МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ «ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по практической работе №10

по дисциплине «Вычислительная математика»

Тема: Использование интерполяционной формулы в вычислении значения заданной функции

Студент гр. 8383	___________________	Ларин А.
Преподаватель	___________________	Сучков А.И.

Санкт-Петербург

2019

Цель работы.

Исследование различных методов интерполяции для равноотстоящих узлов с последующей реализацией на одном из языков программирования.

Основные теоретические положения.

Если значения функции $f_k = f(x_k)$ заданы в точках $x_k = x_0 + kh$ $(k = \overline{0, n})$ с постоянным положительным шагом, то часто используется интерполяционный многочлен Ньютона для интерполяции вперёд:

$$L_n(x) = L_n(x_0 + qh) = f_0 + \frac{q}{1!}\Delta f_0 + \frac{q(q-1)}{2!}\Delta^2 f_0 + \cdots + \left(\prod_{i=0}^{n-1}(q-i)\right)\frac{\Delta^n f_0}{n!},$$

где $q = \frac{x - x_0}{h}$, а конечные разности $\Delta^r f_0$, носящие названия нисходящих разностей, находят из соотношений

$$\Delta f_k = f_{k+1} - f_k,$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$\Delta^r f_k = \Delta^{r-1} f_{k+1} - \Delta^{r-1} f_k = \sum_{j=0}^{r}(-1)^j \binom{r}{j} f_{k+r-j}.$$

Интерполяционный многочлен для интерполяции вперёд удобно использовать при работе в начале таблицы значений функции и для экстраполяции левее точки x_0.

Интерполяционный многочлен с узлами $x_0, x_{-1}, \ldots, x_{-n}$ где $x_{-k} = x_0 - kh$, имеет вид:

$$L_n(x) = L_n(x_0 + qh) = f_0 + \frac{q}{1!}\nabla f_0 + \frac{q(q+1)}{2!}\nabla^2 f_0 + \cdots + \left(\prod_{i=0}^{n-1}(q+i)\right)\frac{\nabla^n f_0}{n!}$$

и называется интерполяционным многочленом Ньютона для интерполяции назад. Его удобно использовать при интерполяции в конце таблицы и для экстраполяции правее точки x_0. Входящие в выражение значения конечных восходящих разностей находят из соотношений

$$\nabla f_k = f_k - f_{k-1} = \Delta f_{k-1},$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$\nabla^r f_k = \nabla^{r-1} f_k - \nabla^{r-1} f_{k-1} = \Delta^r f_{k-r}.$$

Если при заданном x в таблице значений функции f с шагом h имеется достаточное число узлов с каждой стороны от x, то целесообразно узлы интерполяции $x_0, x_1, \dots, x_n$ выбрать так, чтобы точка x оказалась как можно ближе к середине минимального отрезка, содержащего узлы. При этом обычно в качестве x_0 берется ближайший к x узел, затем за x_1 принимается ближайший к x узел, расположенный с противоположной от x стороны, чем x_0. Следующие узлы назначаются поочередно с разных сторон от x и должны быть расположены как можно ближе к x. Одной из возможных схем интерполяции в этом случае является схема Стирлинга с интерполяционным многочленом вида

$$\begin{aligned} L_n(x) &= L_n(q) \\ &= f_0 + q\frac{\Delta f_{-1} + \Delta f_0}{2} + \frac{q^2}{2!}\Delta^2 f_{-1} + \frac{q(q^2-1^2)}{3!}\frac{\Delta^3 f_{-2} + \Delta^3 f_{-1}}{2} \\ &+ \frac{q^2(q^2-1^2)}{4!}\Delta^4 f_{-2} + \dots + \frac{q}{(2n-1)!}\left(\prod_{k=1}^{n-1}(q^2-k^2)\right) \\ &\cdot \frac{\Delta^{2n-1} f_{-n} + \Delta^{2n-1} f_{-(n-1)}}{2} + \frac{q^2}{(2n)!}\left(\prod_{k=1}^{n-1}(q^2-k^2)\right)\cdot \Delta^{2n} f_{-n}. \end{aligned}$$

В этом выражении учитывается, что дано нечетное число $n+1 = 2m+1$ значений функции $f_k = f(x_0 + kh)$, где $k = 0, \pm 1, \pm 2, \dots, \pm m$. Обычно эту формулу целесообразно использовать при $|q| \leqslant 0.25$.

Постановка задачи.

В соответствии с заданием, полученным от преподавателя, студентам необходимо разработать программу, обеспечивающую вычисление значения функции в заданных точках с использованием подходящих для каждого конкретного случая интерполяционных формул. Используя подходящую интерполяционную формулу, вычислите в точках $\hat{x}_1$, $\hat{x}_2$, $\hat{x}_3$ значения функции, заданной таблицей, для узлов с равноотстоящим шагом. Порядок выполнения работы следующий:

1. Определить расположение заданных точек на сетке и, исходя из этого, выбрать методы нахождения значений в искомых узлах.
2. Составить необходимые подпрограммы-функции: `NFORWARD` (Ньютон вперёд), `NBACKWARD` (Ньютон назад), `STIRLING` (Стирлинг) – для заданного варианта.
3. Составить головную программу, содержащую обращение к соответствующим подпрограммам и осуществляющую печать результатов (в том числе и промежуточных вычислений) как на экран, так и в файл. Входные данные также считываются из файла.
4. Провести вычисления по программе. Построить множество точек, соединённых последовательно, отметить искомые узлы.

Выполнение работы.

Интерполируем значения неизвестной функции $f(x)$ по набору значений представленных в табл. 1.

Таблица 1 – Данный набор точек неизвестной функции $f(x)$

Номер i	Значение x_i	Значение y_i
0	0.3310	-0.3630
1	0.5290	-0.0900
2	0.7280	0.0210
3	0.9270	0.0180
4	1.1260	-0.0520
5	1.3240	-0.1440
6	1.5230	-0.2110
7	1.7220	-0.2040
8	1.9210	-0.0770
9	2.1190	0.2150
10	2.3180	0.7240

Требуется интерполировать значение функции в точках $\hat{x}_1 = 0.6190, \hat{x}_2 = 1.3990, \hat{x}_3 = 1.1000$

Визуализируем данные значения на графике. График представлен на рис. 1.

Была написана программа для интерполяции функции многочленом Ньютона для интерполяции вперед и назад. Она принимает на вход следующие значения: `n` – количество известных точек, `x` – интерполируемая точка, `xx, yy` – список известных точек, их абсциссы и ординаты соответственно. Программа выводит слагаемые многочлена и интерполированное значение функции в точке в стандартный поток вывода. Код программы представлен в приложении А.

Аппроксимированная при помощи многочлена Ньютона для интерполяции вперед функция представлена на графике на рис. 2, для интерполяции назад – на графике на рис. 3.

Была написаны программы для интерполяции функции многочленом Стирлинга. Она принимает на вход следующие значения: `x` – интерполируемая точка, `x0` – относительный узел для интерполяции, `xx, yy` – список известных точек, их абсциссы и ординаты соответственно. Программа выводит слагаемые многочлена и интерполированное значение функции в точке в стандартный поток вывода. Код программы представлен в приложении А.

Аппроксимированные при помощи многочлена Стирлинга функция для точек $\hat{x}_2, \hat{x}_3$ представлены на графиках на рис. 4, 5 соответственно.

Подберем необходимые методы для интерполяции каждой точки. Метод Стирлинга следует применять при $|q| = \left|\frac{x-x_0}{h}\right| \leq 0.25$. Это верно для точек $\hat{x}_2, \hat{x}_3 (q \approx 0.13)$. Слагаемые многочлена Стирлинга и результат интерполяции представлены в табл. 2.

Многочлен Ньютона для интерполяции вперед подходит для точек расположенных близ узлов в начале таблицы. Используем его для интерполяции точек $\hat{x}_1, \hat{x}_3$. Слагаемые многочлена Ньютона для интерполяции вперед и результат интерполяции представлены в табл. 3.

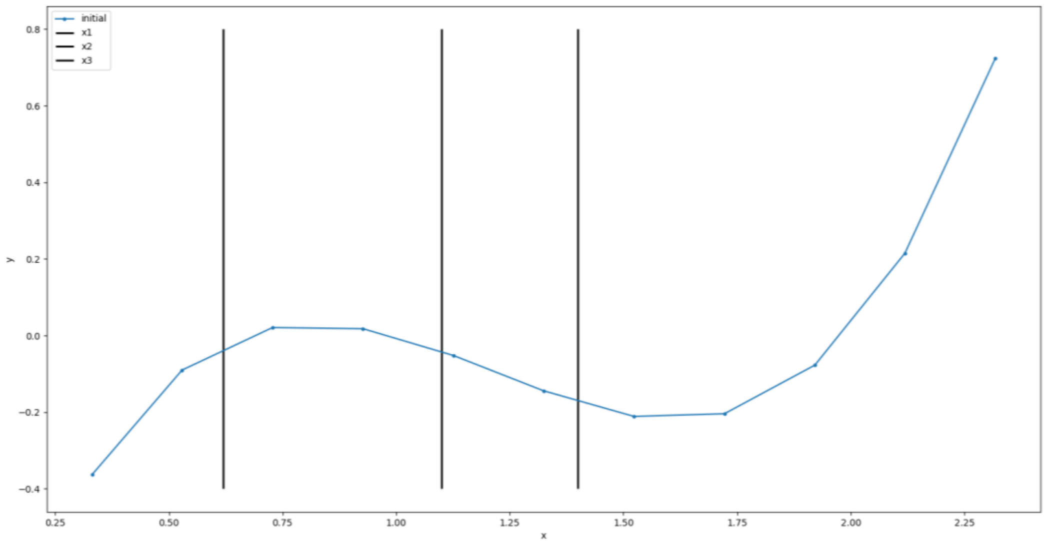

Рисунок 1 – Исходный набор точек функции

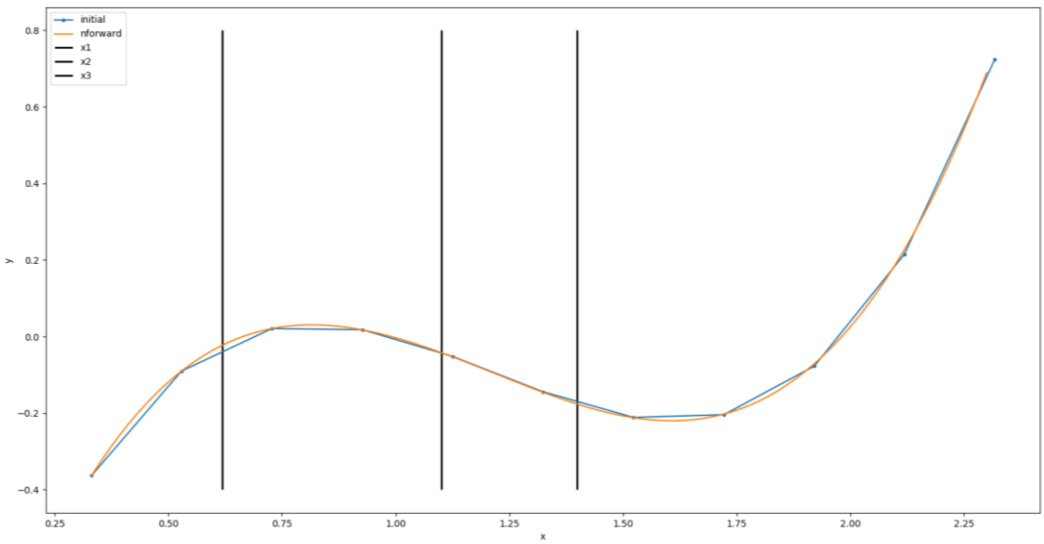

Рисунок 2 – Интерполяция многочленом Ньютона для интерполяции вперед

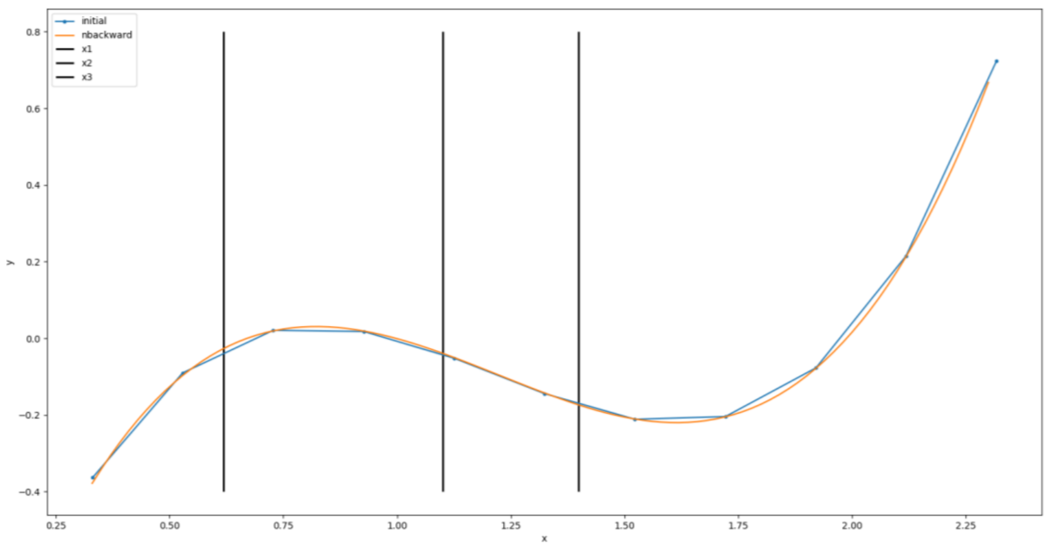

Рисунок 3 – Интерполяция многочленом Ньютона для интерполяции назад

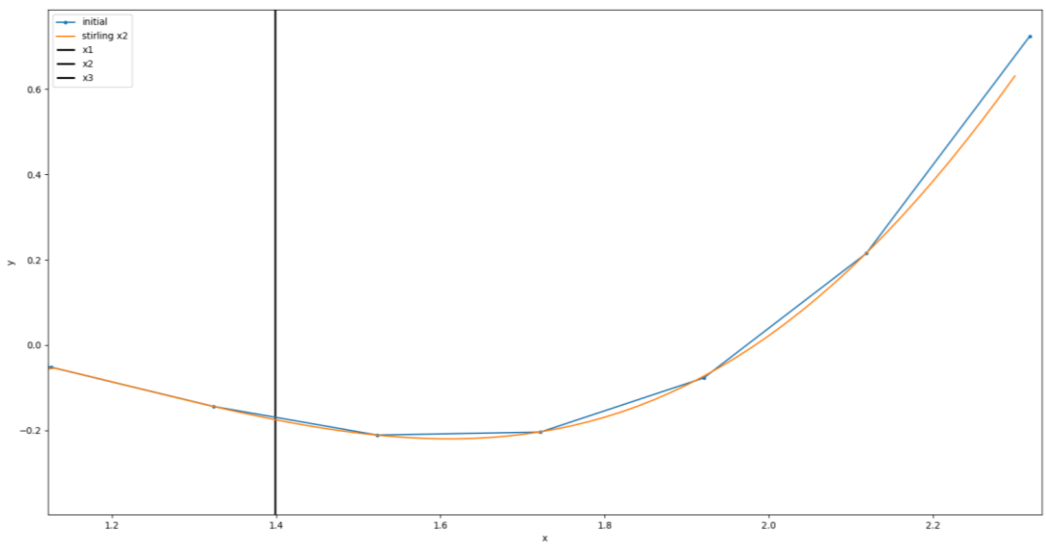

Рисунок 4 – Интерполяция многочленом Стирлинга относительно узла $\hat{x}_2$

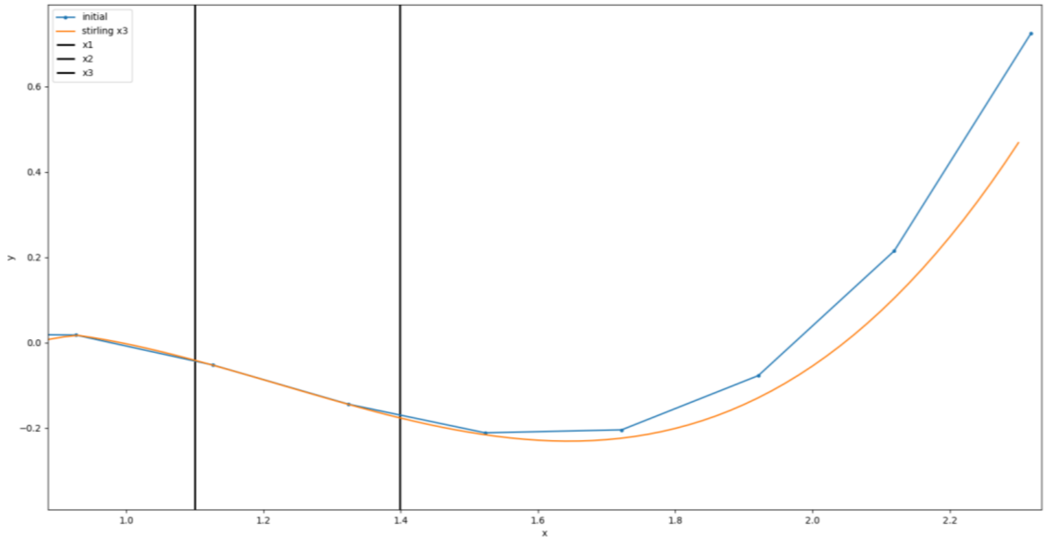

Рисунок 5 – Интерполяция многочленом Стирлинга относительно узла $\hat{x}_3$

Таблица 2 – Значения слагаемых s_i многочлена Ньютона для интерполяции вперед(nforward) и назад(nbackward), и интерполированного значения в точке - f

	nforward			nbackward		
	$\hat{x}_1$	$\hat{x}_2$	$\hat{x}_3$	$\hat{x}_1$	$\hat{x}_2$	$\hat{x}_3$
s_1	3,9709E-01	1,4725E+00	1,0603E+00	-4,3676E+00	-2,3625E+00	-3,1311E+00
s_2	-5,3554E-02	-1,9198E+00	-9,0723E-01	7,0579E+00	1,8338E+00	3,4383E+00
s_3	-2,8850E-03	6,4351E-01	1,6880E-01	-3,7100E+00	-3,8691E-01	-1,1402E+00
s_4	-2,3222E-05	-8,0235E-03	-7,7703E-04	6,9680E-01	2,1373E-02	1,2093E-01
s_5	1,1822E-05	-2,2369E-03	1,8052E-05	-7,2958E-01	-3,1334E-03	-5,9470E-02
s_6	3,4930E-05	7,3432E-04	1,6791E-05	3,2656E-01	-1,4045E-04	8,5600E-03
s_7	4,0827E-05	1,1444E-04	9,1369E-06	2,0066E-02	4,5432E-06	3,0880E-05
s_8	2,5156E-05	2,0422E-05	3,1636E-06	-5,1547E-02	1,7413E-05	4,2577E-05
s_9	-1,1435E-05	-3,6959E-06	-9,0429E-07	6,6531E-03	1,2996E-05	1,7490E-05
s_{10}	-5,3493E-05	-8,2631E-06	-2,8684E-06	3,3252E-04	6,7538E-06	5,9400E-06
f	-2,2323E-02	-1,7610E-01	-4,1878E-02	-2,6489E-02	-1,7346E-01	-3,8867E-02

Таблица 2 – Значения слагаемых s_i многочлена Стирлинга для интерполяции (stirling), и интерполированного значения в точке - f

	stirling	
	$\hat{x}_2$	$\hat{x}_3$
s_1	-2,83201E-02	-1,08260E-02

Окончание таблицы 2

s_2	-2,60576E-03	-9,90787E-04
s_3	-2,93581E-05	8,19122E-06
s_4	-1,34461E-05	-
f	-1,74969E-01	-6,38086E-02

Выводы.

Проанализировав результаты аппроксимации графиков по формуле Ньютона для интерполяции вперед можно прийти к выводу, что точность аппроксимации снижается для точек близ узлов в конце таблицы. То же верно для формулы Ньютона для интерполяции назад для узлов в начале таблицы.

По графикам, построенным по многочленам Стирлинга можно сказать, что они достаточно хорошо аппроксимируют значения графика в окрестностях узла, и тем хуже, чем дальше от искомого узла удаляется значение.

При сравнении интерполированных по формуле Ньютона для интерполяции вперед и по многочлену Стирлинга видно, что значение дальние от узлов в начале таблицы различаются сильнее. И тем сильнее, чем дальше. Оценив визуально(по графикам), что значения интерполированные по многочлену Стирлинга интерполируют функцию точнее, возьмем их за эталон и оценим разницу. Разница составляет порядка $1 * 10^{-3}$, то есть совпадают два знака после запятой. Значение $\hat{x}_1$ по методу Стирлинга не оценивалось, т.к. для данной точки недостаточно узлов с левой стороны для оценки.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
import matplotlib.pyplot as plt
import math




def descdiv(r, k, yy):
    if (r == 1):
        return yy[k + 1] - yy[k]
    return descdiv(r - 1, k + 1, yy) - descdiv(r - 1, k, yy)


def ascdiv(r, k, yy):
    # yy=yy[::-1]
    if (r == 1):
        return yy[k] - yy[k - 1]
    return ascdiv(r - 1, k, yy) - ascdiv(r - 1, k - 1, yy)
    # return descdiv(r, k - r, yy)


def nforward(n, x, xx, yy):
    s = yy[0]
    h = abs(xx[1] - xx[0])
    q = (x - xx[0]) / h
    for i in range(1, n + 1):
        m = 1
        for j in range(i):
            m *= (q - j)
        m *= descdiv(i, 0, yy) / math.factorial(i)
        s += m
        print("m=\t" + str(m))
    return s


def nbackward(n, x, xx, yy):
    s = yy[-1]
    h = abs(xx[1] - xx[0])
    q = (x - xx[-1]) / h
    for i in range(1, n + 1):
        m = 1
        for j in range(i):
            m *= (q + j)
        m *= descdiv(i, n-i, yy) / math.factorial(i)
```

```
        s += m
        print("m=\t"+str(m))
    return s


def stirling(x, x0, xx, yy):
    h = abs(xx[1] - xx[0])
    x0-=h

    xy = [[xx[j], yy[j]] for j in range(len(xx))]

    _xy = [j for j in xy if j[0] < x0]
    xy_ = [j for j in xy if j[0] > x0]

    _xy.sort(key=lambda k: abs(k[0] - x0))
    xy_.sort(key=lambda k: abs(k[0] - x0))

    _xy = [_xy[j] for j in range(min(len(_xy), len(xy_)))]
    xy_ = [xy_[j] for j in range(min(len(_xy), len(xy_)))]

    _xy.sort(key=lambda x:x[0])

    xy=_xy+xy_

    '''
    if (abs(_xy[0][0] - x0) < abs(xy_[0][0] - x0)):
        xy = mergelists(_xy, xy_)
    else:
        xy = mergelists(xy_, _xy)
    '''
    xx = [j[0] for j in xy]
    yy = [j[1] for j in xy]
    '''
    min=abs(xx[0]-x)
    j0=0
    n=len(xx)-1
    for j in range(n+1):
        if(min>abs(xx[j]-x)):
            min = abs(xx[j]-x)
            j0=j

    if(j0>n-j0):
        n=n-j0
    else:
        n=j0
    '''
```

```
        n=int(len(xx)/2)
        s = yy[n]
        q = abs(x - xx[n]) / h
        for i in range(1, n):
            e=0
            m = 1
            m *= q
            m /= math.factorial(2 * i - 1)
            for j in range(1, i):
                m *= (q ** 2 - j ** 2)
            m *= (descdiv(2 * i - 1, n-i, yy) + descdiv(2 * i - 1,
n-(i - 1), yy)) / 2
            e+=m
            s += m
            m = 1

            m /= math.factorial(2 * i)
            for j in range(i):
                m *= (q ** 2 - j ** 2)
            m *= descdiv(2 * i, n-i, yy)
            e+=m
            s += m

            print("m=\t"+str(e))

        return s


    def mergelists(l1, l2):
        N = 2
        temp_iter = iter(l1)
        res = []
        for ele in l2:
            res.extend([next(temp_iter) for _ in range(N - 1)])
            res.append(ele)
        res.extend(temp_iter)
        return res



    def linSpace(a, b, s):
        x = []
        # x.append(a)
        e = 1 / s
        e0 = 0
        while e0 <= 1:
```

```
        _x = round(a + (b - a) * e0, int(abs(math.log10(e))))
        # _x=a+(b-a)*e0
        x.append(_x)
        e0 += e
    return x

x1=input()
x2=input()
x3=input()
n = input()
for i in range(n):
    x[i]=input()
for i in range(n):
    y[i]=input()

xx = linSpace(x[0], x[-1], 100)
#xx = linSpace(2, 2.5, 100)

yy = [nbackward(n, i, x, y) for i in xx]
#yy = [stirling(i, 1.0, x, y) for i in xx]

print("nfoward(x1)=\t"+str(nforward(n,x1,x,y)))
print("nfoward(x2)=\t"+str(nforward(n,x2,x,y)))
print("nfoward(x3)=\t"+str(nforward(n,x3,x,y)))

print("nbackward(x1)=\t"+str(nbackward(n,x1,x,y)))
print("nbackward(x2)=\t"+str(nbackward(n,x2,x,y)))
print("nbackward(x3)=\t"+str(nbackward(n,x3,x,y)))
plt.plot(x, y, marker=".")

# print([x[j] - x[j-1] for j in i[1:]])

plt.plot(xx, yy)
plt.rcParams['lines.linewidth'] = 2  # Синтакс 2
#
# for j in range(len(i)-1,-1,-1):
#     print(str(i[j])+"\t"+str(d[j]))
# plt.plot(i,d)
plt.vlines(x1, -0.4, 0.8)
plt.vlines(x2, -0.4, 0.8)
plt.vlines(x3, -0.4, 0.8)
plt.xlabel('x')
plt.ylabel('y')
plt.legend(['initial','staffff', 'x1','x2','x3'])
plt.show()
```