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

Факультет Программной Инженерии и Компьютерной Техники

Лабораторная работа по вычислительной математике №5

Интерполяция функции

Преподаватель: Малышева Татьяна Алексеевна

Выполнила: Голованова Дарья Владимировна

Группа: Р3222

Санкт-Петербург,
2022г

Цель лабораторной работы:

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

Для исследования использовать:

- многочлен Лагранжа;
- многочлен Ньютона;
- многочлен Гаусса.

Описание использованного метода:

Интерполяцией называют такую разновидность аппроксимации, при которой кривая построенной функции проходит через имеющиеся точки.

Многочлен Ньютона с конечными разностями:

Интерполирующий полином ищется в виде:

$$P_n(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_n(x - x_0) \dots (x - x_{n-1})$$

Построение многочлена сводится к определению коэффициентов a_i При записи коэффициентов пользуются конечными разностями. Конечные разности первого порядка пишутся в виде:

$$\Delta^k y_0 = \Delta^{k-1} y_1 - \Delta^{k-1} y_0$$

$$\Delta^k y_1 = \Delta^{k-1} y_2 - \Delta^{k-1} y_1$$

$$\dots$$

$$\Delta^k y_{n-2} = \Delta^{k-1} y_{n-1} - \Delta^{k-1} y_{n-2}$$

Коэффициенты anнаходятся из Pn $(x_i) = y_i$. Находим a0полагая, что x = x_0

$$a_0 = P(x_0) = y_0$$

Далее подставляем значение x = x_1, получим

$$P_n(x_1) = y_1 = y_0 + a_1(x - x_0)$$

$$a_1 = \frac{y_1 - y_0}{x_1 - x_0} = \frac{\Delta y_0}{h}$$

Общая формула нахождения a_i:

$$a_i = \frac{\Delta^i y_0}{i!\, h^i}$$

В результате самая первая формула примет вид:

$$P_n(x) = y_0 + \frac{\Delta \quad y_0}{1!\,h}(x - x_0) + \frac{\Delta^2 y_0}{2!\,h^2}(x - x_0)(x - x_1) + \cdots + \frac{\Delta^i y_0}{i!\,h^i}(x - x_0)\ldots(x - x_{n-1})$$

Данный многочлен называют первым полиномом Ньютона.

Многочлен Лагранжа:

Интерполяционный многочлен Лагранжа — многочлен минимальной степени, принимающий данные значения в данном наборе точек. Для n+1 пар чисел $(x_0, y_0), (x_1, y_1) \ldots (x_n, y_n)$, где все x_i различны, существует единственный многочлен L(x) степени не более n, для которого $L(x_i) = y_i$.

Лагранж предложил способ вычисления таких многочленов:

$$L(x) = \sum_{j=0}^{n} y_j l_j(x)$$

где базисные полиномы определяются по формуле:

$$l_j(x) = \prod_{i=0, j\neq i}^{n} \frac{x - x_i}{x_j - x_i} = \frac{x - x_0}{x_j - x_0} \cdots \frac{x - x_{j-1}}{x_j - x_{j-1}} \frac{x - x_{j+1}}{x_j - x_{j+1}} \cdots \frac{x - x_n}{x_j - x_n}$$

Легко видеть, что $l_j(x)$ обладают такими свойствами:

- Это полиномы степени n
- $l_j(x_j) = 1$
- $l_j(x_i) = 0$ при $i \neq j$

Отсюда следует, что L(x), как линейная комбинация $l_j(x)$, может иметь степень не больше n, и $L(x_j) = y_j$

Вычислительная реализация:

$x = 0,55$

$y = 2,5356$

$x_1 = 0,751$

$x_2 = 0,651$

N	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	0,5	1,5320	1,0036	0,0014	−0,0008	−0,0012	0,0059	−0,0166
1	0,55	2,5356	1,0050	0,0006	−0,002	0,0047	−0,0107	
2	0,6	3,5406	1,0056	−0,0014	0,0027	−0,006		
3	0,65	4,5462	1,0042	0,0013	−0,0033			
4	0,7	5,5504	1,0055	−0,002				
5	0,75	6,5559	1,0035					
6	0,8	7,5594						

Воспользуюсь 2-ой интер-й формой Ньютона для интерполирования назад:

Для $x_1 = 0,751$:

$$t = \frac{(x - x_n)}{h} = \frac{0,751 - 0,8}{0,05} = -0,98$$

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

$$+ \frac{t(t+1)(t+2)(t+3)(t+4)}{5!}\Delta^5 y_1 + \frac{t(t+1)(t+2)(t+3)(t+4)(t+5)}{6!}\Delta^6 y_0 =$$

$$N_6(x) = 7,5594 + (-0,98)\cdot 1,0035 + \frac{(-0,98(-0,98+1)}{2}\cdot(-0,002) + \frac{(-0,98(-0,98+1)(-0,98+2)}{3!}(-0,003)$$

$$+ \frac{(-0,98(-0,98+1)(-0,98+2)(-0,98+3)}{4!}\cdot(-0,006) + \frac{(-0,98(-0,98+1)(-0,98+2)(-0,98+3)(-0,98+4)}{5!}\cdot(-0,0107) +$$

$$+ \frac{(-0,98(-0,98+1)(-0,98+2)(-0,98+3)(-0,98+4)(-0,98+5)}{6!}\cdot(-0,0166) = 7,5594 + (-0,98343) +$$

$$+ (1,96\cdot 10^{-5}) + (1,09956\cdot 10^{-5}) + (1,0096\cdot 10^{-5}) + (1,08747\cdot 10^{-5}) +$$

$$+ (6,78215\cdot 10^{-5}) \approx 6,5760$$

Воспользуюсь формулой Гаусса для $x_2 = 0{,}651 > x_0 = 0{,}650 \Rightarrow$

$\Rightarrow$ 1-ая интерполяционная формула для интер-и вперёд

$x_2 = 0{,}651$

N	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	0,5	1,5320	1,0036	0,0014	-0,0008	-0,0012	0,0059	-0,0166
-2	0,55	2,5356	1,0050	0,0006	-0,002	0,0047	-0,0107	
-1	0,6	3,5406	1,0056	-0,0014	0,0027	-0,006		
0	0,65	4,5462	1,0042	0,0013	-0,0033			
1	0,7	5,5504	1,0055	-0,002				
2	0,75	6,5559	1,0035					
3	0,8	7,5594						

$$t = \frac{(x - x_0)}{h} = \frac{0{,}651 - 0{,}65}{0{,}05} = 0{,}02$$

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

$$+ \frac{t(t+1)(t-1)(t-2)}{4!}\Delta^4 y_{-2} + \frac{t(t+1)(t+2)(t-1)(t-2)}{5!}\Delta^5 y_{-2} +$$

$$+ \frac{t(t+1)(t+2)(t-1)(t-2)(t-3)}{6!}\Delta^6 y_{-3};$$

$$y(0{,}651) \approx 4{,}5462 + 0{,}02 \cdot 1{,}0042 + \frac{0{,}02(0{,}02-1)}{2!}(-0{,}0014) +$$

$$+ \frac{0{,}02(0{,}02+1)(0{,}02-1)}{3!}0{,}0027 + \frac{0{,}02(0{,}02+1)(0{,}02-1)(0{,}02-2)}{4!} \cdot 0{,}0047 +$$

$$+ \frac{0{,}02(0{,}02+1)(0{,}02+2)(0{,}02-1)(0{,}02-2)}{5!}(-0{,}0107) +$$

$$+ \frac{0{,}02(0{,}02+1)(0{,}02+2)(0{,}02-1)(0{,}02-2)(0{,}02-3)}{6!}(-0{,}0166) \approx 4{,}5662$$

Листинг численного метода:

```
def newton(x: list, y: list, x0: float):
    if not check_nodes(x):
        raise Exception('Узлы не являются равноотстоящими, метод Ньютона с конечными разностями не применим.')
    if x0 in x:
        return y[x.index(x0)], ""

    dy = get_finite_differences(id(y))
    h = (x[1] - x[0])
    nearest_point = -1
    for index in range(len(x)):
        if x[index] >= x0:
            nearest_point = index - 1
            break
    result = 0
    if x0 - x[0] < x[-1] - x0:
        t = (x0 - x[nearest_point]) / h
        for i in range(len(dy) - nearest_point):
            result += dy[nearest_point][i] * get_t(i, t)
    else:
        try:
            nearest_point += 1
            t = (x0 - x[nearest_point]) / h
            for i in range(nearest_point, -1, -1):
                result += dy[i][nearest_point - i] * get_t(nearest_point - i, t, back=True)
        except Exception:
            return -10, ""
    return result, ""


def lagrange(x: list, y: list, x0: float):
    result = 0
    for j in range(len(y)):
        mul = 1
        for i in range(len(x)):
            mul *= (x0 - x[i]) / (x[j] - x[i]) if i != j else 1
        result += y[j] * mul
    return result, ""


# Функция для вычисления значения с использованием
# Формула Стирлинга (нечет)

def stirling(x, fx, x1):
    message = ""
    d = 1
    temp1 = 1
    temp2 = 1
```

```
    k = 1
    n = len(fx)
    l = 1

    delta = [[0 for i in range(n)] for j in range(n)]
    h = x[1] - x[0]
    s = math.floor(n / 2)
    a = x[s]
    t = (x1 - a) / h
    if abs(t) > 0.25:
        message = "не выполняется условие |t|<=0.25"
    # заполнение таблички разностей
    for i in range(n - 1):
        delta[i][0] = fx[i + 1] - fx[i]

    for i in range(1, n - 1):
        for j in range(n - i - 1):
            delta[j][i] = (delta[j + 1][i - 1] - delta[j][i - 1])
    # Расчет f (x) по формуле Стирлинга
    y1 = fx[s]
    for i in range(1, n):
        if i % 2 != 0:
            temp1 *= (pow(t, k) - pow((k - 1), 2))
            k += 1
            d *= i
            s = math.floor((n - i) / 2)
            y1 += (temp1 / (2 * d)) * (delta[s][i - 1] + delta[s - 1][i -
1])
        else:
            temp2 *= (pow(t, 2) - pow((l - 1), 2))
            l += 1
            d *= i
            s = math.floor((n - i) / 2)
            y1 += (temp2 / d) * (delta[s][i - 1])
    return y1, message


# используя интерполяцию Бесселя (чет)

def ucal(u, n):
    if n == 0:
        return 1
    temp = u
    for i in range(1, int(n / 2 + 1)):
        temp = temp * (u - i)
    for i in range(1, int(n / 2)):
        temp = temp * (u + i)
    return temp


# расчет факториала
```

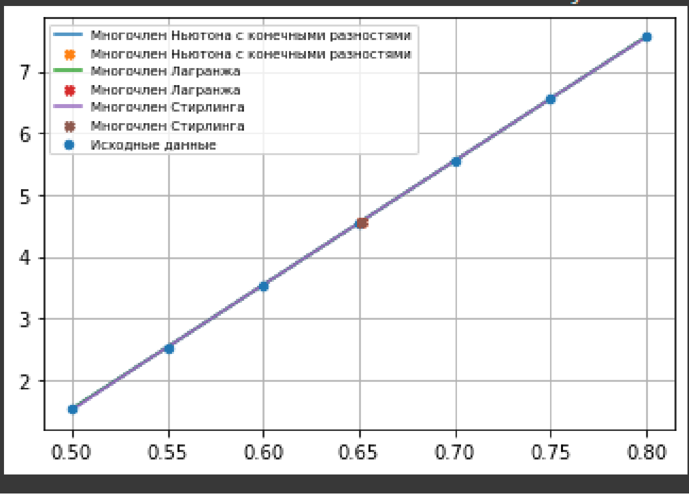
```
# заданный номер n

def fact(n):
    f = 1
    for i in range(2, n + 1):
        f *= i
    return f


# Расчет центрального
# таблица разностей
def bessel(x, y_vals, value):
    message = ""
    n = len(x)
    y = [[0 for i in range(n)] for j in range(n)]
    for i in range(n):
        y[i][0] = y_vals[i]
    for i in range(1, n):
        for j in range(n - i):
            y[j][i] = y[j + 1][i - 1] - y[j][i - 1]

    # Инициализация и сумма
    summary = (y[n // 2 - 1][0] + y[n // 2][0]) / 2
    # k является источником, то есть f (0)
    if (n % 2) > 0:  # происхождение для нечетных
        k = int(n / 2)
    else:
        k = int(n / 2 - 1)  # происхождение даже

    t = (value - x[k]) / (x[1] - x[0])
    if not 0.25 <= abs(t) <= 0.75:
        message = "не выполняется условие 0.25<=|t|<= 0.75"

    # Решение по формуле Бесселя
    for i in range(1, n):
        if i % 2:
            summary = summary + ((t - 0.5) * ucal(t, i - 1) * y[k][i]) / f
act(i)
        else:
            summary = summary + (ucal(t, i) * (y[k][i] + y[k - 1][i]) / (f
act(i) * 2))

    return summary, message
```

Пример работы программы:

Конечные разности:

yi	d1 Yi	d2 Yi	d3 Yi	d4 Yi	d5 Yi	d6 Yi
1.5320	1.0036	0.0014	-0.0008	-0.0012	0.0059	-0.0166
2.5356	1.0050	0.0006	-0.0020	0.0047	-0.0107	
3.5406	1.0056	-0.0014	0.0027	-0.0060		
4.5462	1.0042	0.0013	-0.0033			
5.5504	1.0055	-0.0020				
6.5559	1.0035					
7.5594						

Многочлен Ньютона с конечными разностями: 4.566306403192002
Многочлен Лагранжа: 4.566294839427489
Многочлен Стирлинга: 4.566294839270752
Многочлен Бесселя: нечётное количество узлов



Вывод:

Интерполяцию применяют в случае, когда требуется найти значение функции y(x) при значении аргумента xi принадлежащего интервалу [x0, … xn], но не совпадающему по значению ни с одним табличным значением этой функции. Графически задача интерполяции заключается в том, чтобы построить такую функцию, которая бы проходила через все заданные точки (узлы).

Интерполяция бывает:

- **Каноническим полиномом**. Задача интерполяции сводится к решению СЛАУ для получения коэффициентов полинома.
- **Линейная интерполяция**. Просто соединить заданные точки прямыми. Простейший метод интерполяции.
- **Интерполяция полиномом Лагранжа**.

$$P_n(x) = \sum_{i=0}^{n} y_i \left(\prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{x - x_k}{x_i - x_k} \right)$$

Интерполяционный полином Лагранжа обычно применяется в теоретических исследованиях (при доказательстве теорем, аналитическом решении задач и т.п.). Минусом данного метода является то, что при добавлении точек происходит перерасчет всего многочлена.

- **Интерполяция полиномом Ньютона**. Интерполяционные формулы ньютона удобно использовать, если точка интерполяции находится в начале таблицы или в конце таблицы. Построение полинома также как и в варианте 1 сводится к определению коэффициентов ai.

Интерполяция каноническим полиномом требует бОльших вычислительных мощностей. Линейная интерполяция крайне неточная. Интерполяция методом Лагранжа хороша, если количество точек не изменяется. Интерполяция полиномом Ньютона хороша, если точки находятся в начале/конце таблицы точек.