

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

ОТЧЕТ

по лабораторной работе №1

дисциплина: Вычислительные методы

Студент: Яссин Мохамад Аламин
Студенческий билет: 1032205004
Группа: НКНБД-01-20

МОСКВА

2022г.

Содержание

Справка – 3 стр.

Код на Python – 4 стр.

Численные расчеты – 6 стр.

Справка

В лабораторной номер один разбирается задача интерполяции, интерполяционный полином Лагранжа и погрешность интерполяции

Полином Лагранжа

Представим интерполяционную функцию в виде полинома

$$P_n(x) = \sum_{i=0}^n y_i Q_{n,i}(x)$$

где $Q_{n,i}(x)$ - полиномы степени n вида:

$$Q_{n,i}(x) = \prod_{j=0, j \neq i}^n \frac{(x - x_j)}{(x_i - x_j)}$$

Очевидно, что $Q_{n,i}(x)$ принимает значение 1 в точке x_i и 0 в остальных узлах интерполяции.

Следовательно в точке x_i исходный полином принимает значение y_i

Таким образом, построенный полином $P_n(x)$ является интерполяционным полиномом для функции $y = f(x)$ на сетке X .

Ход работы

Мой вариант:

18.	$\sqrt{x} - x$	$x \in [0, 2]$
-----	----------------	----------------

Код на питоне:

- 1- первый раздел кода состоит из 2 частей, переменных частей и функции Лагранжа.

```
1  import math
2
3  # variables
4  n = 10
5  m = n * 3
6  a = 0
7  b = 2
8
9
10 # Lagrange's function
11 def lagrange(x, y, n, xx):
12     l = 0.0
13     for i in range(n):
14         q = 1.0
15         for j in range(n):
16             if i != j:
17                 q = q * (xx - x[j]) / (x[i] - x[j])
18             l = l + y[i] * q
19     return l
20
```

- 2- В то время как во втором разделе мы заполняем массивы x и y элементами на основе «n» в первой и второй задаче, а в четвертом на основе переменной «m», в то же время мы заполняем массив «lag» с ответами от функции Лагранжа.

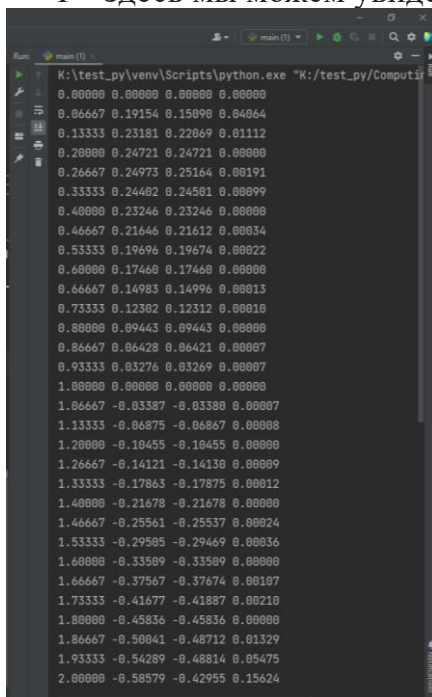
```
22 # first & second task
23 x_values = []
24 for i in range(n + 1):
25     x_values.append(round((0.2 * i), 1))
26
27 y_values = []
28 for i in range(n + 1):
29     y_values.append(round(math.sqrt(x_values[i]) - x_values[i], 5))
30
31 # third task
32 lag = []
33 for i in range(n + 1):
34     lag.append(lagrange(x_values, y_values, n + 1, x_values[i]))
35
36 # fourth task
37 xm_values = []
38 for i in range(m + 1):
39     xm_values.append(round(((2 / m) * i), 5))
```

3- Здесь, в последнем разделе, мы видим применение последних двух задач, вычисление диапазона задержки и дельты в пятой и печать в правильной форме, а в шестой мы проверяем, какие поля значений «n» будут подходит для наших условий.

```
41 # fifth task
42 ym_values = []
43 lag_m = []
44 delta = []
45 for i in range(m + 1):
46     ym_values.append(round(math.sqrt(xm_values[i]) - xm_values[i], 5))
47     lag_m.append(lagrange(x_values, y_values, n, xm_values[i]))
48     delta.append(abs(lag_m[i] - ym_values[i]))
49
50     print(format(xm_values[i], '.5f'), format(ym_values[i], '.5f'), format(lag_m[i], '.5f'), format(delta[i], '.5f'))
51
52     print("\n\n\n\n")
53
54 # sixth task
55 for i in range(m + 1):
56     if delta[i] <= 0.01:
57         print(x_values[i])
58         print(format(xm_values[i], '.5f'), format(ym_values[i], '.5f'), format(lag_m[i], '.5f'),
59               format(delta[i], '.5f'))
60
```

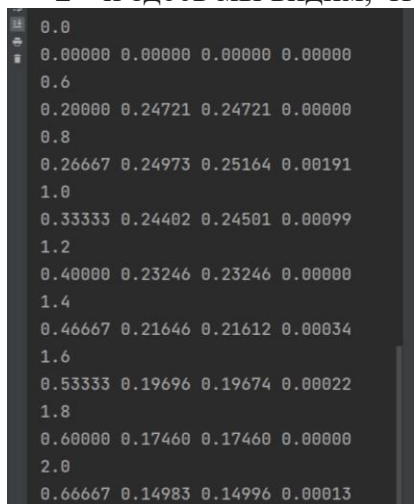
Вывод.

1- Здесь мы можем увидеть результаты для «m»



```
0.00000 0.00000 0.00000 0.00000
0.06667 0.19154 0.15090 0.04064
0.13333 0.23181 0.22069 0.01112
0.20000 0.24721 0.24721 0.00000
0.26667 0.24973 0.25164 0.00191
0.33333 0.24402 0.24501 0.00099
0.40000 0.23246 0.23246 0.00000
0.46667 0.21646 0.21612 0.00034
0.53333 0.19696 0.19674 0.00022
0.60000 0.17460 0.17460 0.00000
0.66667 0.14983 0.14996 0.00013
0.73333 0.12302 0.12312 0.00010
0.80000 0.09443 0.09443 0.00000
0.86667 0.06428 0.06421 0.00007
0.93333 0.03276 0.03269 0.00007
1.00000 0.00000 0.00000 0.00000
1.06667 -0.03387 -0.03380 0.00007
1.13333 -0.06075 -0.06067 0.00008
1.20000 -0.10455 -0.10455 0.00000
1.26667 -0.14121 -0.14130 0.00009
1.33333 -0.17863 -0.17875 0.00012
1.40000 -0.21678 -0.21678 0.00000
1.46667 -0.25561 -0.25537 0.00024
1.53333 -0.29505 -0.29469 0.00036
1.60000 -0.33509 -0.33509 0.00000
1.66667 -0.37567 -0.37674 0.00107
1.73333 -0.41677 -0.41887 0.00210
1.80000 -0.45836 -0.45836 0.00000
1.86667 -0.50041 -0.48712 0.01329
1.93333 -0.54289 -0.48814 0.05475
2.00000 -0.58579 -0.42955 0.15624
```

2- и здесь мы видим, что условие будет выполнено, когда $2.0 \geq n \geq 0.6$



A screenshot of a terminal window with a dark background and light gray text. The terminal displays a table of numerical data. The first column contains values from 0.0 to 2.0 in increments of 0.2. The subsequent three columns contain floating-point numbers. The data is as follows:

0.0	0.00000	0.00000	0.00000	0.00000
0.6	0.20000	0.24721	0.24721	0.00000
0.8	0.26667	0.24973	0.25164	0.00191
1.0	0.33333	0.24402	0.24501	0.00099
1.2	0.40000	0.23246	0.23246	0.00000
1.4	0.46667	0.21646	0.21612	0.00034
1.6	0.53333	0.19696	0.19674	0.00022
1.8	0.60000	0.17460	0.17460	0.00000
2.0	0.66667	0.14983	0.14996	0.00013