Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа №5
по дисциплине «Вычислительная математика»

Вариант 16

Преподаватель:

Машина Е.А.

Выполнила:

Шайхутдинова Н.В.

P3208

Санкт-Петербург

2024 г.

Цель лабораторной работы

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

Рабочие формулы методов

Многочлен Лагранжа

$$L_n(x) = \sum_{i=0}^{n} y_i \, l_i(x)$$

$$Ln(x) = \sum_{i=0}^{n} y_i \frac{(x-x_0)(x-x_1)\dots(x-x_{i-1})(x-x_{i+1})\dots(x-x_n)}{(x_i-x_0)(x_i-x_1)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_n)}$$

$$L_n(x) = \sum_{i=0}^{n} y_i \prod_{\substack{j=0\\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j}$$

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

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

Для правой половины отрезка разности вычисляют справа налево: $t = (x - x_n)/h$. Тогда получим формулу Ньютона, которая называется **второй интерполяционной формулой Ньютона для интерполирования назад:**

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

Первая интерполяционная формула Гаусса $(x > a)$

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

Вторая интерполяционная формула Гаусса $(x < a)$

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

Порядок выполнения работы

Вычислительная реализация задачи

Лаб. №5

Вариант 16

x	0,25	0,3	0,35	0,4	0,45	0,5	0,55
y	1,2557	2,1764	3,1218	4,0482	5,9875	6,9195	7,8359

$X_1 = 0{,}534$

$X_2 = 0{,}384$

2) Таблица конечных разностей: Excel

x_i	y_i	Dy_i	D^2y_i	D^3y_i	D^4y_i	D^5y_i	D^6y_i
0,25	1,2557	0,9207	0,0247	-0,0437	1,0756	-4,1277	10,1917
0,3	2,1764	0,9454	-0,019	1,0319	-3,0521	6,064	
0,35	3,1218	0,9264	1,0129	-2,0202	3,0119		
0,4	4,0482	1,9393	-1,0073	0,9917			
0,45	5,9875	0,932	-0,0156				
0,5	6,9195	0,9164					
0,55	7,8359						

3) Формула Ньютона для интерполирования назад:

$$t = \frac{x - x_n}{h} = \frac{0{,}534 - 0{,}55}{0{,}05} = -0{,}32$$

$$N_6(x) = y_6 + t\Delta y_5 + \frac{t(t+1)}{2!}\Delta^2 y_4 + \frac{t(t+1)(t+2)}{3!}\Delta^3 y_3 + \frac{t(t+1)(t+2)(t+3)}{4!}\Delta^4 y_2 + \frac{t(t+1)(t+2)(t+3)(t+4)}{5!}\Delta^5 y_1 + \frac{t(t+1)(t+2)(t+3)(t+4)(t+5)}{6!}\Delta^6 y_0 \Rightarrow$$

$$N(0{,}534) = 7{,}8359 + (-0{,}32)\cdot 0{,}9164 + \frac{(1-0{,}32)(-0{,}32)}{2}\cdot(-0{,}0156) + \frac{(1-0{,}32)(2-0{,}32)(-0{,}32)}{3\cdot 2}\cdot(0{,}9917) + \frac{(3-0{,}32)(2-0{,}32)(1-0{,}32)(-0{,}32)}{4\cdot 6}\cdot 3{,}0119 + \frac{(4-0{,}32)(3-0{,}32)(2-0{,}32)(1-0{,}32)\cdot(-0{,}32)}{20\cdot 6}\cdot 6{,}064 + \frac{(5-0{,}32)(4-0{,}32)(3-0{,}32)(2-0{,}32)(1-0{,}32)(-0{,}32)}{6\cdot 20\cdot 6}\cdot 10{,}1917 = \underline{6{,}889942}$$

4) $a = 0{,}4$, $X_2 = 0{,}384$ } $X_2 < a \Rightarrow$ вторая интерполяционная формула Гаусса.

$$t = \frac{(x - x_0)}{h} = \frac{0{,}384 - 0{,}4}{0{,}05} = -0{,}32$$

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

$$P(0{,}384) = 4{,}0482 + (-0{,}32)\cdot 0{,}9264 + \frac{(-0{,}32)(1-0{,}32)\cdot 1{,}0129}{2} + \frac{(0{,}32)(1-0{,}32)(0{,}32+1)\cdot 1{,}0319}{6} + \frac{(2-0{,}32)(1-0{,}32)(-0{,}32)(0{,}32+1)}{24}\cdot(-3{,}0521) + \frac{(2-0{,}32)(1-0{,}32)(-0{,}32)(0{,}32+1)(0{,}32+2)}{20\cdot 6}\cdot(-4{,}1277) + \frac{(3-0{,}32)(2-0{,}32)(1-0{,}32)(-0{,}32)(1+0{,}32)(2+0{,}32)}{30\cdot 4\cdot 6}\cdot 10{,}1917 = \underline{3{,}649554}$$

Программная реализация задачи

```
def getPolynomialNewton(xm, xa, ym, table, n):
    h = xm[1] - xm[0]
    fact = 1
```

```
    xp = xa - xm[0]
    sum = ym[0] + (table[0][1] / h) * xp
    for i in range(n - 2):
        xp *= xa - xm[i + 1]
        fact *= i + 2
        h *= xm[1] - xm[0]
        sum += (xp / (fact * h)) * table[i + 1][1]
    return sum


def f(xm, ym, x):
    if len(x) == 2:
        return (ym[x[1]] - ym[x[0]]) / (xm[x[1]] - xm[x[0]])
    else:
        x1 = copy.deepcopy(x)
        x2 = copy.deepcopy(x)
        x1.pop(-1)
        x2.pop(0)
        fx1 = f(xm, ym, x1)
        fx2 = f(xm, ym, x2)
        return (fx2 - fx1) / (xm[x[-1]] - xm[x[0]])


def newtonMethod1(ym, n, xa, xm):
    sum = ym[0]
    xi = []
    xi.append(0)
    xp = 1
    for i in range(n - 1):
        xi.append(i + 1)
        xp *= xa - xm[i]
        fx = f(xm, ym, xi)
```

```
        sum += fx * xp
    return sum


def newtonMethod2(xm, xa, ym, table, n):
    h = xm[1] - xm[0]
    if xa <= (xm[-1] - xm[0]) / 2:
        t = (xa - xm[0]) / h
        fact = 1
        sum = ym[0] + t * table[0][1]
        tp = t
        for i in range(n - 2):
            tp *= t - i - 1
            fact *= i + 2
            sum += (tp / fact) * table[i + 1][1]
    else:
        t = (xa - xm[-1]) / h
        fact = 1
        sum = ym[n - 1] + t * table[0][n - 1]
        tp = t
        for i in range(n - 2):
            tp *= t + i + 1
            fact *= i + 2
            sum += (tp / fact) * table[i + 1][n - 2 - i]
    return sum


def getTable(n, y):
    table = []
    checker = True
    it = n
    counter = 1
```

```
    while checker:
        a = []
        last = []
        s = "d^" + str(counter) + "y"
        a.append(s)
        for i in range(it - 1):
            a.append(y[i + 1] - y[i])
            last.append(y[i + 1] - y[i])
        table.append(a)
        for i in range(len(table[-1]), n + 1):
            table[-1].append("-")
        y = last
        if it == 2:
            checker = False
        it -= 1
        counter += 1
    return table


def genFunc(userchoice):
    answer = []
    print("Таблица конечных разностей:")
    n = userchoice[2]
    y = userchoice[3]
    table = getTable(n, y)
    x = PrettyTable()
    fields = []
    fields.append("i")
    x_row = []
    y_row = []
    x_row.append("x")
    y_row.append("y")
```

```
for i in range(n):
    fields.append(i)
    x_row.append(userchoice[1][i])
    y_row.append(userchoice[3][i])
x.field_names = fields
x.add_row(x_row)
x.add_row(y_row)
for i in range(len(table)):
    x.add_row(table[i])
x.border = True
x.header = True
x.padding_width = 1
print(x)

print("Метод Лагранжа:")
xm = userchoice[1]
ym = userchoice[3]
xa = userchoice[4]
sum = 0
for i in range(n):
    s1 = 1
    s2 = 1
    for j in range(n):
        if i != j:
            s1 *= xa - xm[j]
            s2 *= xm[i] - xm[j]
    sum += ym[i] * s1 / s2
print("y =", sum)
answer.append(sum)

print("Метод Ньютона с разделёнными разностями:")
sum = newtonMethod1(ym, n, xa, xm)
```

```
    print("y =", sum)
    answer.append(sum)


    print("Метод Ньютона с конечными разностями:")
    sum = newtonMethod2(xm, xa, ym, table, n)
    print("y =", sum)
    answer.append(sum)
    return answer
```

Результат программы

Введите "1" для ввода входных данных через консоль, "2" для чтения из файла, "3" для выбора функции, шага и интервала из предложенных, "4" для выхода из программы:

1

Введите от 4 до 10 значений x через пробел

0.1 0.2 0.3 0.4 0.5

Введите от 4 до 10 значений y через пробел

1.25 2.38 3.79 5.44 7.14

Введите значение аргумента, приближённое значение функции для которого вы хотите вычислить:

0.47

Таблица конечных разностей:

i	0	1	2	3	4
x	0.1	0.2	0.3	0.4	0.5
y	1.25	2.38	3.79	5.44	7.14
d^1y	1.13	1.4100000000000001	1.6500000000000004	1.6999999999999993	-
d^2y	0.28000000000000025	0.2400000000000002	0.04999999999998934	-	-
d^3y	-0.04000000000000036	-0.19000000000000128	-	-	-
d^4y	-0.15000000000000124	-	-	-	-

Метод Лагранжа:

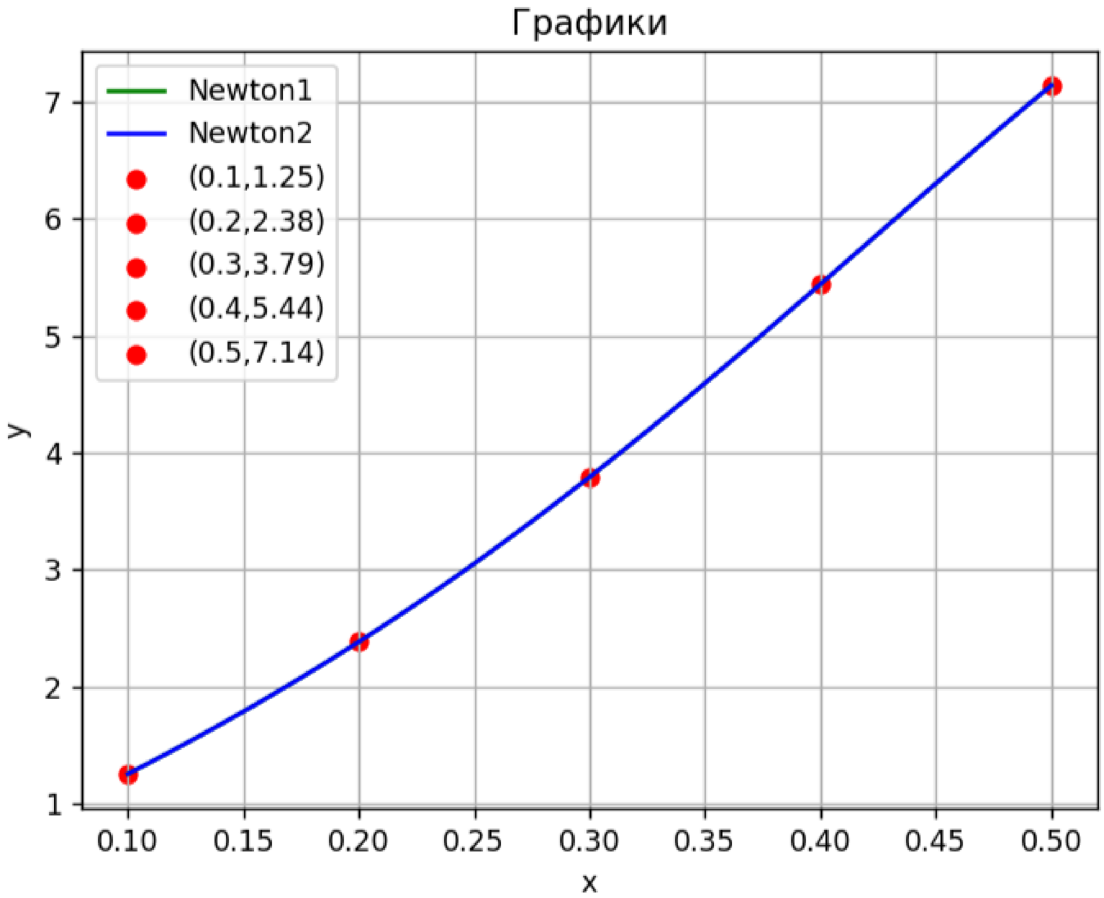
y = 6.642079375

Метод Ньютона с разделёнными разностями:

y = 6.642079374999999

Метод Ньютона с конечными разностями:

y = 6.642079374999999



Вывод

В ходе выполнения лабораторной работы я изучила методы для решения задачи интерполяции такие как многочлен Лагранжа, многочлен Ньютона, многочлен Гаусса и нашла с помощью них значения функции при заданных значениях аргумента, отличных от узловых точек.