Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №5

По вычислительной математике

Вариант 3

Выполнил:

Студент группы P3216

Векшин Арсений Иванович

Преподаватель:

Малышева Татьяна Алексеевна

г. Санкт-Петербург

2024 г.

Оглавление

Цель работы

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

Исходные данные:

Пользователь вводит таблично заданную функцию.

Пользователь вводит аргумент функции, которую требуется найти

Программная реализация задачи:

1. Исходные данные задаются тремя способами:
    1. В виде набора данных (таблицы x, y); пользователь вводит значения с клавиатуры;
    2. В виде сформированных в файле данных (подготовить не менее трех тестовых вариантов);
    3. 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).
2. Сформировать и вывести таблицу конечных разностей;
3. Вычислить приближенное значение функции для заданного значения аргумента, введенного с клавиатуры, указанными методами (см. табл. 2). Сравнить полученные значения.
4. Построить графики заданной функции с отмеченными узлами интерполяции и интерполяционного многочлена Ньютона/Гаусса (разными цветами);

Вычислительная реализация задачи

1. Выбрать из табл. 1 заданную по варианту таблицу $y = f(x)$ (таблица 1.1 – таблица 1.5);
2. Построить таблицу конечных разностей для заданной таблицы. Таблицу отразить в отчете;
3. Вычислить значения функции для аргумента X (см. табл.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), используя первую или вторую интерполяционную формулу Гаусса. Обратить внимание, какой конкретно формулой необходимо воспользоваться;
5. ***Подробные вычисления привести в отчете.***

Вычислительная реализация

0,2234	1,0204	0,0002	0,0132	-0,0368	0,0762	-0,1313
1,2438	1,0206	0,0134	-0,0236	0,0394	-0,0551	
2,2644	1,034	-0,0102	0,0158	-0,0157		
3,2984	1,0238	0,0056	-0.0001			
4,3222	1,0294	0,0057				
5,3516	1,0351					
6,3867						

Для X1 = 1,121 считаем первой формулой (интерполирование вперед), так как это левая половина:

$$N_n(x) = y_0 + t\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + \cdots + \frac{t(t-1)\ldots(t-n+1)}{n!}\Delta^n y_0$$

$$t = \frac{x - x_0}{h} = 0.14$$

$$N_n(x) = 0{,}2234 + 0{,}14 * 1{,}0204 + \cdots = 0{,}371$$

Для X2 = 1.482 считаем первой интерполяционной формулой, так как x > a:

$$P_n(x) = y_0 + t\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_{-1} + \frac{(t+1)t(t-1)}{3!}\Delta^3 y_{-1} + \frac{(t+1)t(t-1)(t-2)}{4!}\Delta^4 y_{-2} + \frac{(t+2)(t+1)t(t-1)(t-2)}{5!}\Delta^5 y_{-2} \ldots + \frac{(t+n-1)\ldots(t-n+1)}{(2n-1)!}\Delta^{2n-1} y_{-(n-1)} + \frac{(t+n-1)\ldots(t-n)}{(2n)!}\Delta^{2n} y_{-n}$$

$$t = \frac{x - x_0}{h} = 0.14$$

$$P_n(x) = 0{,}2234 + 0{,}14 * 1{,}0204 + \ldots = 0.366$$

Исходный код программы

Лагранж

```
class Lagrange_Polynomial(Polynomial):
    def __init__(self, points):
        super().__init__(points, list(np.zeros(len(points))), "lagrange")

        for i in range(len(points)):
            numerator = []
            buff = points[i][1]

            for j in range(len(points)):
                if i==j: continue
                buff /= points[i][0] - points[j][0] # считаем знаменатель
                numerator.append(-1 * points[j][0]) # собираем числитель

            _polynom = expand_brackets(numerator)
            _polynom = [elem * buff for elem in _polynom]
            self.koofs = [self.koofs[i] + _polynom[i] for i in range(len(self.koofs)) ]
```

Ньютон

```
class Newton_Polynomial(Polynomial):

    tree = []
    def __init__(self, points):

        super().__init__(points, list(np.zeros(len(points))), "newton")

        # Соберем таблицу по слоям
        self.tree.append([])

        for i in range(1, len(points)): # собираем первый слой
            self.tree[-1].append((points[i][1]-points[i-1][1]) / (points[i][0]-points[i-1][0]))

        for depth in range(1, len(points)-1):
            self.tree.append([])
            left, right = 0, depth+1
            for i in range(1, len(self.tree[-2])):
                # print(self.tree[-2][i], self.tree[-2][i-1],"/", points[right][0], points[left][0] )
                self.tree[-1].append((self.tree[-2][i] - self.tree[-2][i-1]) /
                                     (points[right][0] - points[left][0]))
                right += 1
                left += 1


        self.koofs[0] = points[0][1]
        for i in range(len(points)-1):
            _polynom = [-1 * points[j][0] for j in range(i+1)]

            _polynom = expand_brackets(_polynom)
            _polynom = [elem * self.tree[i][0] for elem in _polynom]

            while len(_polynom)<len(self.koofs): _polynom = _polynom + [0]
            self.koofs = [self.koofs[i] + _polynom[i] for i in range(len(self.koofs))]
```

Ньютон для равностоящих

```python
class Newton_Stable_Polynomial(Polynomial):

    h = 0
    tree = []


    def __init__(self, points):

        super().__init__(points, list(np.zeros(len(points))), "newton_stable")
        self.h = points[1][0] - points[0][0]


        # Соберем таблицу по слоям
        self.tree.append([y for x,y in points])
        self.tree.append([])

        for i in range(1, len(points)): # собираем первый слой
            self.tree[-1].append(points[i][1]-points[i-1][1])

        for depth in range(1, len(points)-1):
            self.tree.append([])
            for i in range(1, len(self.tree[-2])):
                # print(self.tree[-2][i], self.tree[-2][i-1],"/", points[right][0], points[left][0]
                self.tree[-1].append(self.tree[-2][i] - self.tree[-2][i-1])
```

Полный код программы:
https://github.com/ArsenyVekshin/ITMO/tree/master/CompMath/lab5

Вывод

В результате работы программы были написаны и наглядно визуализированы различные виды интерполяции.

```
   .∧＿∧
(  ・ω・ )つ━☆・*。
⊂  ノ               ・゜+.
 しーJ           °。 + *´¨)
                        .· ´¸.·*´¨)
                              (¸.·´ (¸.·'* ☆ Нужно читать методичку
```