**Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»
Факультет Программной Инженерии и Компьютерной Техники**

Вариант №9
Лабораторная работа №5
по теме
Интерполяция функции
по дисциплине
Вычислительная математика

Выполнил Студент группы P3212
Кобелев Р.П.
к. т. н. Преподаватель:
Наумова Н.А.

г. Санкт-Петербург
2024г.

Содержание

1 Цель работы

Решить задачу интерполяции, найти значения функ- ции при заданных значениях аргумента, отличных от узловых точек.

2 Порядок выполнения работ

2.1 Обязательное задание (до 80 баллов)

2.1.1 Вычислительная реализация задачи:

1. Выбрать из табл. 1 заданную по варианту таблицу $y = f(x)$ (таблица 1.1);

2. Построить таблицу конечных разностей для заданной таблицы. Таблицу отразить в отчете;

3. Вычислить значения функции для аргумента X_1 (см. табл. 1.1), используя первую или вторую интерполяционную формулу Ньютона. Обратить внимание какой конкретно формулой необходимо воспользоваться;

4. Вычислить значения функции для аргумента X_2 (см. табл. 1.1), используя первую или вторую интерполяционную формулу Гаусса. Обратить внимание какой конкретно формулой необходимо воспользоваться;

5. Подробные вычисления привести в отчете.

2.1.2 Программная реализация задачи:

1. Исходные данные задаются тремя способами:

2. (a) в виде набора данных (таблицы x, y), пользователь вводит значения с клавиатуры;
   (b) в виде сформированных в файле данных (подготовить не менее трех тестовых вариантов);
   (c) на основе выбранной функции, из тех, которые предлагает программа, например, $\sin x$. Пользователь выбирает уравнение, исследуемый интервал и количество точек на интервале (не менее двух функций).

3. Сформировать и вывести таблицу конечных разностей;

4. Вычислить приближенное значение функции для заданного значения аргумента, введенного с клавиатуры, указанными методами (см. табл. 2). Сравнить полученные значения;

5. Построить графики заданной функции с отмеченными узлами интерполяции и интерполяционного многочлена Ньютона/Гаусса (разными цветами);

6. Программа должна быть протестирована на различных наборах данных, в том числе и некорректных.

7. Проанализировать результаты работы программы.

2.2 Необязательное задание (до 20 баллов)

1. Реализовать в программе вычисление значения функции для заданного значения аргумента, введенного с клавиатуры, используя схемы Стирлинга;

2. Реализовать в программе вычисление значения функции для заданного значения аргумента, введенного с клавиатуры, используя схемы Бесселя.

3 Вычислительная реализация задачи

3.1 Основные данные

x	y
1.05	0.1213
1.15	1.1316
1.25	2.1459
1.35	3.1565
1.45	4.1571
1.55	5.1819
1.65	6.1969

X_1	X_2
1.562	1.362

3.2 Вычисление таблицы разностей

Для первого ряда мы имеем формулу:

$$\Delta y_i = y_{i+1} - y_i$$

Вычислим значения для 1 ряда:
$\Delta y_1 = 1.1316 - 0.1213 = 1.0103$
$\Delta y_2 = 2.1459 - 1.1316 = 1.0143$
$\Delta y_3 = 3.1565 - 2.1459 = 1.0106$
$\Delta y_4 = 4.1571 - 3.1565 = 1.0006$
$\Delta y_5 = 5.1819 - 4.1571 = 1.0248$
$\Delta y_6 = 6.1969 - 5.1819 = 1.015$
В следующем у нас зануляется нижний элемент и дальше используем формулу:

$$\Delta^2 y_i = \Delta y_{i+1} - \Delta y_i$$

Вычислим значения для 2 ряда:
$\Delta^2 y_1 = 1.0143 - 1.0103 = 0.004$
$\Delta^2 y_2 = 1.0106 - 1.0143 = -0.0037$
$\Delta^2 y_3 = 1.0006 - 1.0106 = -0.01$
$\Delta^2 y_4 = 1.0248 - 1.0006 = 0.0242$
$\Delta^2 y_5 = 1.015 - 1.0248 = -0.0098$
И так далее для следующих рядов - мы получиили таблицу разностей:

y	Δy	$\Delta^2 y$	$\Delta^3 y$	$\Delta^4 y$	$\Delta^5 y$	$\Delta^6 y$
0.1213	1.01	0.004	−0.0077	0.0014	0.0391	−0.1478
1.132	1.014	−0.0037	−0.0063	0.0405	−0.1087	0.0
2.146	1.011	−0.01	0.0342	−0.0682	0.0	0.0
3.156	1.001	0.0242	−0.034	0.0	0.0	0.0
4.157	1.025	−0.0098	0.0	0.0	0.0	0.0
5.182	1.015	0.0	0.0	0.0	0.0	0.0
6.197	0.0	0.0	0.0	0.0	0.0	0.0

3.3 Вычисление методом Ньютона для $X_1 = 1.562$

Смотря на значение x можно прийти к выводу, что для вычисления мы используем формулу Ньютона для интерполирования назад, так как X_1 лежит во второй половине.

$$t = \frac{x - x_n}{h} = \frac{1.562 - 1.65}{0.1} = -0.88$$

$$N_5(x) = y_5 + t\Delta y_4 + \frac{t(t+1)}{2!}\Delta^2 y_3 + \frac{t(t+1)(t+2)}{3!}\Delta^3 y_2 +$$

$$\frac{t(t+1)(t+2)(t+3)}{4!}\Delta^4 y_1 + \frac{t(t+1)(t+2)(t+3)(t+4)}{5!}\Delta^5 y_0$$

$$N_5(1.562) = 5.182 + -0.88 \cdot 1.025 + \frac{-0.88(-0.88+1)}{2!}0.0242 + \frac{-0.88(-0.88+1)(-0.88+2)}{3!}0.0342+$$

$$\frac{-0.88(-0.88+1)(-0.88+2)(-0.88+3)}{4!}0.0405+\frac{-0.88(-0.88+1)(-0.88+2)(-0.88+3)(-0.88+4)}{5!}0.0391 = 4.2773698$$

3.4 Вычисление методом Гаусса для $X_2 = 1.362$

$$t = \frac{x - x_0}{h} = \frac{1.362 - 1.35}{0.1} = 0.12$$

$$P_3(x) = y_0 + t\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_-1 + \frac{(t+1)t(t-1)}{3!}\Delta^3 y_-1 + \frac{(t+1)t(t-1)(t-2)}{4!}\Delta^4 y_-2+$$

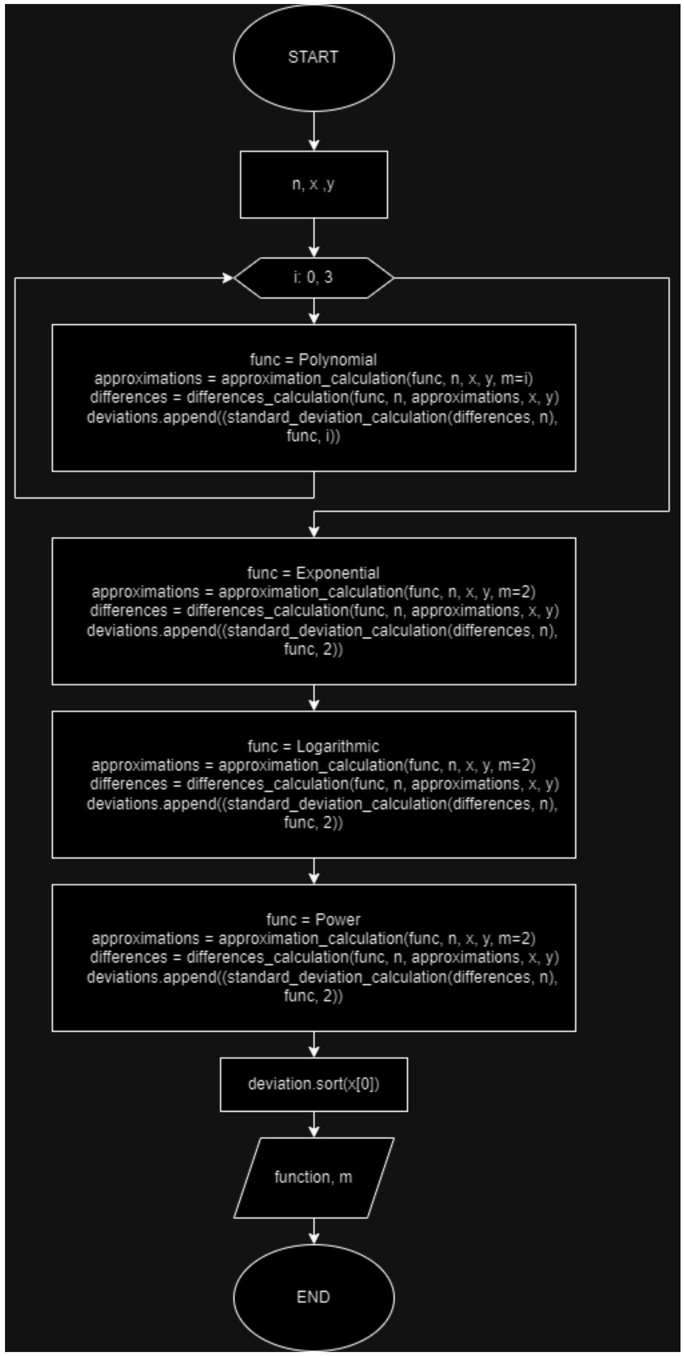
$$\frac{(t+2)(t+1)t(t-1)(t-2)}{5!}\Delta^5 y_-2 + \frac{(t+2)(t+1)t(t-1)(t-2)(t-3)}{6!}\Delta^6 y_-3$$

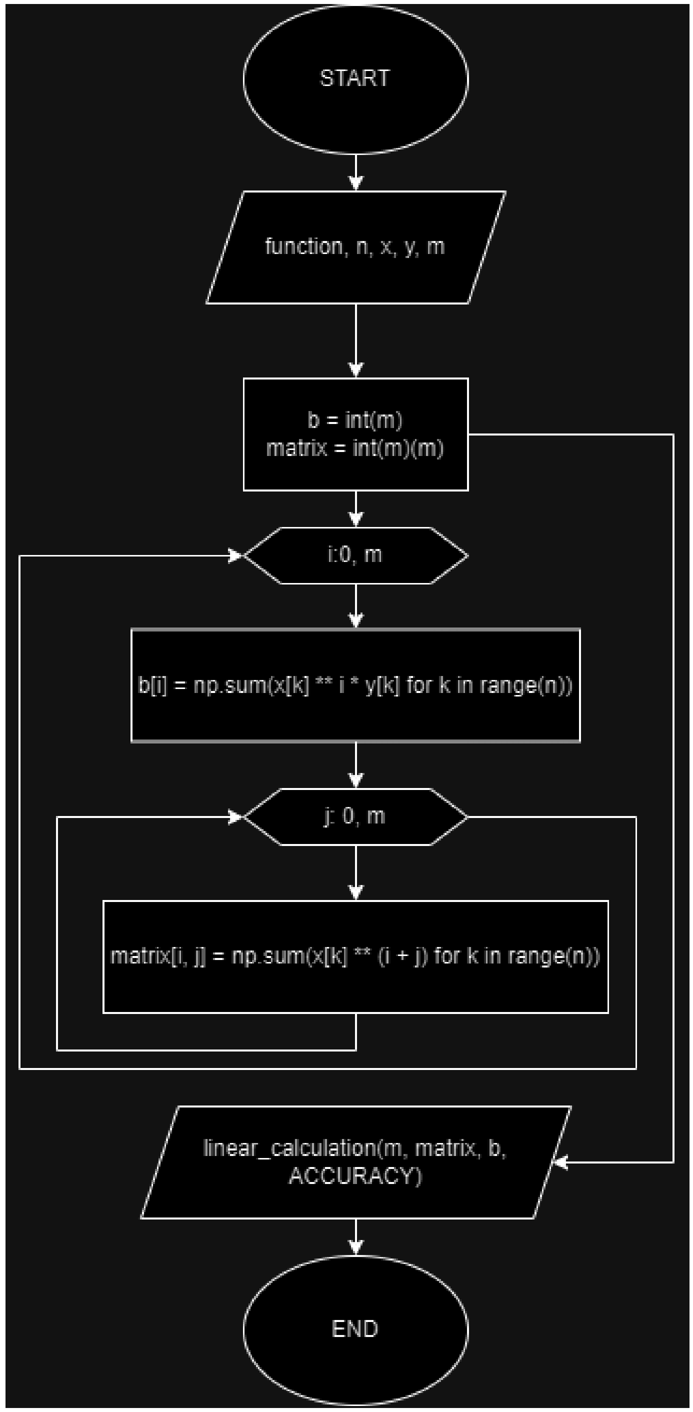
$$P_3(1.362) = 3.156 + (0.12) \cdot 1.001 + \frac{(0.12)(0.12-1)}{2} \cdot -0.01 + \frac{(0.12+1)(0.12)(0.12-1)}{6} \cdot 0.0342+$$

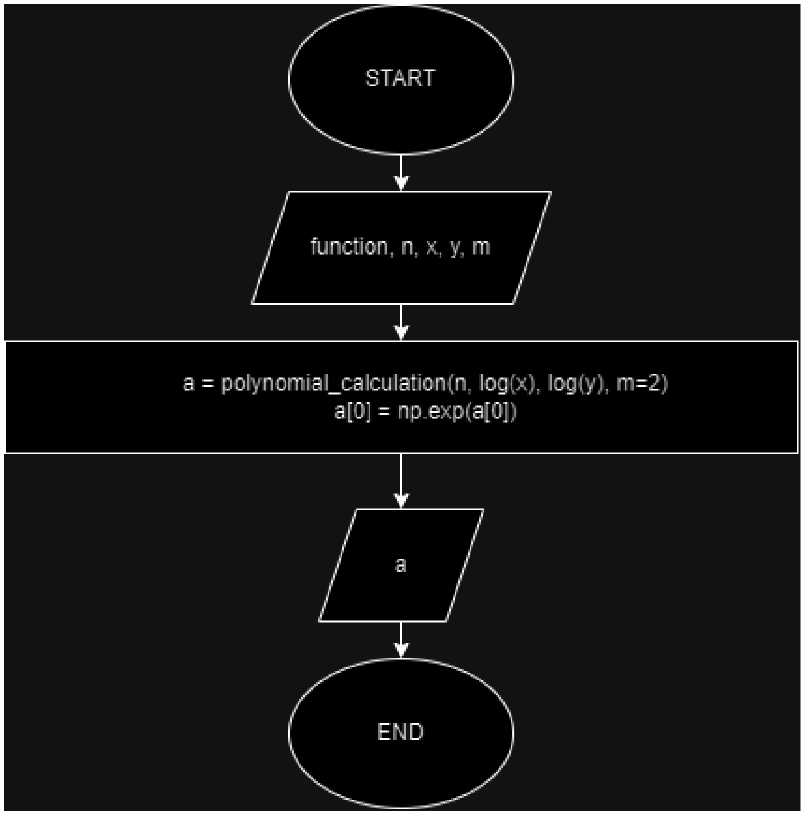
$$\frac{(0.12+1)0.12(0.12-1)(0.12-2)}{24} \cdot 0.0405 + \frac{(0.12+2)(0.12+1)0.12(0.12-1)(0.12-2)}{120} \cdot -0.1087+$$

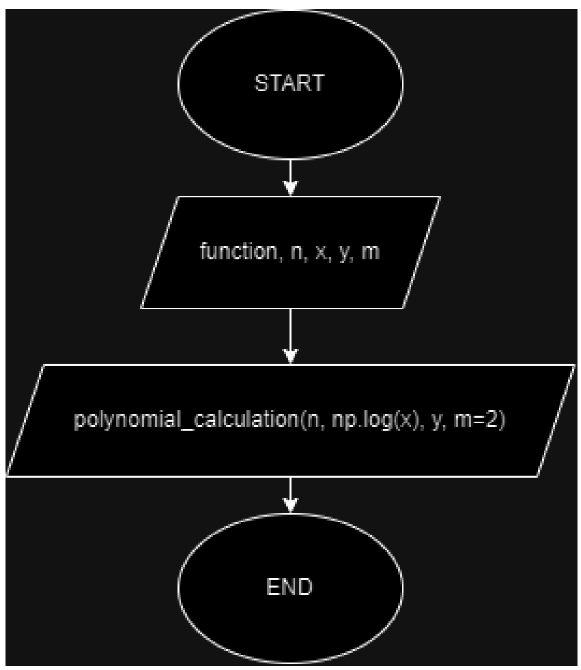
$$\frac{(0.12+2)(0.12+1)0.12(0.12-1)(0.12-2)(0.12-3)}{720} \cdot -0.1478 = 3.2762$$

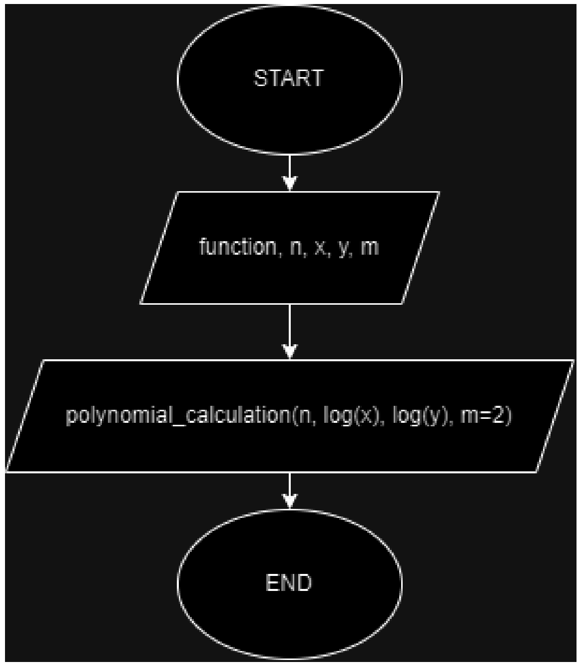
4 Программная реализация задачи




START
function, n, x, y, m
b = int(m)
matrix = int(m)(m)
i:0, m
b[i] = np.sum(x[k] ** i * y[k] for k in range(n))
j: 0, m
matrix[i, j] = np.sum(x[k] ** (i + j) for k in range(n))
linear_calculation(m, matrix, b, ACCURACY)
END


START
function, n, x, y, m
a = polynomial_calculation(n, log(x), log(y), m=2)
a[0] = np.exp(a[0])
a
END


START
function, n, x, y, m
polynomial_calculation(n, np.log(x), y, m=2)
END



4.1 Листинг программы

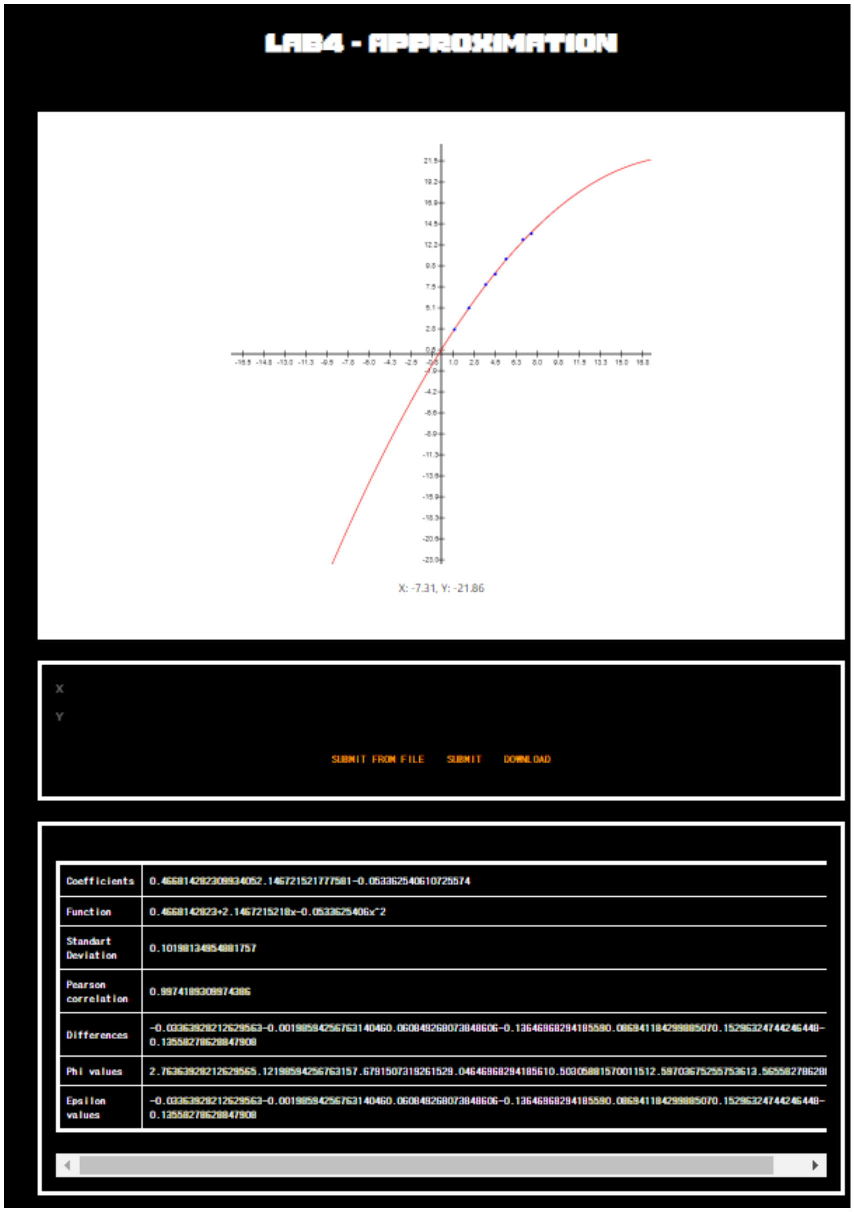
Interpolation.py

```
from dataclasses import dataclass
import numpy as np
from math import factorial

@dataclass
class Interpolation:
  n: int
  x: []
  y: []


  def difference_table(self):
    n = len(self.y)
    defy = [[0] * n for _ in range(n)]

    for i in range(n):
      defy[i][0] = self.y[i]

    for i in range(1, n):
      for j in range(n - i):
          defy[j][i] = defy[j + 1][i - 1] - defy[j][i - 1]
    self.defy = defy
    return

  def lagrange(self, v):
    sum = 0.0
    for i in range(self.n):
        product = 1.0
        for j in range(self.n):
          if j == i:
            continue
```

```
          else:
            product *= (v - self.x[j]) / (self.x[i] - self.x[j])
        sum += self.y[i] * product
    return sum


  def diff(self, k, i):
    n = len(self.x)
    if k == 0:
      return self.y[i]
    elif i + k >= n:
      raise ValueError(”Index out of bounds”)
    else:
      return (self.diff(k - 1, i + 1) - self.diff(k - 1, i)) / (self.x[i + k] - self.x[i])

  def newton(self, v):
    sum = self.y[0]
    for i in range(1, self.n):
        product = 1.0
        for j in range(i):
          product *= v - self.x[j]
        sum += self.diff(i, 0) * product
    return sum

  def gauss(self, v, h):
    a = len(self.y) // 2
    if v>self.x[a]:
      t = (v- self.x[a]) / h
      n = len(self.defy)
      pn = self.defy[a][0] + t * self.defy[a][1] + ((t * (t - 1)) / 2) * self.defy[a - 1][2]
      tn = t * (t - 1)
      for i in range(3, n):
        if i % 2 == 1:
            n = int((i + 1) / 2)
            tn *= (t + n - 1)
            pn += ((tn / factorial(i)) * self.defy[a - n + 1][i])
        else:
            n = int(i / 2)
            tn *= (t - n)
            pn +=((tn / factorial(i)) * self.defy[a - n][i])

    elif v < self.x[a]:
      t = (v- self.x[a]) / h
      n = len(self.defy)

      pn = self.defy[a][0] + t * self.defy[a - 1][1] + ((t * (t + 1)) / 2) * self.defy[a - 1][2]
      tn = t * (t + 1)
      for i in range(3, n):
        if i % 2 == 1:
            n = int((i + 1) / 2)
            tn *= (t + n -1)
        else:
            n = int(i / 2)
```

```
            tn *= (t - n)

        fact = factorial(i)
        pn += (tn / fact) * self.defy[a - n][i]
    else:
      raise ValueError("Error in Gauss")

    return pn

  def bessel(self, v, h):
    n = len(self.x) - 1
    center = n // 2
    a = self.x[center]
    t = (v - a) / h
    result = (self.defy[center][0]+self.defy[center+1][0])/2 + (t- 1/2)*self.defy[center][1] + t*(t-1)/2*(self.defy[center - 1][2] + self.defy[center][2])/2
    term = t*(t-1)/2
    for k in range(3, n + 1):
      if k % 2 == 0:
        term /= (t-1/2)
        term*= (t + (k // 2 - 1)) * (t - (k // 2 ))/k
        result += term * (self.defy[center - 1 - (k // 2 - 1)][k] + self.defy[center- (k // 2 - 1 )][k]) / 2
      else:
        term *= (t - 1/2)/ k
        result += term * self.defy[center - k // 2][k]

    return result

  def sterling(self, v, h):
    n = len(self.x) - 1
    center = n // 2
    a = self.x[center]
    t = (v - a) / h

    result = self.defy[center][0] + t * (self.defy[center - 1][1] + self.defy[center][1]) / 2 + t**2/2 * self.defy[center - 1][2]
    term = t**2/2
    for k in range(3, n):
      if k % 2 == 0:
        term *= t / k
        result += term * self.defy[center - k//2][k]
      else:
        term *= (t**2 - int(k / 2)**2) / (k * t)
        result += term * (self.defy[center - k//2 - 1][k] + self.defy[center - k//2][k]) / 2

    return result
```

4.2 Пример работы программы



5 Github

Ссылка на GitHub

6 Вывод

В этой работе я ознакомился с различными методами квадратичной аппроксимации