Федеральное государственное автономное

образовательное учреждение высшего образования

Национальный Исследовательский Университет ИТМО

Лабораторная работа 4

«Аппроксимация функции методом наименьших квадратов»

Дисциплина: Вычислительная математика

Вариант 13

Выполнил: Терехин Никита Денисович

Факультет: Программной инженерии и компьютерной техники

Группа: P3208

Преподаватель: Машина Екатерина Алексеевна

г. Санкт-Петербург, 2024 год

Оглавление

Цель работы

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

Текст задания

Для исследования использовать:
- многочлен Лагранжа;
- многочлен Ньютона;
- многочлен Гаусса.

Вычислительная реализация задачи
1. Выбрать из табл. 1 заданную по варианту таблицу $y = f(x)$;
2. Построить таблицу конечных разностей для заданной таблицы. Таблицу отразить в отчете;
3. Вычислить значения функции для аргумента $X1$ (см. табл.1), используя первую или вторую интерполяционную формулу Ньютона. Обратить внимание какой конкретно формулой необходимо воспользоваться;
4. Вычислить значения функции для аргумента $X2$ (см. табл. 1), используя первую или вторую интерполяционную формулу Гаусса. Обратить внимание какой конкретно формулой необходимо воспользоваться;
5. Подробные вычисления привести в отчете.

Рабочие формулы методов

Многочлен Ньютона для интерполирования вперед

$$t = \frac{(x - x_i)}{h}$$

$$N(t) = y_i + t\Delta y_i + \frac{t(t-1)}{2!}\Delta^2 y_i + \frac{t(t-1)(t-2)}{3!}\Delta^3 y_i + \frac{t(t-1)(t-2)(t-3)}{4!}\Delta^4 y_i + $$
$$+ \frac{t(t-1)(t-2)(t-3)(t-4)}{5!}\Delta^5 y_i + \dots + \frac{t(t-1)\dots(t-n+1)}{n!}\Delta^n y_i$$

Вторая интерполяционная формула Гаусса:

$$t = \frac{(x - x_0)}{h}$$

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

Вычислительная реализация

Исходные данные

	x	y	X_1	X_2
Таблица 1.3	1,10	0,2234	1,168	1,463
	1,25	1,2438		
	1,40	2,2644		
	1,55	3,2984		
	1,70	4,3222		
	1,85	5,3516		
	2,00	6,3867		

Таблица конечных разностей

i	x_i	y_i	Δy_i	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^4 y_i$	$\Delta^5 y_i$	$\Delta^6 y_i$
0	1,10	0,2234	1,0204	0,0002	0,0132	-0,0368	0,0762	-0,1313
1	1,25	1,2438	1,0206	0,0134	-0,0236	0,0394	-0,0551	
2	1,40	2,2644	1,034	-0,0102	0,0158	-0,0157		
3	1,55	3,2984	1,0238	0,0056	0,0001			
4	1,70	4,3222	1,0294	0,0057				
5	1,85	5,3516	1,0351					
6	2,00	6,3867						

Для вычисления $X_1 = 1,168 \in [x_0;\ x_1]$ воспользуемся первой интерполяционной формулой Ньютона для интерполирования вперед:

$$t = \frac{(x - x_1)}{h} = \frac{(X_1 - x_1)}{h} = \frac{(1{,}168 - 1{,}25)}{0{,}15} \approx -\ 0,547$$

$N(1,168) = y_1 + t\Delta y_1 + \frac{t(t-1)}{2!}\Delta^2 y_1 + \frac{t(t-1)(t-2)}{3!}\Delta^3 y_1 + \frac{t(t-1)(t-2)(t-3)}{4!}\Delta^4 y_1 +$
$+ \frac{t(t-1)(t-2)(t-3)(t-4)}{5!}\Delta^5 y_1 = 1,2438 + (-0,547) \cdot$
$\cdot 1,0206 + \frac{(-0,547)\cdot(-1,547)}{2} \cdot 0,0134 + \frac{(-0,547)\cdot(-1,547)\cdot(-2,547)}{6} \cdot (-0,0236) +$
$+ \frac{(-0,547)\cdot(-1,547)\cdot(-2,547)\cdot(-3,547)}{24} \cdot 0,0394 + \frac{(-0,547)\cdot(-1,547)\cdot(-2,547)\cdot(-3,547)\cdot(-4,547)}{120} \cdot$
$\cdot (-0,0551) \approx 0,3821$

i	x_i	y_i	Δy_i	$\Delta^2 y_i$	$\Delta^3 y_i$	$\Delta^4 y_i$	$\Delta^5 y_i$	$\Delta^6 y_i$
-3	1,10	0,2234	1,0204	0,0002	0,0132	-0,0368	0,0762	-0,1313
-2	1,25	1,2438	1,0206	0,0134	-0,0236	0,0394	-0,0551	
-1	1,40	2,2644	1,034	-0,0102	0,0158	-0,0157		
0	1,55	3,2984	1,0238	0,0056	0,0001			
1	1,70	4,3222	1,0294	0,0057				
2	1,85	5,3516	1,0351					
3	2,00	6,3867						

$a = x_0 = 1,55$

Для вычисления $X_2 = 1,463 < a$ воспользуемся второй интерполяционной формулой Гаусса:

$t = \frac{(x - x_0)}{h} = \frac{(X_2 - a)}{h} = \frac{(1,463 - 1,55)}{0,15} \approx -0,58$

$P(1,463) = y_0 + t\Delta y_{-1} + \frac{t(t+1)}{2!}\Delta^2 y_{-1} + \frac{(t-1)t(t-2)}{3!}\Delta^3 y_{-2} + \frac{(t+2)(t+1)t(t-1)}{4!}\Delta^4 y_{-2} +$
$+ \frac{(t+2)(t+1)t(t-1)(t-2)}{5!}\Delta^5 y_{-3} + \frac{(t+3)(t+2)(t+1)t(t-1)(t-2)}{6!}\Delta^6 y_{-3} = 3,2984 + (-0,58) \cdot$
$\cdot 1,034 + \frac{(-0,58)\cdot(0,42)}{2} \cdot (-0,0102) + \frac{(0,42)\cdot(-1,58)\cdot(-1,58)}{6} \cdot (-0,0236) +$

$$+ \frac{(1{,}42)\cdot(0{,}42)\cdot(-0{,}58)\cdot(-1{,}58)}{24} \cdot 0,0394 \ + \frac{(1{,}42)\cdot(0{,}42)\cdot(-0{,}58)\cdot(-1{,}58)}{120} \cdot$$

$$\cdot \ (0,0762) \ + \frac{(1{,}42)\cdot(0{,}42)\cdot(-0{,}58)\cdot(-1{,}58)\cdot(-2{,}58)\cdot(-3{,}58)}{720} \cdot (-\ 0,1313) \approx 2,9172$$

Программная реализация

Листинг программы

```
# main.py

import math

import matplotlib.pyplot as plt
from tabulate import tabulate

from P3208.Terekhin_367558.lab1.exceptions import
InterpolationError
from P3208.Terekhin_367558.lab2.main import request_from_list
from P3208.Terekhin_367558.lab2.readers import AbstractReader,
READERS
from P3208.Terekhin_367558.lab5.interpolation import
INTERPOLATIONS

if __name__ == '__main__':
    reader: AbstractReader = request_from_list(READERS)
    points: list[tuple[float, float]] =
sorted(reader.read_interpolation_data())
    argument: float = reader.read_interpolation_argument()

    n: int = len(points)
    x: list[float] = [points[i][0] for i in range(n)]
    y: list[float] = [p[1] for p in points]
    x_range: list[float] = [i / 100 for i in range(math.floor(x[0]
* 100), math.ceil(x[-1] * 100))]

    colors = ['red', 'green', 'blue', 'orange', 'purple', 'cyan',
'magenta', 'pink', 'yellow', 'brown']
    color_index: int = 0

    for interpolation in INTERPOLATIONS:
        try:
            interpolation.set_points(points)
            y_range: list[float] = [interpolation.interpolate(i)
for i in x_range]
            plt.plot(x_range, y_range, color=colors[color_index %
len(colors)])
```

```
            color_index += 1
            print(tabulate([interpolation.description]))
            print(f'For x = {argument} y =
{round(interpolation.interpolate(argument), 3)}')
        except InterpolationError as e:
            print(e)

    plt.scatter(x, y)
    plt.show()
```

```
# interpolation.py

from abc import abstractmethod
from typing import Final

from tabulate import tabulate

from P3208.Terekhin_367558.lab1.exceptions import
InterpolationError
from P3208.Terekhin_367558.lab2.functions import Describable


class Interpolation(Describable):
    option_name = 'interpolation'

    def __init__(self, description: str):
        super().__init__(description)
        self.points: list[tuple[float, float]] = []

    def set_points(self, points: list[tuple[float, float]]) ->
None:
        self.points = points

    @abstractmethod
    def interpolate(self, x: float) -> float:
        pass


class LagrangeInterpolation(Interpolation):
    def __init__(self):
        super().__init__('Lagrange interpolation')

    def interpolate(self, x: float) -> float:
        res: float = 0
        for i in range(len(self.points)):
            subres: float = 1
            for j in range(len(self.points)):
```

```
                if i != j:
                    subres *= x - self.points[j][0]
                    subres /= self.points[i][0] - self.points[j][0]
            res += self.points[i][1] * subres
        return res


class NewtonSeparateInterpolation(Interpolation):
    def __init__(self):
        super().__init__('Newton\'s separate interpolation')
        self.diverse_table = []

    def interpolate(self, x: float) -> float:
        res: float = self.points[0][1]
        subres: float = 1
        for i in range(len(self.diverse_table) - 1):
            subres *= x - self.points[i][0]
            res += self.diverse_table[i + 1][0] * subres
        return res

    def set_points(self, points: list[tuple[float, float]]) ->
None:
        self.points = points
        n = len(self.points)
        self.diverse_table = [[p[1] for p in points]]
        for i in range(1, n):
            self.diverse_table.append([0] * n)
        headers = ['y']
        for i in range(1, n):
            headers.append(f'f{i}')
            for j in range(n - i):
                self.diverse_table[i][j] = (self.diverse_table[i -
1][j + 1]
                                            - self.diverse_table[i
- 1][j]) / (self.points[j + i][0] - self.points[j][0])
        print(tabulate([[headers[k]] + list(map(lambda x: round(x,
4), self.diverse_table[k])) for k in range(n)],
tablefmt='pretty'))


class NewtonFiniteInterpolation(Interpolation):
    def __init__(self):
        super().__init__('Newton\'s finite interpolation')
        self.diverse_table = []
        self.h = 0

    def set_points(self, points: list[tuple[float, float]]):
        h = points[1][0] - points[0][0]
```

```
        for i in range(len(points) - 1):
            if (points[i + 1][0] - points[i][0]) - h >= 0.001:
                raise InterpolationError('Can\'t solve using finite
sums. Step isn\'t constant')
        self.points = points
        self.diverse_table = [[p[1] for p in points]]
        self.h = h
        n = len(self.points)
        for i in range(1, n):
            self.diverse_table.append([0] * n)
        headers = ['y']
        for i in range(1, n):
            headers.append(f'Δ^{i}y' if i != 1 else 'Δy')
            for j in range(n - i):
                self.diverse_table[i][j] = (self.diverse_table[i -
1][j + 1]
                                            - self.diverse_table[i
- 1][j])
        print(tabulate([[headers[k]] + list(map(lambda x: round(x,
4), self.diverse_table[k])) for k in range(n)], tablefmt='pretty',
numalign='decimal'))

    def interpolate(self, x: float) -> float:
        ind: int = 0
        res: float = 0
        subres = 1
        mid = (self.points[-1][0] + self.points[0][0]) / 2
        if x <= mid:
            while x >= self.points[ind + 1][0]:
                ind += 1
            res += self.points[ind][1]
            for i in range(len(self.diverse_table) - 1 - ind):
                subres *= (x - self.points[ind + i][0]) / self.h
                res += self.diverse_table[i + 1][ind] * subres
        else:
            while x <= self.points[-(ind + 2)][0]:
                ind += 1
            res += self.points[-(ind + 1)][1]
            for i in range(len(self.diverse_table) - 1 - ind):
                subres *= (x - self.points[-(i + ind + 1)][0]) /
self.h
                res += self.diverse_table[i + 1][-(ind + i + 2)] *
subres
        return res


INTERPOLATIONS: Final[list[Interpolation]] = [
    LagrangeInterpolation(),
```

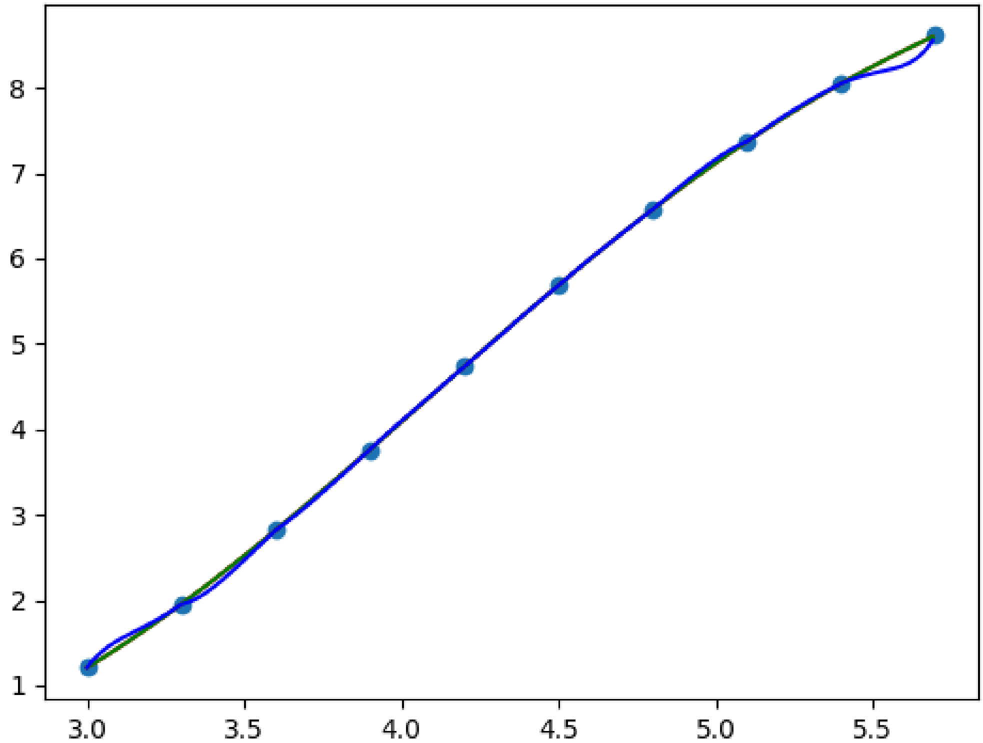
```
    NewtonSeparateInterpolation(),
    NewtonFiniteInterpolation()
]
```

Результаты работы программы

```
1. From console
2. From file
3. Using function
Choose option:
3
1. x^3 + 4,81x^2 - 17,37x + 5,38
2. 2x^3 - 1,89x^2 - 5x + 2,34
3. e^(x / 3) - 2cos(x + 4)
4. x^2 - e^x
Choose function:
3
Input interval using two numbers:
3 6
Input points amount:
point
Should be integer number. Try again:
2
From 4 to 20 points needed for approximation. Try again:
10
Enter argument to interpolate:
argument
Should be float number. Try again:
-3
-  -  -  -  -  -  -  -    -  -  -  -  -  -  -  -  -  -  -  -  -
L  a  g  r  a  n  g  e    i  n  t  e  r  p  o  l  a  t  i  o  n
-  -  -  -  -  -  -  -    -  -  -  -  -  -  -  -  -  -  -  -  -
For x = -3.0 y = -24.081
+----+---------+---------+---------+---------+---------+---------+
---------+---------+-------+--------+
| y  | 1.2105  |  1.952  | 2.8176  | 3.7613  | 4.7335  | 5.6857  |
6.5752  | 7.3694  | 8.049 | 8.6106 |
| f1 | 2.4718  | 2.8853  | 3.1457  | 3.2407  |  3.174  |  2.965  |
2.6472  | 2.2655  | 1.872 |   0    |
| f2 | 0.6892  |  0.434  | 0.1584  | -0.1111 | -0.3483 | -0.5296 |
-0.6363 | -0.6558 |   0   |   0    |
| f3 | -0.2836 | -0.3062 | -0.2994 | -0.2635 | -0.2015 | -0.1185 |
-0.0217 |    0    |   0   |   0    |
| f4 | -0.0189 | 0.0057  | 0.0299  | 0.0517  | 0.0691  | 0.0806  |
0    |    0    |   0   |   0    |
| f5 | 0.0164  | 0.0162  | 0.0145  | 0.0116  | 0.0077  |    0    |
0    |    0    |   0   |   0    |
```

```
| f6 | -0.0001 | -0.0009 | -0.0016 | -0.0022 |    0    |    0    |
0    |    0    |   0   |   0    |
| f7 | -0.0004 | -0.0003 | -0.0003 |    0    |    0    |    0    |
0    |    0    |   0   |   0    |
| f8 |   0.0   |   0.0   |    0    |    0    |    0    |    0    |
0    |    0    |   0   |   0    |
| f9 |   0.0   |    0    |    0    |    0    |    0    |    0    |
0    |    0    |   0   |   0    |
+----+---------+---------+---------+---------+---------+---------+
---------+---------+-------+--------+
- - - - - - - -   - - - - - - - -   - - - - -
- - - - - - - -
N e w t o n ' s   s e p a r a t e   i n t e r
p o l a t i o n
- - - - - - - -   - - - - - - - -   - - - - -
- - - - - - - -
For x = -3.0 y = -24.081
+------+---------+---------+---------+---------+---------+--------
-+---------+---------+-------+--------+
|  y   | 1.2105  |  1.952  | 2.8176  | 3.7613  | 4.7335  | 5.6857
| 6.5752  | 7.3694  | 8.049  | 8.6106 |
|  Δy  | 0.7415  | 0.8656  | 0.9437  | 0.9722  | 0.9522  | 0.8895
| 0.7942  | 0.6796  | 0.5616 |   0    |
| Δ^2y | 0.1241  | 0.0781  | 0.0285  |  -0.02  | -0.0627 | -0.0953
| -0.1145 | -0.1181 |   0    |   0    |
| Δ^3y | -0.0459 | -0.0496 | -0.0485 | -0.0427 | -0.0326 | -0.0192
| -0.0035 |    0    |   0    |   0    |
| Δ^4y | -0.0037 | 0.0011  | 0.0058  | 0.0101  | 0.0134  | 0.0157
|    0    |    0    |   0    |   0    |
| Δ^5y | 0.0048  | 0.0047  | 0.0042  | 0.0034  | 0.0022  |    0
|    0    |    0    |   0    |   0    |
| Δ^6y | -0.0001 | -0.0005 | -0.0009 | -0.0012 |    0    |    0
|    0    |    0    |   0    |   0    |
| Δ^7y | -0.0004 | -0.0004 | -0.0003 |    0    |    0    |    0
|    0    |    0    |   0    |   0    |
| Δ^8y |   0.0   | 0.0001  |    0    |    0    |    0    |    0
|    0    |    0    |   0    |   0    |
| Δ^9y |   0.0   |    0    |    0    |    0    |    0    |    0
|    0    |    0    |   0    |   0    |
+------+---------+---------+---------+---------+---------+--------
-+---------+---------+-------+--------+
- - - - - - - -   - - - - - -   - - - - - - -
- - - - - -
N e w t o n ' s   f i n i t e   i n t e r p o
l a t i o n
- - - - - - - -   - - - - - -   - - - - - - -
- - - - - -
For x = -3.0 y = -78682899.792
```

```
Process finished with exit code 0
```



Выводы

В ходе выполнения работы я научился решать задачу интерполяции, найти значения функции при заданных значениях аргумента, отличных от узловых точек.

Наиболее точным на глобальной области показал себя метод Лагранжа. В то же время на малых участках метод Ньютона обеспечивает лучшую точность.

Использование конкретного метода зависит от целей интерполирования