Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

Высшего образования

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа 5 по вычислительной математике

Интерполяция функции

Вариант №13

Группа: P3116

Выполнил:

Сиразетдинов А.Н.

Проверил:

Малышева Т.А.

Г. Санкт-Петербург

2024

Оглавление

Цель работы

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

Порядок выполнения работы

1) Вычислительная часть работы
2) Программная часть работы

Рабочие формулы

интерполяционные формулы Ньютона
для равноотстоящих узлов

Введем обозначение: $t = (x - x_0)/h$. Тогда получим формулу Ньютона, которая называется **первой интерполяционной формулой Ньютона для интерполирования вперед:**

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

Полученное выражение может аппроксимировать функцию на всем отрезке изменения аргумента $[x_0, x_n]$. Однако более целесообразно (с точки зрения повышения точности расчетов) использовать эту формулу для $x_0 \le x \le x_1$. При этом за x_0 может приниматься любой узел интерполяции x_k. Например, для $x_1 \le x \le x_2$, вместо x_0 надо взять значение x_1. Тогда интерполяционный многочлен Ньютона:

$$\boldsymbol{N_n(x) = y_i + t\Delta y_i + \frac{t(t-1)}{2!}\Delta^2 y_i + \cdots + \frac{t(t-1)\dots(t-n+1)}{n!}\Delta^n y_i} \quad (*)$$

Интерполяционную формулу (*) обычно используют для вычислений значений функции в точках левой половины отрезка.

Интерполяционные многочлены Гаусса

Вторая интерполяционная формула Гаусса $(x < a)$

$$\begin{aligned}P_n(x) = y_0 + t\Delta y_{-1} + \frac{t(t+1)}{2!}\Delta^2 y_{-1} + \frac{(t+1)t(t-1)}{3!}\Delta^3 y_{-2} \\ + \frac{(t+2)(t+1)t(t-1)}{4!}\Delta^4 y_{-2} + \cdots \\ + \frac{(t+n-1)\dots(t-n+1)}{(2n-1)!}\Delta^{2n-1} y_{-n} \\ + \frac{(\boldsymbol{t+n})(t+n-1)\dots(t-n+\boldsymbol{1})}{(2n)!}\Delta^{2n} y_{-n}\end{aligned}$$

Оценка погрешности:

$$R_n(x) \approx \frac{\Delta^{2n+1} y_0}{(n+1)!} t(t-1)\dots(t-n)$$

Вычислительная часть

Таблица

	x	y	№ варианта	X_1	X_2
Таблица 1.3	1,10	0,2234	3	1,121	1,482
	1,25	1,2438	8	1,852	1,652
	1,40	2,2644	13	1,168	1,463
	1,55	3,2984	18	1,875	1,575
	1,70	4,3222	23	1,189	1,491
	1,85	5,3516	28	1,891	1,671
	2,00	6,3867	33	1,217	1,473

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

X_1 = 1,168

X_2 = 1,463

Таблица конечных разностей

N	x	y	dy	d2y	d3y	d4y	d5y	d6y
0	1,1	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	1,4	2,2644	1,034	-0,0102	0,0158	-0,0157		
3	1,55	3,2984	1,0238	0,0056	1E-04			
4	1,7	4,3222	1,0294	0,0057				
5	1,85	5,3516	1,0351					
6	2	6,3867						

Интерполяция Ньютона

Вычислить значения функции для аргумента X1 используя первую или вторую интерполяционную формулу Ньютона

Воспользуемся формулой интерполирования **вперед** потому что X1=1,168 лежит в первой половине отрезка

Для X=1,168 $t = \frac{(x-x_0)}{h} = \frac{(1,168-1,1)}{0,15} = 0,453$

$$N_6(x) = y_0 + t\Delta y_0 + \frac{t(t-1)}{2!}\Delta^2 y_0 + \frac{t(t-1)(t-2)}{3!}\Delta^3 y_0 + \frac{t(t-1)(t-2)(t-3)}{4!}\Delta^4 y_0 + \frac{t(t-1)(t-2)(t-3)(t-4)}{5!}\Delta^5 y_0 + \frac{t(t-1)(t-2)(t-3)(t-4)(t-5)}{6!}\Delta^6 y_0$$

$$N_6(x) = 0,2234 + 0,453 * 1,0204 + \frac{0,453(0,453-1)}{2!}0,0002 + \frac{0,453(0,453-1)(0,453-2)}{3!}0,0132 + \frac{0,453(0,453-1)(0,453-2)(0,453-3)}{4!}(-0,0368) + \frac{0,453(0,453-1)(0,453-2)(0,453-3)(0,453-4)}{5!}0,0762 + \frac{0,453(0,453-1)(0,453-2)(0,453-3)(0,453-4)(0,453-5)}{6!}(-0,1313) =$$

$y(1,168) = 0.6934$

Интерполяция Гаусса

Вычислить значение функции для аргумента X2, используя интерполяционную формулу Гаусса

Центральная точка a=1,55. X2 = 1,463<a, следовательно, используем **вторую** интерполяционную формулу

$$t = \frac{(x - x0)}{h} = -0,58$$

$$P_6(x) = y_0 + t\Delta y_{-1} + \frac{t(t+1)}{2!}\Delta^2 y_{-1} + \frac{(t+1)t(t-1)}{3!}\Delta^3 y_{-2} + \frac{(t+2)(t+1)t(t-1)}{4!}\Delta^4 y_{-2} + \frac{(t+2)(t+1)t(t-1)(t-2)}{5!}\Delta^5 y_{-3} + \frac{(t+3)(t+2)(t+1)t(t-1)(t-2)}{6!}\Delta^6 y_{-3}$$

$$P_6(x) = 3,2984 - 0,58*1,034 + \frac{-0,58(-0,58+1)}{2!}(-0,0102) + \frac{(-0,58+1)(-0,58)(-0,58-1)}{3!}(-0,0236) + \frac{(-0,58+2)(-0,58+1)(-0,58)(-0,58-1)}{4!}(0,0394) + \frac{(-0,58+2)(-0,58+1)(-0,58)(-0,58-1)(-0,58-2)}{5!}0,0762 + \frac{(-0,58+3)(-0,58+2)(-0,58+1)(-0,58)(-0,58-1)(-0,58-2)}{6!}(-0,1313)$$

$y(1,463) = 2,699$

Листинг программы

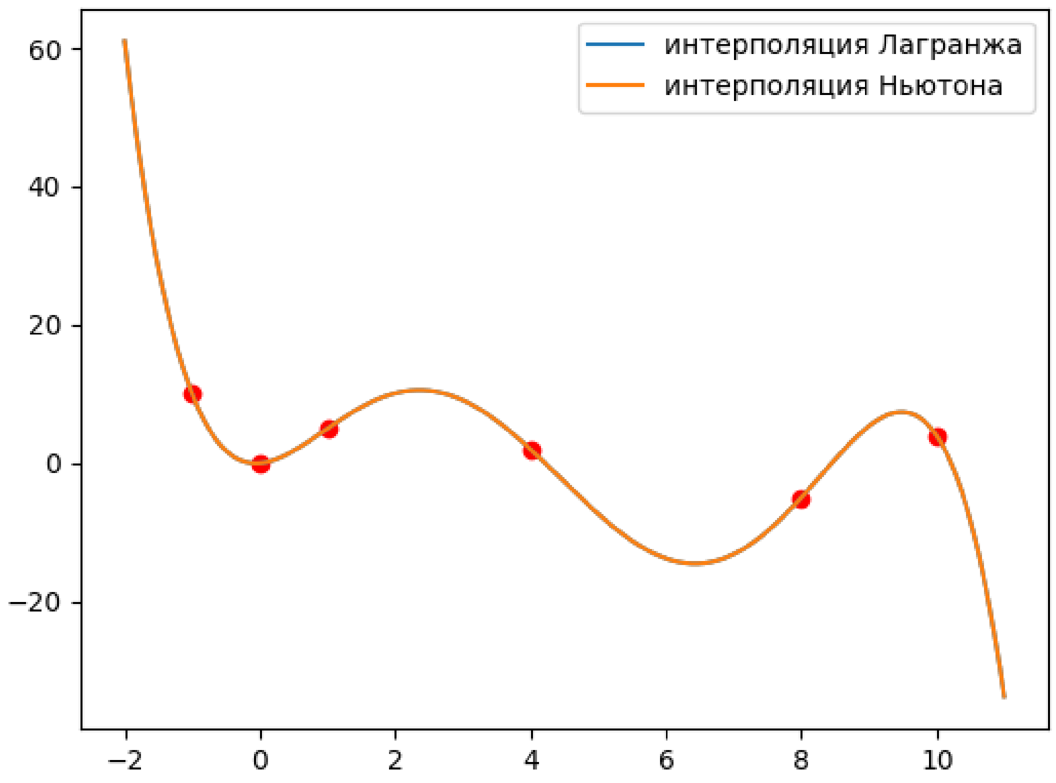
```python
def create_function(self) -> Callable[[float], float]:
    return lambda x: sum(
        yi * math.prod(
            (x - xj) / (xi - xj)
            for xj, yj in self.table
            if xj != xi                 # assuming x are unique this is eqv of i != j
        )
        for xi, yi in self.table
    )
```

```python
class Newton(AbstractInterpolation):
    name = "интерполяция Ньютона"

    @staticmethod
    def check(table: List[Point]) -> bool:
        n = len(table)
        h = table[1].x - table[0].x
        for i in range(2, n):
            if table[i].x - table[i - 1].x != h:
                return True
        return False

    def divided_diffs(self) -> list[list[float]]:
        diff = [[0 for _ in range(self.n)] for _ in range(self.n)]
        for i in range(self.n):
            diff[i][0] = self.table[i].y
        for j in range(1, self.n):
            for i in range(self.n - j):
                diff[i][j] = (diff[i + 1][j - 1] - diff[i][j - 1]) / (self.table[i + j].x - self.table[i].x)
        return diff
...

    def create_function(self) -> Callable[[float], float]:
        diff = self.divided_diffs()
        self.print_diffs(diff)

        return lambda x: diff[0][0] + sum([
            diff[0][k] * math.prod([
                x - self.table[j].x
                for j in range(k)
            ])
            for k in range(1, self.n)
        ])
```

```python
def create_function(self) -> Callable[[float], float]:
    diff = self.divided_diffs()
    self.print_divided_diffs(diff)
    h = self.table[1].x - self.table[0].x
    t_forward = lambda x: (x - self.table[0].x) / h

    forward = lambda x: self.table[0].y + sum([
        diff[0][i] *
        math.prod([
            t_forward(x) - j
            for j in range(i)
        ]) / math.factorial(i)
        for i in range(1, self.n)
    ])

    t_backward = lambda x: (x - self.table[-1].x) / h
    backward = lambda x: self.table[-1].y + sum([
        diff[-i - 1][i] *
        math.prod([
            t_backward(x) + j
            for j in range(i)
        ]) / math.factorial(i)
        for i in range(1, self.n)
    ])

    return lambda x: forward(x) if (self.table[-1].x - self.table[0].x) / 2 < x else backward(x)
```

Результат выполнения программы

```
Выберите вид ввода:
0: сформированные в таблице данные
1: выбранная функция
2: набор данных
0
Введите название файла input.txt
Введите значение аргумента для интерполяции 0.5
+----------------------------------------------------------------+
|                      Разделенные разности                      |
+--------+--------+---------+--------+--------+--------+--------+
|   x    |   y    |   d1y   |  d2y   |  d3y   |  d4y   |  d5y   |
+--------+--------+---------+--------+--------+--------+--------+
| -1.000 | 10.000 | -10.000 | 7.500  | -1.800 | 0.219  | -0.020 |
| 0.000  | 0.000  |  5.000  | -1.500 | 0.174  | -0.005 |   0    |
| 1.000  | 5.000  |  -1.000 | -0.107 | 0.128  |   0    |   0    |
| 4.000  | 2.000  |  -1.750 | 1.042  |   0    |   0    |   0    |
| 8.000  | -5.000 |  4.500  |   0    |   0    |   0    |   0    |
| 10.000 | 4.000  |    0    |   0    |   0    |   0    |   0    |
+--------+--------+---------+--------+--------+--------+--------+
Приближенное значение функции в точке 0.5 методом интерполяция Лагранжа = 1.7883374763257578
Приближенное значение функции в точке 0.5 методом интерполяция Ньютона = 1.7883374763257578
```



Вывод

В ходе выполнения данной лабораторной работы я рассмотрел и реализовал методы интерполяции Ньютона и Гаусса для заданной таблицы данных. С помощью разработанной программы были вычислены приближенные значения функции для заданных аргументов с использованием методов Ньютона и Гаусса. Было проведено сравнение результатов, полученных разными методами.