Федеральное государственное автономное образовательное учреждение высшего образования

Университет ИТМО

Факультет программной инженерии и компьютерной техники

Дисциплина: **Вычислительная математика**

Лабораторная работа №5

“ «Интерполяция функции”

Вариант: 9

Выполнил: Кузнецов Максим Александрович

Группа: P3211

Преподаватель: Малышева Татьяна Алексеевна

Санкт-Петербург 2022 г

Цель работы:

Решить задачу интерполяции, найти значения функции при заданных значениях аргумента, отличных от узловых точек.

Для исследования использовать:

- многочлен Лагранжа;
- многочлен Ньютона;
- многочлен Гаусса.

Условия и задание:

2. Вычислительная реализация задачи:
   - 2.1. Используя первую или вторую интерполяционную формулу Ньютона, первую или вторую интерполяционную формулу Гаусса вычислить значения функции при данных значениях аргумента (для значения X_1 и X_2, см. табл. 1 - 4).
   - 2.2. Построить таблицу конечных разностей.
   - 2.3. **Подробные вычисления привести в отчете.**

x	y
0,25	1,2557
0,30	2,1764
0,35	3,1218
0,40	4,0482
0,45	5,9875
0,50	6,9195
0,55	7,8359

№ варианта	X_1	X_2
1	0,251	0,402
5	0,512	0,372
9	0,255	0,405
13	0,534	0,384
17	0,272	0,445
21	0,551	0,351
25	0,294	0,437

3. Программная реализация задачи:
   - 3.1. Исходные данные задаются в виде: а) набора данных (таблицы x,y), б) на основе выбранной функции (например, sin x).
   - 3.2. Вычислить приближенное значение функции для заданного значения аргумента, введенного с клавиатуры, указанными методами (см. табл.5).
   - 3.3. Построить графики заданной функции с отмеченными узлами интерполяции и интерполяционного многочлена Ньютона/Гаусса (разными цветами).

9	1, 2

Вычислительная часть:

Многочлен Лагранжа

$$L_n(x) = \sum_{i=0}^{n} y_i \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

Формула Ньютона для интерполирования вперёд ($t = (x-x_i) / h$)

$$N_n(x) = y_i + t\Delta y_i + \frac{t(t-1)}{2!}\Delta^2 y_i + \cdots + \frac{t(t-1)\ldots(t-n+1)}{n!}\Delta^n y_i$$

Формула Ньютона для интерполирования назад ($t = (x-x_n) / h$)

$$N_n(x) = y_n + t\Delta y_{n-1} + \frac{t(t+1)}{2!}\Delta^2 y_{n-2} + \cdots + \frac{t(t+1)\ldots(t+n-1)}{n!}\Delta^n y_0$$

i/k	k0	k1	k2	k3	k4	k5	k6
i0	1,2557	0,9207	0,0247	-0,0437	1,0756	-4,1277	10,1917
i1	2,1764	0,9454	-0,019	1,0319	-3,0521	6,064	
i2	3,1218	0,9264	1,0129	-2,0202	3,0119		
i3	4,0482	1,9393	-1,0073	0,9917			
i4	5,9875	0,932	-0,0156				
i5	6,9195	0,9164					
i6	7,8359						

Для X = 0.255:

i/k	k0	k1	k2	k3	k4	k5	k6
i0	1,2557	0,9207	0,0247	-0,0437	1,0756	-4,1277	10,1917
i1	2,1764	0,9454	-0,019	1,0319	-3,0521	6,064	
i2	3,1218	0,9264	1,0129	-2,0202	3,0119		
i3	4,0482	1,9393	-1,0073	0,9917			
i4	5,9875	0,932	-0,0156				
i5	6,9195	0,9164					
i6	7,8359						

$t = (x-x_n) / h = (0.255 - 0.3) / 0.05 = -0.9$

$N_1 = 2{,}1764 + (-0.9) * 0.9207 = 1.348$

Для X = 0.405:

i/k	k0	k1	k2	k3	k4	k5	k6
i0	1,2557	0,9207	0,0247	-0,0437	1,0756	-4,1277	10,1917
i1	2,1764	0,9454	-0,019	1,0319	-3,0521	6,064	
i2	3,1218	0,9264	1,0129	-2,0202	3,0119		
i3	4,0482	1,9393	-1,0073	0,9917			
i4	5,9875	0,932	-0,0156				
i5	6,9195	0,9164					
i6	7,8359						

$t = (x-x_n) / h = (0.405 - 0.45) / 0.05 = -0.9$

$$N_1 = 5{,}9875 + (-0.9) * 1{,}9393 + \frac{(-0.9)(-0.9+1)}{2!} * (1{,}0129) + \cdots + \frac{(-0.9)(-0.9+1)\ldots(-0.9+3)}{4!} * (1.0756) = 4{,}17021$$

Листинг программы:

```
import numpy as np
from numpy import genfromtxt
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.lines as mlines
from matplotlib.pyplot import figure
import sys
import math
import pandas as pd
from math import factorial as fc
def in_float(s = 'Введите число', integer = False, check = [False, 0, 0]):
    flag = True
    while flag:
        flag = False
        try:
            if integer:
                val = int(input(s +': '))
            else:
                val = float(input(s +': '))
            if check[0] and (val < check[1] or val > check[2]):
                raise ValueError
        except ValueError:
            flag = True
            if check[0]:
```

```
                print(f'Попробуйте снова! Введенное число должно принадлежать
интервалу [{check[1]}; {check[2]}]\n')
            else:
                print(f'Попробуйте снова!\n')
    return val

def parse():
    flag = True
    while flag:
        path = input('Путь:\n').strip()
        try:
            a = genfromtxt(path, delimiter=',')
            if True in np.isnan(a) or a.shape[0] != 2:
                raise ValueError
            return a
        except ValueError:
            print('В файле должно быть 2 строчки, в каждой одинаковое
количество чисел\n')
        except OSError:
            print('Такого файла нет.\n')
        print('Попробуйте снова!\n')

def input_vals():
    n = in_float(s = 'Введите количество точек', integer = True)
    print()
    a = []
    for i in range(int(n)):
        a.append([in_float('x'), in_float('y')])
        print()
    return np.array(a).transpose()

def to_df(array, f, eps, f_type = 'f'):
    np_arr = np.concatenate((array, [f], [eps]), axis=0)
    return pd.DataFrame(data=np_arr, index=["X", "Y", f_type, 'eps'])


def newline(p1, p2, color = 'black'):
    ax = plt.gca()
    xmin, xmax = ax.get_xbound()

    if(p2[0] == p1[0]):
        xmin = xmax = p1[0]
        ymin, ymax = ax.get_ybound()
    else:
        ymax = p1[1]+(p2[1]-p1[1])/(p2[0]-p1[0])*(xmax-p1[0])
        ymin = p1[1]+(p2[1]-p1[1])/(p2[0]-p1[0])*(xmin-p1[0])

    l = mlines.Line2D([xmin,xmax], [ymin,ymax], color = color)
```

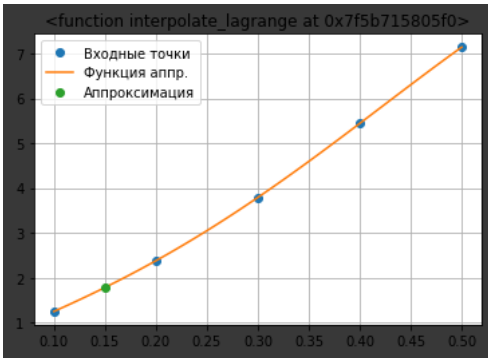
```
    ax.add_line(l)
    return l

def check_and_draw(x, y, approximate_function, point):
    fig, ax = plt.subplots()
    xnew=np.linspace(np.min(x),np.max(x),100)
    ynew=[approximate_function(x,y,i) for i in xnew]
    plt.plot(x,y,'o', label = 'Входные точки')
    plt.plot(xnew,ynew, label = 'Функция аппр.')
    plt.plot(point[0], point[1], '.', markersize=12, label = 'Аппроксимация')
    plt.title(approximate_function)
    ax.legend()
    plt.grid(True)
    plt.show()

def interpolate_lagrange(x, y, x_cur) :
    res = 0.0
    for i in range (0, len(x)):
        p = 1.0
        for j in range (0, len(x)):
            if(i != j):
                p *= (x_cur - x[j])/(x[i]-x[j])
        res += p*y[i]
    return res

def coef(y, n, i):
    if n == 0:
        return (y[i + 1] - y[i])
    return (coef(y, n - 1, i + 1) - coef(y, n - 1, i))

def newton_forward_interpolation(x, y, x_cur):
    i = 0
    n = len(x) - 1
    t = (x_cur - x[i]) / (x[1] - x[0])
    return y[i] + sum(np.prod([t - j for j in range(k)]) / math.factorial(k) *
coef(y, k - 1, i) for k in range(1, n - i + 1))

def newton_backward_interpolation(x, y, x_cur):
    n = len(x) - 1
    t = (x_cur - x[n]) / (x[1] - x[0])
    return y[n] + sum(np.prod([t + j for j in range(k)]) / math.factorial(k) *
coef(y, k - 1, n - k) for k in range(1, n + 1))

def interpolate_newton(x, y, x_cur):
    if x_cur > x[int((x[-1]+x[0])/2)]:
      return newton_backward_interpolation(x, y, x_cur)
    else:
      return newton_forward_interpolation(x, y, x_cur)
```

```
def run():
    again = True
    while again:
        again = False
        in_type = input('Введите:\n\t* k - если вводить с клавиатуры\n\t* f - 
если вводить из файла\n')
        if in_type.strip() == 'k':
            data = input_vals()
        elif in_type.strip() == 'f':
            data = parse()
        else:
            print('Введено неверно, попробуйте снова.')
            again = True
    print(f'Итоговый сет данных:\n{data[0]}\n{data[1]}')

    cur_x = in_float('Число, для которого интерполировать значение: ', check = 
[True, min(data[0]), max(data[0])])

    lagrange_result = interpolate_lagrange(data[0],data[1],cur_x)
    print(f'Lagrange\nОтвет методом Лагранжа: {lagrange_result}')
    check_and_draw(data[0], data[1], interpolate_lagrange, [cur_x, 
lagrange_result])
    print()

    newtone_result = interpolate_newton(data[0],data[1],cur_x)
    print(f'Newtone\nОтвет методом Ньютона: {newtone_result}')
    check_and_draw(data[0], data[1], interpolate_newton, [cur_x, 
newtone_result])


run()
```

Пример работы:

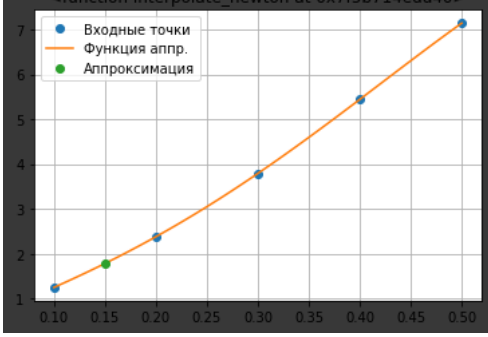
```
Введите:
        * k - если вводить с клавиатуры
        * f - если вводить из файла
f
Путь:
data.txt
Итоговый сет данных:
[0.1 0.2 0.3 0.4 0.5]
[1.25 2.38 3.79 5.44 7.14]
Число, для которого интерполировать значение: : 0.15
Lagrange
Ответ методом Лагранжа: 1.783359375
```

<function interpolate_lagrange at 0x7f5b715805f0>




```
Newtone
Ответ методом Ньютона: 1.7833593750000007
```

<function interpolate_newton at 0x7f5b714edd40>



Вывод:

В результате выполнения лабораторной работы я:

- Познакомился с различными способами решения интерполирования функции, нахождении с помощью численных методов аппр. точек.
- Попрактиковался как в решении задачи на бумаге, так и написании программной реализации.
- Поработал с Python, в особенности с библиотекой отрисовки графиков matplotlib и numpy.