

УНИВЕРСИТЕТ ИТМО

ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА

Лабораторная работа №5

ФИО студента: Готовко Алексей Владимирович
Направление подготовки: 09.03.04 (СППО)
Учебная группа: Р32101
ФИО преподавателей:
Мальшева Т.А.
Рыбаков С.Д.

Санкт-Петербург
2023г.

1 Цель работы

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

2 Вычислительная реализация задачи

Точки по варианту:

x	1.10	1.25	1.40	1.55	1.70	1.85	2.00
y	0.2234	1.2438	2.2644	3.2984	4.3222	5.3516	6.3867

$$X_1 = 1.121, X_2 = 1.482$$

2.1 Таблица конечных разностей

	y_i	Δy_i	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^4 y_i$	$\Delta^5 y_i$	$\Delta^6 y_i$
$x_0(x_{-3})$	0.2234	1.0204	0.0002	0.0132	-0.0368	0.2562	-1.2113
$x_1(x_{-2})$	1.2438	1.0206	0.0134	-0.0236	0.2194	-0.9551	
$x_2(x_{-1})$	2.2644	1.034	-0.0102	0.1958	-0.7357		
$x_3(x_0)$	3.2984	1.0238	0.1856	-0.5399			
$x_4(x_1)$	4.3222	1.2094	-0.3543				
$x_5(x_2)$	5.5316	0.8551					
$x_6(x_3)$	6.3867						

2.2 Интерполяция по Ньютону

Так как $X_1 = 1.121 < \frac{(x_6+x_0)}{2} = 1.55$, то будет интерполировать вперед.

$$t = \frac{X_1 - x_0}{h} = \frac{1.121 - 1.10}{0.15} = 0.14$$

$$\begin{aligned} N_6(X_1) = y_0 + t\Delta y_0 + \frac{t(t-1)}{2}\Delta^2 y_0 + \frac{t(t-1)(t-2)}{3!}\Delta^3 y_0 + \frac{t(t-1)(t-2)(t-3)}{4!}\Delta^4 y_0 + \\ + \frac{t(t-1)(t-2)(t-3)(t-4)}{5!}\Delta^5 y_0 + \frac{t(t-1)(t-2)(t-3)(t-4)(t-5)}{6!}\Delta^6 y_0 \approx 0.3932 \end{aligned}$$

2.3 Интерполяция по Гауссу

Так как $X_2 = 1.482 < \frac{(x_6+x_0)}{2} = 1.55$, то будет использовать вторую интерполяцию Гаусса.

$$t = \frac{X_2 - x_0}{h} = \frac{X_2 - a}{h} = \frac{1.482 - 1.55}{0.15} \approx -0.4533$$

$$\begin{aligned} P_6(X_2) = y_0 + t\Delta y_{-1} + \frac{t(t+1)}{2}\Delta^2 y_{-1} + \frac{(t+1)t(t-1)}{3!}\Delta^3 y_{-2} + \frac{(t+2)(t+1)t(t-1)}{4!}\Delta^4 y_{-2} + \\ + \frac{(t+2)(t+1)t(t-1)(t-2)}{5!}\Delta^5 y_{-3} + \frac{(t+3)(t+2)(t+1)t(t-1)(t-2)}{6!}\Delta^6 y_{-3} \approx 2.8306 \end{aligned}$$

3 Программная реализация задачи

```
1  # interpolation_methods.py
2  from math import factorial
3
4  from interpolation_util import InterpolationData
5  from io_util import print_warn
6
7
8  def lagrange(data: InterpolationData, target_x):
9      points = data.point_arr
10     n = data.points_cnt
11
12     val = 0
13     for i in range(n):
14         multiplier = 1
15         for j in range(n):
16             if i == j:
17                 continue
18             multiplier *= (target_x - points[j].x) / (points[i].x - points[j].x)
19         val += multiplier * points[i].y
20
21     return val
22
23
24  def newton(data: InterpolationData, target_x):
25      t: float
26      is_backwards: bool
27      if target_x < (data.max_x + data.min_x) / 2:
28          is_backwards = False
29          t = (target_x - data.min_x) / data.x_step
30      else:
31          is_backwards = True
32          t = (target_x - data.max_x) / data.x_step
33
34      val = 0
35      t_multiplier = 1
36      diff = data.differences
37
38      idx = -1 if is_backwards else 0
39      t_delta = 1 if is_backwards else -1
40      for i in range(data.points_cnt):
41          val += diff[i][idx] * t_multiplier
42          t_multiplier = t_multiplier * (t + t_delta * i) / (i + 1)
43
44      return val
45
46
47  def stirling(data: InterpolationData, target_x):
48      n = data.points_cnt // 2
49      t = (target_x - data.point_arr[n].x) / data.x_step
50
51      if abs(t) > 0.25:
52          print_warn(f"|t| = {round(abs(t), 4)} > 0.25, Stirling's interpolation may work improperly")
53
54      diff = data.differences
55
56      val = diff[0][n] + t * (diff[1][n] + diff[1][n - 1]) / 2 + t ** 2 / 2 * diff[2][n - 1]
57      t_multiplier = t * (t ** 2 - 1) / 6
58      iter_odd = True
```

```

59
60     i = 3
61     for j in range(1, n):
62         if iter_odd:
63             dy = (diff[i][n - j - 1] + diff[i][n - j - 2]) / 2
64         else:
65             dy = diff[i][n - j - 1]
66
67         val += dy * t_multiplier
68         if iter_odd:
69             t_multiplier *= t / i
70         else:
71             t_multiplier *= t ** 2 - (i + 1) ** 2 / i
72         iter_odd = not iter_odd
73         i += 1
74
75     return val
76
77
78 def bessel_calc(u, n):
79     if n == 0:
80         return 1
81
82     var = u
83     for i in range(1, n // 2 + 1):
84         var *= u - i
85
86     for i in range(1, n // 2):
87         var *= u + i
88
89     return var
90
91
92 def bessel(data: InterpolationData, target_x):
93     t = (target_x - data.point_arr[data.points_cnt // 2].x) / data.x_step
94
95     if not 0.25 <= abs(t) <= 0.75:
96         print_warn(f"|t| = {round(abs(t), 4)} not in [0.25, 0.75], "
97                   "Bessel's interpolation may work improperly")
98
99     n = data.points_cnt
100    x = [p.x for p in data.point_arr]
101
102    y = [[0 for _ in range(n)] for _ in range(n)]
103
104    for i in range(n):
105        y[i][0] = data.point_arr[i].y
106
107    for i in range(1, n):
108        for j in range(n - i):
109            y[j][i] = y[j + 1][i - 1] - y[j][i - 1]
110
111    val = (y[2][0] + y[3][0]) / 2
112
113    if n % 2 == 1:
114        k = n // 2
115    else:
116        k = n // 2 - 1
117
118    u = (target_x - x[k]) / (x[1] - x[0])
119

```

```

120     for i in range(1, n):
121         if i % 2 == 1:
122             val += ((u - 0.5) * bessell_calc(u, i - 1) * y[k][i]) / factorial(i)
123         else:
124             val += (bessell_calc(u, i) * (y[k][i] + y[k - 1][i]) / (factorial(i) * 2))
125             k -= 1
126
127     return val
128

```

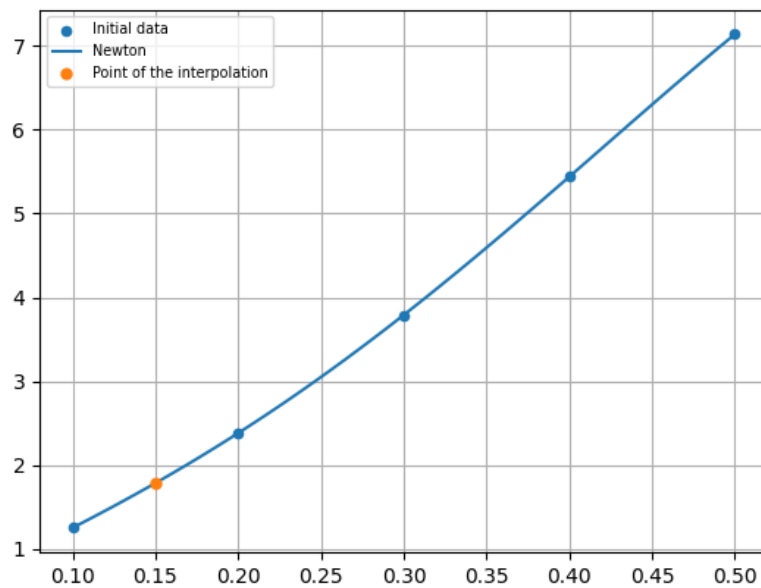
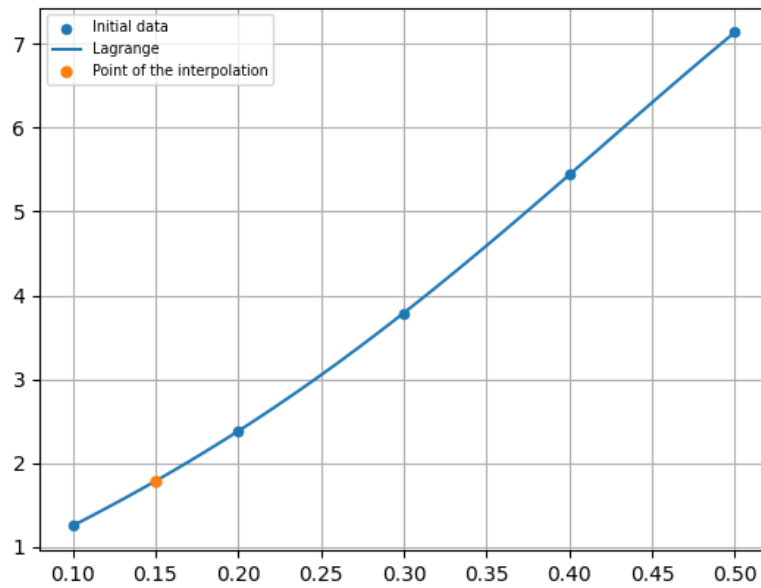
Полный код программы доступен по [ссылке](#).

4 Результат работы программы

```

1  # Input:
2  # X      Y
3  # 0.1    1.25
4  # 0.2    2.38
5  # 0.3    3.79
6  # 0.4    5.44
7  # 0.5    7.14
8
9  Finite differences
10
11  -----
12  | i |      yi      |      dyi      |      d2yi      |      d3yi      |      d4yi      |
13  |---|-----|-----|-----|-----|
14  | 0 |      1.250    |      1.130    |      0.280    |      -0.040    |      -0.150    |
15  | 1 |      2.380    |      1.410    |      0.240    |      -0.190    |
16  | 2 |      3.790    |      1.650    |      0.050    |
17  | 3 |      5.440    |      1.700    |
18  | 4 |      7.140    |
19
20  |t| = 1.5 > 0.25, Stirling's interpolation may work improperly
21  |t| = 1.5 not in [0.25, 0.75], Bessel's interpolation may work improperly
22  Newton's method      :      1.7834
23  Lagrange's method    :      1.7834
24  Stirling's method     :      1.8009
25  Bessel's method      :      1.8039
26

```



5 Вывод

Для наиболее оптимального распределения человеческих ресурсов и поддержания количества нервных клеток в головном мозгу, настоятельно рекомендуется использовать готовые библиотеки, содержащие наиболее эффективные реализации алгоритмов, вместо самостоятельной реализации оных.

Иными словами, вообще лучше зачлнить и не изобретать велосипед.