 for j in range(n - 1 - i):  
 difs[j][1] = get\_dif(difs, j, j + 1, j, j + 1 + i)  
 tmp = difs[0][1]  
 for k in range (i + 1):  
 tmp \*= x - difs[k][0]  
 rez += tmp  
  
 return rez  
  
cur\_x = float(input("Введите x: "))  
print("x =", cur\_x)  
  
print ("|-------------------------------|")  
print ("| Степень | Ньютон | Эрмит |")  
print ("|-------------------------------|")  
for i in range(1, 5):  
 stepen = i  
 print("|{:^9d}".format(stepen), end = "")  
  
 new\_data\_Newton = make\_data\_Newton(cur\_x, stepen, table1)  
  
 if (not new\_data\_Newton == []):  
 polinom\_Newton = calculate\_polinom(new\_data\_Newton, stepen, cur\_x)  
 if (polinom\_Newton > 0):  
 print("| {:^8.6f} |".format(polinom\_Newton), end = "")  
 else:  
 print("|{:^8.6f} |".format(polinom\_Newton), end = "")  
  
 new\_data\_Ermit = make\_data\_Ermit(cur\_x, stepen, table1)  
  
 if (not new\_data\_Ermit == []):  
 polinom\_Ermit = calculate\_polinom(new\_data\_Ermit, stepen, cur\_x)  
 if (polinom\_Ermit > 0):  
 print(" {:^8.6f} |".format(polinom\_Ermit), end = "")  
 else:  
 print("{:^8.6f} |".format(polinom\_Newton), end = "")  
  
 print("\n|-------------------------------|")  
  
cur\_y = 0  
  
inverse\_table1 = []  
for i in range(len(table1)):  
 inverse\_table1.append((table1[i][1], table1[i][0]))  
  
print ("|--------------------|")  
print ("| Степень | Корень |")  
print ("|--------------------|")  
for i in range(1, 5):  
 stepen = i  
 print("|{:^9d}".format(stepen), end = "")  
  
 new\_data\_Newton = make\_data\_Newton(cur\_y, stepen, inverse\_table1)  
  
 if (not new\_data\_Newton == []):  
 polinom\_Newton = calculate\_polinom(new\_data\_Newton, stepen, cur\_y)  
 if (polinom\_Newton > 0):  
 print("| {:^8.6f} |".format(polinom\_Newton), end = "")  
 else:  
 print("|{:^8.6f} |".format(polinom\_Newton), end = "")  
  
 print("\n|--------------------|")

# 3. Результаты работы

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



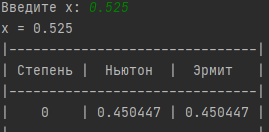
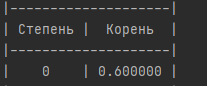
2. Найти корень заданной выше табличной функции с помощью обратной интерполяции, используя полином Ньютона.



# 4. Вопросы при защите лабораторной работы

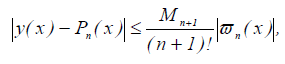
***1. Будет ли работать программа при степени полинома n=0?***

В моём случае программа сработает, но вадаст лишь ближнее значение из начальной таблицы – соответственно ни о какой точности в таком случае говорить нельзя

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

Практически оценить погрешность интерполяции можно оценкой первого отброшенного члена.

Теоретическую погрешность многочлена Ньютона можно оценить по формуле

Но для её вычисления необходимо знать производные интерполируемой функции, которые обычно неизвестны. Поэтому для определения погрешности удобнее воспользоваться оценкой первого отброшенного члена.

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

Полином третей степени

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

При построении конфигурации узлов на которой будем строить полином. Это важно потому, что необходимо определить между какими значениями из таблицы аргумент, для которого выполняется интерполяция, находится.

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

Это те переменные которые мы получаем путём простых преоброзований чтобы в них график был близок к прямой хотя бы на отдельных участках. Интерполяцию проводят в новых переменных, а затем обратным интерполированием находят значения изначайльной функции