

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет ПИ и КТ

Лабораторная работа №5

по дисциплине: «Вычислительная математика»

«Интерполяция функции»

Вариант 1

Выполнил:

**Болорболд Аригуун**,

группа P3211

Преподаватель:

**Малышева Татьяна Алексеевна**

Санкт-Петербург

2024



1. **Цель работы:**

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

1. **Вычислительная реализация задачи.**

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

Таблица конечных разностей:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *x* | 0,25 | 0,30 | 0,35 | 0,40 | 0,45 | 0,50 | 0,55 |
| *y* | 1,2557 | 2,1764 | 3,1218 | 4,0482 | 5,9875 | 6,9195 | 7,8359 |
| *Δyi* | 0,9207 | 0,9454 | 0,9264 | 1,9393 | 0,9320 | 0,9164 |  |
| *Δ2yi* | 0,0247 | –0,0190 | 1,0129 | –1,0073 | –0,0156 |  |  |
| *Δ3yi* | –0,0437 | 1,0319 | –2,0202 | 0,9917 |  |  |  |
| *Δ4yi* | 1,0756 | –3,0521 | 3,0119 |  |  |  |  |
| *Δ5yi* | –4,1277 | 6,064 |  |  |  |  |  |
| *Δ6yi* | 10,1917 |  |  |  |  |  |  |

будем применять первую интерполяционную формулу Ньютона.

находится левее середины отрезка.

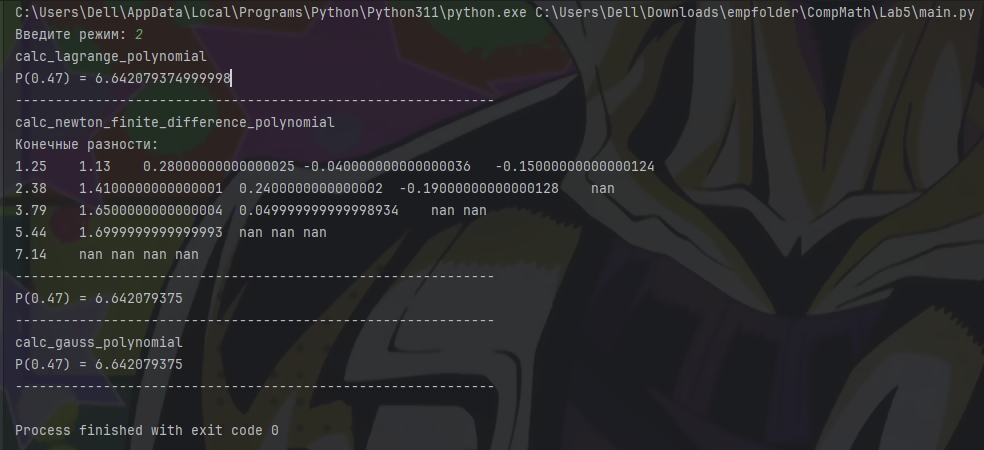
находится левее середины отрезка.

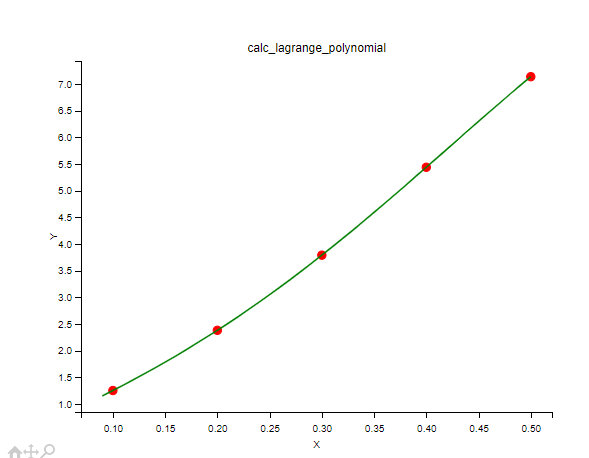
Таблица конечных разностей:

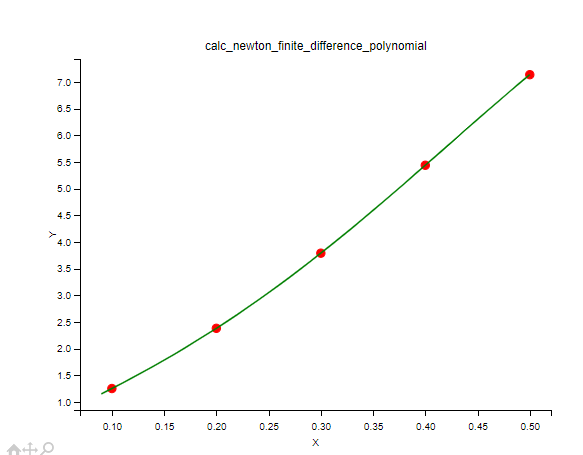
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *i* | –3 | –2 | –1 | 0 | 1 | 2 | 3 |
| *x* | 0,25 | 0,30 | 0,35 | 0,40 | 0,45 | 0,50 | 0,55 |
| *y* | 1,2557 | 2,1764 | 3,1218 | 4,0482 | 5,9875 | 6,9195 | 7,8359 |
| *Δyi* | 0,9207 | 0,9454 | 0,9264 | 1,9393 | 0,9320 | 0,9164 |  |
| *Δ2yi* | 0,0247 | –0,0190 | 1,0129 | –1,0073 | –0,0156 |  |  |
| *Δ3yi* | –0,0437 | 1,0319 | –2,0202 | 0,9917 |  |  |  |
| *Δ4yi* | 1,0756 | –3,0521 | 3,0119 |  |  |  |  |
| *Δ5yi* | –4,1277 | 6,064 |  |  |  |  |  |
| *Δ6yi* | 10,1917 |  |  |  |  |  |  |

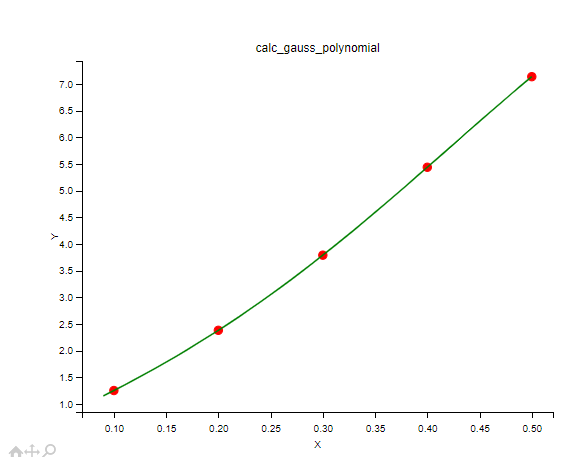
1. **Программная реализация задачи.**
2. from functools import reduce  
   from math import factorial  
   import pandas as pd  
   from matplotlib import pyplot as plt  
     
     
   def calc\_lagrange\_polynomial(xs, ys):  
    n = len(xs) - 1  
    f = lambda x: sum([ys[i] \*  
    reduce(lambda a, b: a \* b,  
    [(x - xs[j]) / (xs[i] - xs[j])  
    for j in range(n + 1) if i != j])  
    for i in range(n + 1)])  
    return f  
     
   def calc\_newton\_finite\_difference\_polynomial(xs, ys, x):  
    median = (xs[len(xs) // 2] + xs[len(xs) // 2 + 1]) / 2  
    fin\_difs = [ys[:]]  
    n = len(xs) - 1  
    for k in range(1, n + 1):  
    last = fin\_difs[-1][:]  
    fin\_difs.append(  
    [last[i + 1] - last[i] for i in range(n - k + 1)])  
    print("Конечные разности:")  
    df = pd.DataFrame(fin\_difs)  
    df = df.transpose()  
    for i in range(len(df)):  
    print(\*df.iloc[i].values, sep='\t')  
    print('-' \* 60)  
    h = xs[1] - xs[0]  
    if x < median:  
    f = lambda x: ys[0] + sum([  
    reduce(lambda a, b: a \* b,  
    [(x - xs[0]) / h - j for j in range(k)])  
    \* fin\_difs[k][0] / factorial(k)  
    for k in range(1, n + 1)])  
    else:  
    f = lambda x: ys[n] + sum([  
    reduce(lambda a, b: a \* b,  
    [(x - xs[n]) / h + j for j in range(0, n - k + 1)])  
    \* fin\_difs[n - k + 1][k - 1] / factorial(n - k + 1)  
    for k in range(n, 0, -1)])  
    return f  
     
   def calc\_gauss\_polynomial(xs, ys, x):  
    n = len(xs) - 1  
    alpha\_ind = n // 2  
    fin\_difs = []  
    fin\_difs.append(ys[:])  
    for k in range(1, n + 1):  
    last = fin\_difs[-1][:]  
    fin\_difs.append(  
    [last[i + 1] - last[i] for i in range(n - k + 1)])  
    h = xs[1] - xs[0]  
    dts1 = [0, -1, 1, -2, 2, -3, 3, -4, 4]  
    f1 = lambda x: ys[alpha\_ind] + sum([  
    reduce(lambda a, b: a \* b,  
    [(x - xs[alpha\_ind]) / h + dts1[j] for j in range(k)])  
    \* fin\_difs[k][len(fin\_difs[k]) // 2] / factorial(k)  
    for k in range(1, n + 1)])  
    f2 = lambda x: ys[alpha\_ind] + sum([  
    reduce(lambda a, b: a \* b,  
    [(x - xs[alpha\_ind]) / h - dts1[j] for j in range(k)])  
    \* fin\_difs[k][len(fin\_difs[k]) // 2 - (1 - len(fin\_difs[k]) % 2)] / factorial(k)  
    for k in range(1, n + 1)])  
    return lambda x: f1(x) if x > xs[alpha\_ind] else f2(x)  
     
   def draw\_plot(a, b, func, dx=0.01):  
    xs, ys = [], []  
    a -= dx  
    b += dx  
    x = a  
    while x <= b:  
    xs.append(x)  
    ys.append(func(x))  
    x += dx  
    plt.plot(xs, ys, 'g')  
     
     
   def main(xs, ys, x):  
    methods = [calc\_lagrange\_polynomial,  
    calc\_newton\_finite\_difference\_polynomial,  
    calc\_gauss\_polynomial]  
    for method in methods:  
    if method is calc\_gauss\_polynomial and len(xs) % 2 == 0:  
    continue  
    print(method.\_\_name\_\_)  
    if method == calc\_lagrange\_polynomial:  
    P = method(xs, ys)  
    else:  
    P = method(xs, ys, x)  
    plt.title(method.\_\_name\_\_)  
    draw\_plot(xs[0], xs[-1], P)  
    for i in range(len(xs)):  
    plt.scatter(xs[i], ys[i], c='r')  
    plt.xlabel("X")  
    plt.ylabel("Y")  
    plt.show()  
    print(f'P({x}) = {P(x)}')  
    print('-' \* 60)  
     
   def read\_number(s: str):  
    while True:  
    try:  
    return float(input(s))  
    except Exception:  
    continue  
     
   if \_\_name\_\_ == '\_\_main\_\_':  
    mode = read\_number("Введите режим: ")  
    while mode not in (1, 2, 3):  
    mode = read\_number("Введите режим: ")  
    if mode == 1:  
    xs = list(map(float, input('Введите иксы: ').split()))  
    ys = list(map(float, input('Введите игреки: ').split()))  
    x = float(input('Введите икс: '))  
    elif mode == 2:  
    with open('tests/1') as f:  
    xs = list(map(float, f.readline().strip().split()))  
    ys = list(map(float, f.readline().strip().split()))  
    x = float(f.readline().strip())  
    elif mode == 3:  
    print('Функции: ')  
    print('1. x ^ 2 - 3 \* x')  
    print('2. x ^ 5')  
    func\_number = read\_number("Выберите функцию")  
    f = lambda x: x \*\* 2 - 3 \* x if func\_number == 1 else x \*\* 5  
    n = int(input('Введите n: '))  
    x0 = float(input('Введите первый x: '))  
    xn = float(input('Введите последний x: '))  
    h = (xn - x0) / (n - 1)  
    xs = [x0 + h \* i for i in range(n)]  
    ys = list(map(f, xs))  
    x = float(input('Введите икс: '))  
    xs = sorted(xs)  
    if len(set(xs)) != len(xs):  
    print('Иксы должны быть разными')  
    else:  
    main(xs, ys, x)
3. **Тестовые данные.**

0.1 0.2 0.3 0.4 0.5  
1.25 2.38 3.79 5.44 7.14  
0.47

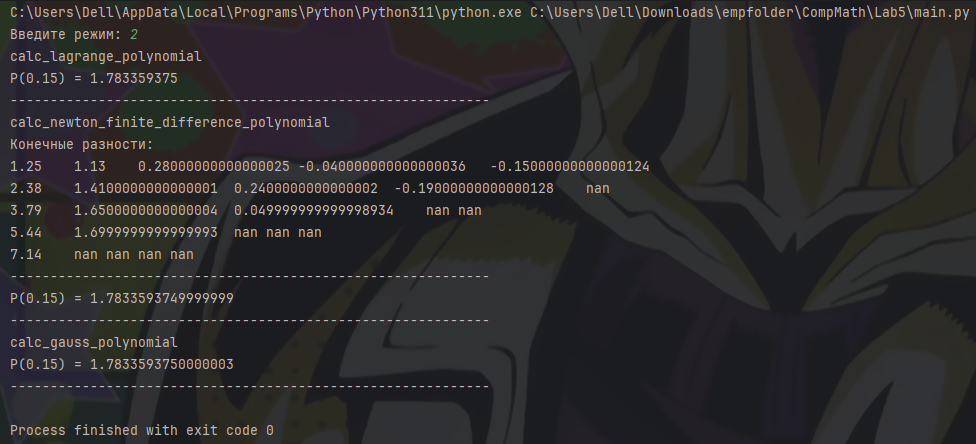


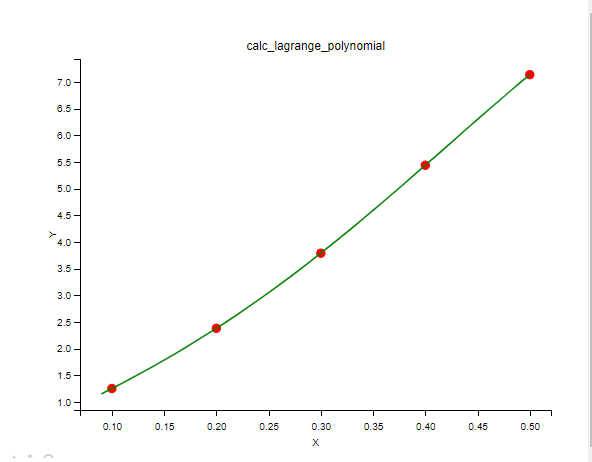


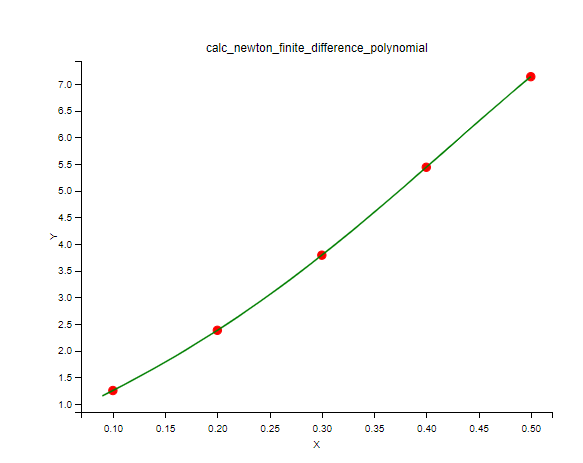


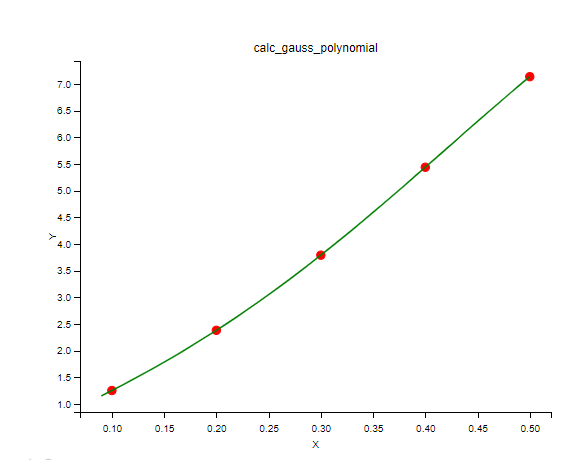


0.1 0.2 0.3 0.4 0.5  
1.25 2.38 3.79 5.44 7.14  
0.15









1. **Выводы.**

В процессе выполнения лабораторной работы я познакомился (точнее изжил) с интерполяцией функции разными методами (Лагранжа, Ньютона, Гаусса, Стирлинга и Бесселя).

Линейная и квадратичная интерполяция — простые, но довольно неточные методы.

Многочлен Лагранжа — хороший метод, но вычислений много (молчу про случая когда количество узлов поменяется). Малая погрешность при .

Многочлен Ньютона (с разделёнными разностями) — хороший метод. Используется для неравноотстоящих узлов. При добавлении новых узлов первые члены остаются неизменными.

Многочлен Ньютона (с конечными разностями) — тоже хороший метод, является особым случаем при равноотстоящих узлов. Существуют формулы для интерполирования вперёд и назад, можно использовать для экстраполирования (но надо учитывать погрешность).