Университет ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04

Программная инженерия

Дисциплина «Вычислительная математика»

Отчет

По лабораторной работе №5

Вариант 11

Выполнил: Савельева Диана Александровна

P32082

Преподаватель: Машина Екатерина Алексеевна

Санкт-Петербург, 2023 г.

***1. Цель лабораторной работы:*** решить задачу интерполяции, найти значения функции при заданных значениях аргумента, отличных от узловых точек.

***2. Порядок выполнения работы:***

*Вычислительная реализация задачи:*

1. Выбрать из табл. 1 заданную по варианту таблицу (таблица 1.1 – таблица 1.5);

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

3. Вычислить значения функции для аргумента (см. табл.1), используя первую или вторую интерполяционную формулу Ньютона. Обратить внимание какой конкретно формулой необходимо воспользоваться;

4. Вычислить значения функции для аргумента (см. табл. 1), используя первую или вторую интерполяционную формулу Гаусса. Обратить внимание какой конкретно формулой необходимо воспользоваться;   
 5. Подробные вычисления привести в отчете.

*Программная реализация задачи:*

1. Исходные данные задаются тремя способами:

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. Пользователь выбирает уравнение, исследуемый интервал и количество точек на интервале (не менее двух функций).

2. Сформировать и вывести таблицу конечных разностей;

3. Вычислить приближенное значение функции для заданного значения аргумента, введенного с клавиатуры, указанными методами (см. табл. 5.2). Сравнить полученные значения;

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

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

6. Проанализировать результаты работы программы.

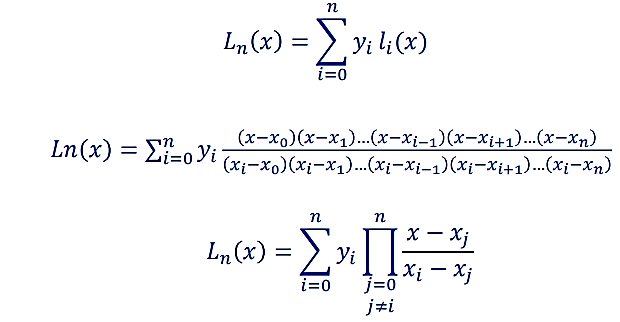
*Необязательное задание (до 20 баллов)*

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

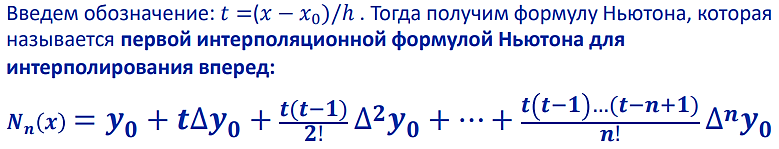
2. Реализовать в программе вычисление значения функции для заданного значения аргумента, введенного с клавиатуры, используя схемы Бесселя.

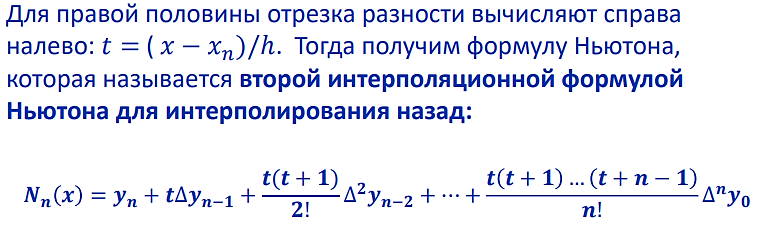
***3. Рабочие формулы:***

* *Многочлен Лагранжа*

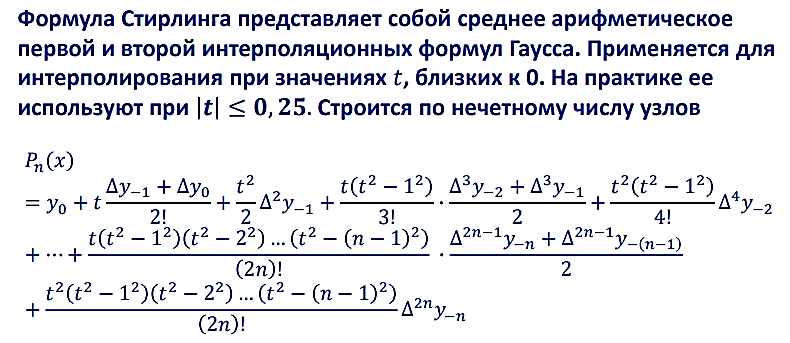
**

* *Многочлен Ньютона с конечными разностями*

**

**

* *Схемы Стирлинга*

******

***4. Вычислительная часть:***

*Вариант№11*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| X | Y | № варианта | X1 | X2 |
| 0,25 | 1,2557 | 11 | 0,255 | 0,405 |
| 0,3 | 2,1764 |
| 0,35 | 3,1218 |
| 0,4 | 4,0482 |
| 0,45 | 5,9875 |
| 0,5 | 6,9195 |
| 0,55 | 7,8359 |

*2. Вычислить значения функции для аргумента X1 (см. табл.1), используя первую или вторую интерполяционную формулу Ньютона. Обратить внимание какой конкретно формулой необходимо воспользоваться.*

Воспользуемся формулой Ньютона для интерполирования вперед, т.к. X1=0,255 лежит в левой половине отрезка и 0,25<0,255<0,3.

Для X1 = 0,255:

t= (x-x0)/h = (0,255-0,25)/0,05 = 0,1.

Тогда *N6(x) = y0 +t∆ y0+ t(t-1)/2! \*∆2 y0 + t(t-1) (t-2)/3! \*∆3 y0 + t(t-1) (t-3)/4! \*∆4 y0+ t(t-1) (t-2) (t-3) (t-4)/5! \*∆5 y0 + t(t-1) (t-2) (t-3) (t-4) (t-5)/6! \*∆6 y0 = 1,1225.*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *№* | *xi* | *yi* | *∆ yi* | *∆2 yi* | *∆3 yi* | *∆4 yi* | *∆5 yi* | *∆6 yi* |
| *0* | *0,25* | *1,2557* | *∆ y0 = 0,921* | *∆2 y0= 0,0247* | *∆3 y0= -0,0437* | *∆4 y0= 1,0756* | *∆5 y0= -4,1277* | *∆6 y0= 10,1917* |
| *1* | *0,3* | *2,1764* | *∆ y1 = 0,945* | *∆2 y1= -0,019* | *∆3 y1= 1,0319* | *∆4 y1= -3,0521* | *∆5 y1= 6,064* |  |
| *2* | *0,35* | *3,1218* | *∆ y2 =*  *0,926* | *∆2 y2= 1,0129* | *∆3 y2= -2,0202* | *∆4 y2= 3,0119* |  |  |
| *3* | *0,4* | *4,0482* | *∆ y3=*  *1,939* | *∆2 y3= -1,0073* | *∆3 y3= 0,9917* |  |  |  |
| *4* | *0,45* | *5,9875* | *∆ y4 =*  *0,932* | *∆2 y4= -0,0156* |  |  |  |  |
| *5* | *0,5* | *6,9195* | *∆ y5 =*  *0,916* |  |  |  |  |  |
| *6* | *0,55* | *7,8359* |  |  |  |  |  |  |

*3. Вычислить значения функции для аргумента X2 (см. табл. 1), используя первую или вторую интерполяционную формулу Гаусса. Обратить внимание какой конкретно формулой необходимо воспользоваться.*

Воспользуемся первой формулой Гаусса, т.к. X2=0,405 лежит в правее a = 0,4.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| *xi* | *yi* | *∆ yi* | *∆2 yi* | *∆3 yi* | *∆4 yi* | *∆5 yi* | *∆6 yi* |
| *x-3=0,25* | *y-3=*  *1,2557* | *∆ y-3 = 0,921* | *∆2 y-3= 0,0247* | *∆3 y-3= -0,0437* | *∆4 y-3= 1,0756* | *∆5 y-3= -4,1277* | *∆6 y-3= 10,1917* |
| *x-2=0,3* | *y-2=*  *2,1764* | *∆ y-2 = 0,945* | *∆2 y-2= -0,019* | *∆3 y-2= 1,0319* | *∆4 y-2= -3,0521* | *∆5 y-2= 6,064* |  |
| *x-1=0,35* | *y-1=*  *3,1218* | *∆ y-1 =*  *0,926* | *∆2 y-1= 1,0129* | *∆3 y-1= -2,0202* | *∆4 y-1= 3,0119* |  |  |
| *x0=0,4* | *y0=*  *4,0482* | *∆ y0=*  *1,939* | *∆2 y0= -1,0073* | *∆3 y0= 0,9917* |  |  |  |
| *x1=0,45* | *Y1=*  *5,9875* | *∆ y1 =*  *0,932* | *∆2 y1= -0,0156* |  |  |  |  |
| *x2=0,5* | *y2=*  *6,9195* | *∆ y2 =*  *0,916* |  |  |  |  |  |
| *x3=0,55* | *y3=*  *7,8359* |  |  |  |  |  |  |

Для X2 = 0,405:

t= (x-x0)/h = (0,405-0,4)/0,05 = 0,005.

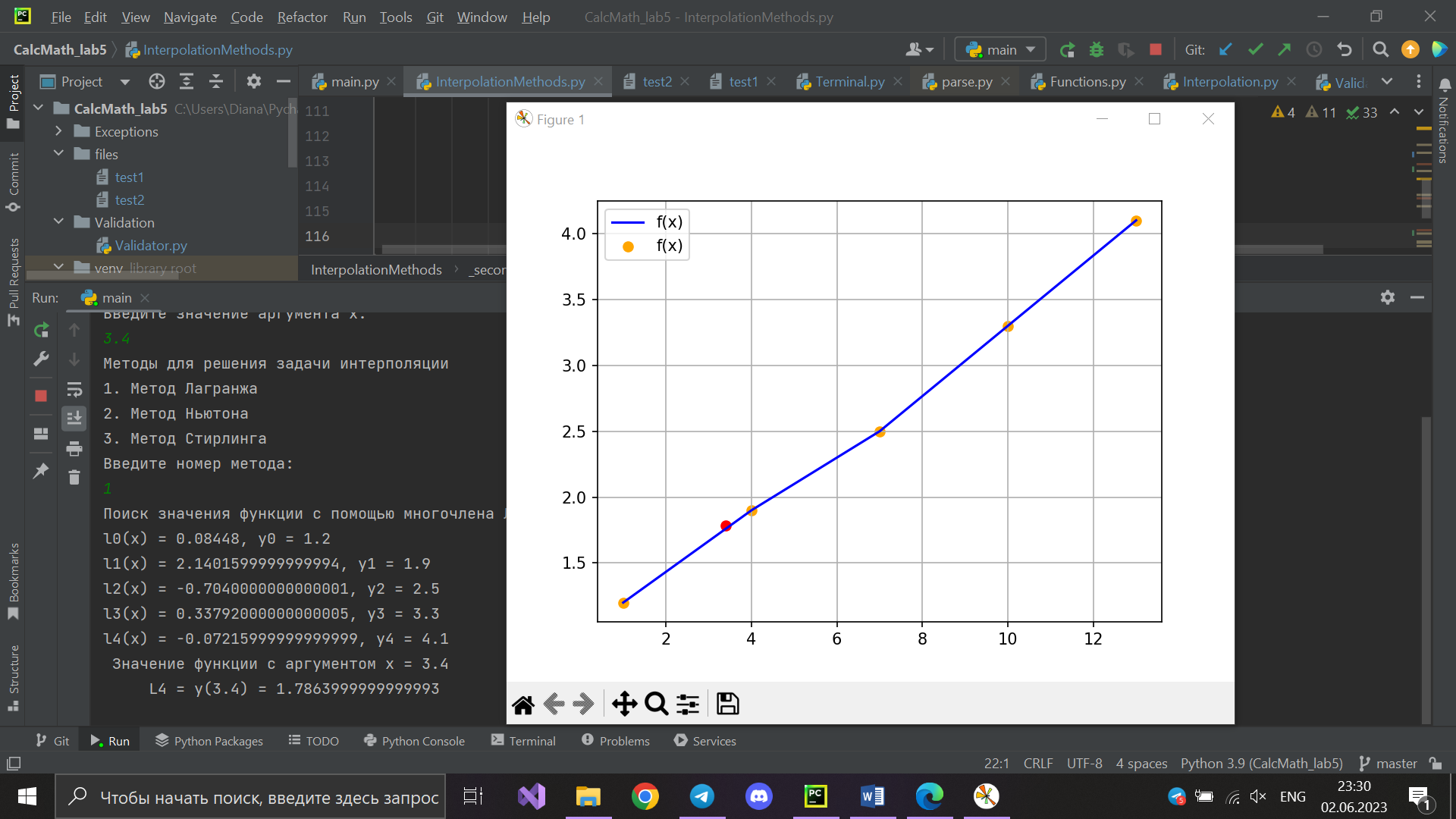
Тогда *P6(x) = y0 +t∆ y0+ t(t-1)/2! \*∆2 y-1 + t(t-1) (t-2)/3! \*∆3 y-1 + t(t-1) (t-3)/4! \*∆4 y-2+ t(t-1) (t-2) (t-3) (t-4)/5! \*∆5 y-2 + t(t-1) (t-2) (t-3) (t-4) (t-5)/6! \*∆6 y-3 = 4,05341.*

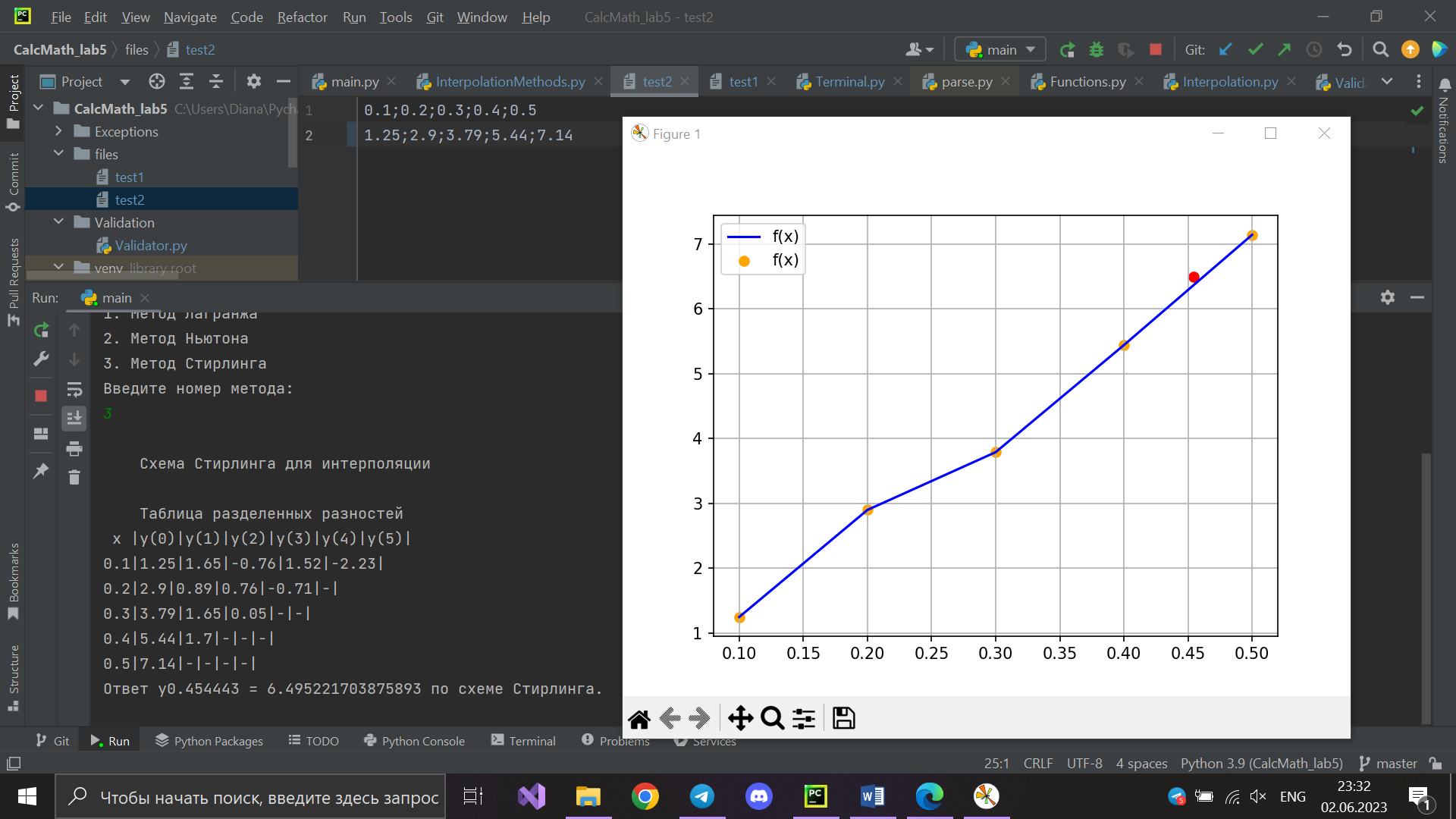
***5. Листинг:***

class Interpolation:  
  
 def \_\_init\_\_(self, x\_array, y\_array, x):  
 self.\_\_x\_array = x\_array  
 self.\_\_y\_array = y\_array  
 self.\_\_x = x  
  
 def getX(self):  
 return self.\_\_x  
  
 def getArrayX(self):  
 return self.\_\_x\_array  
  
 def getArrayY(self):  
 return self.\_\_y\_array  
  
 def setX(self, x):  
 self.\_\_x = x

import statistics  
from math import factorial, ceil, floor  
  
import numpy  
from numpy import delete  
  
from Interpolation import Interpolation  
  
  
class InterpolationMethods(Interpolation):  
  
 def \_calculateLagrangeInterpolation(self):  
 x\_array = self.getArrayX()  
 y\_array = self.getArrayY()  
 length = len(x\_array)  
 x = self.getX()  
 L = []  
 for i in range(length):  
 l\_i = 1  
 for j in range(length):  
 if i != j:  
 l\_i \*= (x - x\_array[j]) / (x\_array[i] - x\_array[j])  
 l\_i \*= y\_array[i]  
 L.append(l\_i)  
 return L  
  
 def \_calculateSumLagrange(self, l):  
 return sum(l)  
  
 def methodLagrange(self):  
 print(f'Поиск значения функции c помощью многочлена Лагранжа с аргументом x = {self.getX()}')  
 L = self.\_calculateLagrangeInterpolation()  
 y\_array = self.getArrayY()  
 x = self.getX()  
 for i in range(len(L)):  
 print(f'l{i}(x) = {L[i]}, y{i} = {y\_array[i]}')  
 print(  
 f' Значение функции с аргументом x = {x}\n\t L{len(y\_array) - 1} = y({x}) = {self.\_calculateSumLagrange(L)}')  
 return self.\_calculateSumLagrange(L)  
  
 def methodNewton(self):  
 print('\n\tМногочлен Ньютона для интерполяции')  
 table = self.formulateTableOfFiniteDifferences()  
 n = len(self.getArrayX())  
 print('\n\tТаблица конечных разностей\n x |', end='')  
 for i in range(n + 1):  
 print(f'y({i})', end='|')  
 print()  
 for row in range(n):  
 for column in range(n + 1):  
 print(table[row][column], end='|')  
 print()  
 median = self.\_findMiddle()  
 index = self.\_findIndexForT()  
 if self.\_defineRule(median):  
 N = self.\_firtsRuleNewton(index, table)  
 print(f'Используем правило №1. Ответ y{self.getX()} = {N}')  
 else:  
 N = self.\_secondRuleNewton(table)  
 print(f'Используем правило №2. Ответ y{self.getX()} = {N}')  
 return N  
  
  
  
 def formulateTableOfFiniteDifferences(self):  
 x\_array = self.getArrayX()  
 y\_array = self.getArrayY()  
 n = len(x\_array)  
 table = [['-' for \_ in range(n + 1)] for \_ in range(n)]  
 for column in range(n + 1):  
 for row in range(n):  
 if column == 1:  
 table[row][column] = y\_array[row]  
 elif column == 0:  
 table[row][column] = x\_array[row]  
 else:  
 if row < n + 1 - column:  
 table[row][column] = round(table[row + 1][column - 1] - table[row][column - 1], 4)  
 return table  
  
 def \_findMiddle(self):  
 return statistics.median(self.getArrayX())  
  
 def \_defineRule(self, median):  
 x = self.getX()  
 return x < median  
  
 def \_firtsRuleNewton(self, index, table):  
 n = len(self.getArrayX())  
 t = self.\_findT(index)  
 N = 0  
 for i in range(2, n + 2):  
 try:  
 y\_i = table[index][i - 1]  
 f = 1  
 for j in range(i - 2):  
 f \*= (t - j)  
 y\_i = y\_i \* f / float(factorial(i - 2))  
 N += y\_i  
 except:  
 continue  
 return N  
  
 def \_secondRuleNewton(self, table):  
 n = len(self.getArrayX())  
 t = self.\_findT(n - 1)  
 N = 0  
 coefficient = 0  
 for i in range(n - 1, -1, -1):  
 try:  
 y\_i = table[i][coefficient + 1]  
 f = 1  
 for j in range(coefficient):  
 f \*= (t + j)  
 y\_i \*= f / factorial(coefficient)  
 N += y\_i  
 coefficient += 1  
 except:  
 continue  
 return N  
  
 def \_findIndexForT(self):  
 x\_array = self.getArrayX()  
 n = len(x\_array)  
 x = self.getX()  
 for i in range(n):  
 if x\_array[i] > x:  
 return i - 1  
  
 def \_findT(self, index):  
 x = self.getX()  
 x\_array = self.getArrayX()  
 return (x - x\_array[index]) / self.\_findH()  
  
 def \_findH(self):  
 x\_array = self.getArrayX()  
 return x\_array[1] - x\_array[0]  
  
 def methodStirling(self):  
 print('\n\tСхема Стирлинга для интерполяции')  
 table = self.formulateTableOfFiniteDifferences()  
 n = len(self.getArrayX())  
 print('\n\tТаблица разделенных разностей\n x |', end='')  
 for i in range(n + 1):  
 print(f'y({i})', end='|')  
 print()  
 for row in range(n):  
 for column in range(n + 1):  
 print(table[row][column], end='|')  
 print()  
 result = self.\_calculateStrirling(table)  
 print(f'Ответ y{self.getX()} = {result} по схеме Стирлинга.')  
 return result  
  
  
 def \_calculateStrirling(self, table):  
 y\_array = self.getArrayY()  
 factors = self.create\_factorial(len(y\_array))  
 table = list(delete(table, [0], 1))  
 middle = len(self.getArrayX()) // 2  
 t = self.\_findT(middle)  
 result = y\_array[middle]  
 for i in range(1, middle + 1):  
 try:  
 mul = 1  
 for j in range(1, i):  
 mul \*= (t \* t - j \* j)  
 result += 1 / factors[2 \* i - 1] \* t \* mul \* \  
 (float(table[-(i - 1) + middle][2 \* i - 1]) + float(table[-i + middle][2 \* i - 1])) / 2  
 result += 1 / factors[2 \* i] \* t \* t \* mul \* (float(table[-i + middle][2 \* i]))  
 except:  
 continue  
 return result  
  
  
 def create\_factorial(self, n):  
 result = []  
 for i in range(n + 1):  
 fact = 1  
 for j in range(1, i + 1):  
 fact \*= j  
 result.append(fact)  
 return result

***6. Результаты выполнения:***





***7. Выводы:*** я научилась решать задачу интерполяции, находить значения функции при заданных значениях аргумента, отличных от узловых точек.