Приближение функций

Калитвин В.А.
kalitvinv@yandex.ru

2025

Интерполирование функций

1. Постановка задачи

Пусть известные значения некоторой функции образуют следующую таблицу

x	x_0	x_1	$\dots$	x_n
$f(x)$	y_0	y_1	$\dots$	y_n

(1)

При этом требуется получить значение функции f для такого значения аргумента x, которое входит в отрезок $[x_0, x_n]$, но не совпадает ни с одним из значений $x_i (i = 0, 1, \dots, n)$.

Применяется прием — построение по исходной информации приближающей функции F, которая в некотором смысле близка к функции f и аналитическим выражением которой можно воспользоваться для вычисления функции, считая приближенно, что $f(x) = F(x)$.

Классический подход к решению задачи построения приближающей функции основывается на требовании строгого совпадения значений $f(x)$ и $F(x)$ в т. x_i $(i = 0, 1, 2, \ldots, n)$, т.е.

$$F(x_0) = y_0, F(x_1) = y_1, \ldots, F(x_n) = y_n$$

В этом случае нахождение приближающей функции называют **интерполяцией (интерполированием)**, а точки $x_0, x_1, \ldots, x_n$ — узлами интерполяции.

Будем искать интерполирующую функцию $F(x)$ в виде многочлена степени n

$$P_n(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n \tag{2}$$

Этот многочлен имеет $n+1$ коэффициент

Условия (1) позволяют однозначно определить коэффициенты многочлена (2)

Требуя выполнения условий, получим систему $n+1$ уравнений с $n+1$ неизвестными

$$\begin{cases} a_0x_0^n + a_1x_0^{n-1} + \cdots + a_{n-1}x_0 + a_n = y_0 \\ a_0x_1^n + a_1x_1^{n-1} + \cdots + a_{n-1}x_1 + a_n = y_1 \\ \vdots \\ a_0x_n^n + a_1x_n^{n-1} + \cdots + a_{n-1}x_n + a_n = y_n \end{cases}$$

Решая эту систему относительно неизвестных $a_0, a_1, \ldots, a_n$, получим аналитическое выражение полинома.

Система имеет единственное решение, так как ее определитель

$$\begin{vmatrix} x_0^n & x_0^{n-1} & \dots & x_0 & 1 \\ x_1^n & x_1^{n-1} & \dots & x_1 & 1 \\ \vdots & & & & \\ x_n^n & x_n^{n-1} & \dots & x_n & 1 \end{vmatrix}$$

(определитель Вандермонда) отличен от нуля. Следовательно интерполяционный многочлен $P_n(x)$ для функции f, заданной таблично, существует и единственен. Интерполяционный многочлен имеет степень не большую, чем n.
Описанный прием можно использовать для решения задачи интерполирования, однако на практике используют другие, более удобные и менее трудоемкие методы.

Интерполяционный многочлен Лагранжа

Пусть функция f задана таблицей (1)
Построим интерполяционный многочлен $L_n(x)$, степень которого не больше n и для которого выполнены условия

$$F(x_0) = y_0, F(x_1) = y_1, \ldots, F(x_n) = y_n \tag{1'}$$

Будем искать $L_n(x)$ в виде

$$L_n(x) = l_0(x) + l_1(x) + \cdots + l_n(x) \tag{2'}$$

где $l_i(x)$ – многочлен степени n, причем

$$l_i(x_k) = \begin{cases} y_i, i = k, \\ 0, i \neq k \end{cases} \tag{3'}$$

Очевидно, что (3') с учетом (2') обеспечивает выполнение условий 1.

Многочлены $l_i(x)$ составим следующим образом

$$l_i(x) = c_i(x - x_0)(x - x_1)\dots(x - x_{i-1})(x - x_{i+1})\dots(x - x_n) \qquad (4')$$

где c_i — постоянный коэффициент, значение которого найдем из (2')

$$c_i = \frac{y_i}{(x_i - x_0)(x_i - x_1)\dots(x_i - x_{i-1})(x_i - x_{i+1})\dots(x_i - x_n)}$$

Подставим c_i в (4') и с учетом (2') получим

$$L_n(x) = \sum_{i=0}^{n} y_i \frac{(x - x_0)\cdot\ldots\cdot(x - x_{i-1})\cdot(x - x_{i+1})\cdot\ldots\cdot(x - x_n)}{(x_i - x_0)\cdot\ldots\cdot(x_i - x_{i-1})\cdot(x_i - x_{i+1})\cdot\ldots\cdot(x_i - x_n)}$$

Пример

Построить интерполяционный многочлен Лагранжа для функции, заданной таблицей

x	1	3	4
$f(x)$	12	4	6

Из таблицы видно, что $n = 2$

$$L_2(x) = 12 \cdot \frac{(x-3)(x-4)}{(1-3)(1-4)} + 4 \cdot \frac{(x-1)(x-4)}{(3-1)(3-4)} + 6 \cdot \frac{(x-1)(x-3)}{(4-1)(4-3)} =$$

$$= 2(x^2 - 7x + 12) - 2(x^2 - 5x + 4) + 2(x^2 - 4x + 3) =$$

$$= 2x^2 - 12x + 22.$$

Пример. Многочлен Лагранжа на C++

Листинг: Код на C++

```
#include <iostream>
using namespace std;
const int n=10;

double lagranz(double X[n], double Y[n], double t);

int main() {
double X[n]={2,5,-6,7,4,3,8,9,1,-2};
double Y[n]={-1,77,-297,249,33,9,389,573,-3,-21};
cout << lagranz(X,Y,7);
return 0;
}
```

Пример. Многочлен Лагранжа на C++

Листинг: Код на C++

```
double lagranz(double X[n], double Y[n], double t){
    double z,p1,p2;
    z=0;
    for (int j=0; j<n; j++){
        p1=1; p2=1;
        for (int i=0; i<n; i++){
            if (i==j){
                p1=p1*1;p2=p2*1;
            }
            else {
                p1=p1*(t-X[i]);
                p2=p2*(X[j]-X[i]);
            }
        }
        z=z+Y[j]*p1/p2;
    }
    return z;
}
```

Формуле Лагранжа можно придать более сжатый вид. Введем обозначения

$$\Pi_{n+1}(x) = (x - x_0)(x - x_1)\ldots(x - x_n)$$

$$\Pi'_{n+1}(x) = \sum_{i=0}^{n}(x - x_0)\ldots(x - x_{i-1})(x - x_{i+1})\ldots(x - x_n).$$

При $x = x_i$ $(i = 0, 1, \ldots, n)$

$$\Pi'_{n+1}(x_i) = (x_i - x_0)\ldots(x_i - x_{i-1})(x_i - x_{i+1})\ldots(x_i - x_n).$$

Тогда формула Лагранжа принимает вид

$$L_n(x) = \sum_{i=0}^{n} y_i \cdot \frac{\Pi_{n+1}(x)}{(x - x_i)\cdot\Pi'_{n+1}(x_i)} = \Pi_{n+1}(x)\sum_{i=0}^{n}\frac{y_i}{(x - x_i)\Pi'_{n+1}(x_i)}$$

Интерполяционные многочлены Ньютона для равноотстоящих узлов

Часто интерполирование ведется для функций, заданных таблицами с равноотстоящими значениями аргумента. В этом случае шаг таблицы $h = x_{i+1} - x_i$ $(i = 01, 2, \dots)$ является величиной постоянной. Для таких таблиц построение интерполяционных формул заметно упрощается.

Конечные разности

Пусть функция задана таблицей с постоянным шагом

x	x_0	x_1	x_2	$\dots$
y	y_0	y_1	y_2	$\dots$

Разности между значениями функции в соседних узлах интерполяции называются конечными разностями первого порядка

$$\Delta y_i = y_{i+1} - y_i (i = 01, 2, \dots)$$

Из конечных разностей первого порядка образуются конечные разности второго порядка

$$\Delta^2 y_i = y\Delta y_{i+1} - \Delta y_i (i = 01, 2, \dots)$$

Продолжая этот процесс, можно по заданной таблице функции составить таблицу конечных разностей

x	y	Δy_i	$\Delta^2 y_i$	$\Delta^3 y_i$	$\dots$
x_0	y_0	Δy_0	$\Delta^2 y_0$	$\Delta^3 y_0$	
x_1	y_1	Δy_1	$\Delta^2 y_1$	$\Delta^3 y_1$	
x_2	y_2	Δy_2	$\Delta^2 y_2$		
x_3	y_3	Δy_3			
x_4	y_4				
$\vdots$	$\vdots$				

Конечные разности любого порядка могут быть представлены через значения функции
Для разностей первого порядка это следует из определения
Для разностей второго порядка имеем:

$$\Delta^2 y_i = \Delta y_{i+1} - \Delta y_i =$$

$$= y_{i+2} - y_{i+1} - y_{i+1} + y_i = y_{i+2} - 2y_{i+1} + y_i$$

$$\Delta^3 y_i = \Delta^2 y_{i+1} - \Delta^2 y_i =$$

$$= \Delta y_{i+2} - \Delta y_{i+1} - \Delta y_{i+1} + \Delta y_i =$$

$$= y_{i+3} - y_{i+2} - y_{i+2} + y_{i+1} - y_{i+2} + y_{i+1} + y_{i+1} - y_i =$$

$$= y_{i+3} - 3y_{i+2} + 3y_{i+1} - y_i$$

...

Методом математической индукции можно доказать, что

$$\Delta^k = y_{i+k} - k y_{i+k-1} + \frac{k(k-1)}{2!} y_{i+k-2} - \dots + (-1)^k y_i$$

(доказать самостоятельно)

Первая интерполяционная формула Ньютона

Пусть для функции, заданной таблицей с постоянным шагом, составлена таблица конечных разностей. Будем искать интерполяционный многочлен в виде

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

Это многочлен n-й степени

Значения коэффициентов $a_0, a_1, \dots, a_n$ найдем из условия совпадения значений исходной функции многочлена в узлах.

Пусть $x = x_0$

$$y_0 = P_n(x_0) = a_0, \Rightarrow \; a_0 = y_0$$

Пусть $x = x_1$

$$y_1 = P_n(x_1) = a_0 + a_1(x_1 - x_0), \Rightarrow \; a_1 = \frac{y_1 - a_0}{x_1 - x_0} = \frac{y_1 - y_0}{x_1 - x_0} = \frac{\Delta y_0}{h}$$

Первая интерполяционная формула Ньютона

Пусть $x = x_2$

$$y_2 = P_n(x_2) = a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1),$$

$$y_2 - a_0 - a_1(x_2 - x_0) = a_2(x_2 - x_0)(x_2 - x_1)$$

$$a_0 = y_0$$

$$a_1(x_2 - x_0) = a_1 \cdot 2h = 2h \cdot \frac{\Delta y_0}{h} = 2\Delta y_0$$

Первая интерполяционная формула Ньютона

$$a_2(x_2 - x_0)(x_2 - x_1) = a_2 \cdot 2h \cdot h = 2h^2 a_2$$

$$y_2 - y_0 - 2\Delta y_0 = 2h^2 a_2$$

$$2\Delta y_0 = 2(y_1 - y_0)$$

$$y_2 - 2y_1 + 2y_0 - y_0 = y_2 - 2y_1 + y_0$$

$$y_2 - 2y_1 + y_0 = \Delta^2 y_0$$

$$\Delta^2 y_0 = 2h^2 a_2$$

$$a_2 = \frac{\Delta^2 y_0}{2h^2}$$

$$a_2 = \frac{\Delta^2 y_0}{2! \cdot h^2}$$

Первая интерполяционная формула Ньютона

Далее проводя аналогичные выкладки, можно получить

$$a_3 = \frac{\Delta^3 y_0}{3! \cdot h^3}$$

В общем случае получим

$$a_k = \frac{\Delta^k y_0}{k! \cdot h^k}$$

Подставим теперь это выражение в многочлен $P_n(x)$

$$P_n(x) = y_0 + \frac{\Delta y_0}{h}(x - x_0) + \frac{\Delta 2 y_0}{2! h^2}(x - x_0)(x - x_1) + \dots$$

$$\dots + \frac{\Delta^n y_0}{n! h^n}(x - x_0) \dots (x - x_{n-1})$$

Первая интерполяционная формула Ньютона

Практически эта формула применяется в другом виде
Пусть $\frac{x-x_0}{h} = t$, т.е. $x = x_0 + ht$
Тогда

$$\frac{x - x_1}{h} = \frac{x - x_0 - h}{h} = t - 1$$

$$\frac{x - x_2}{h} = \frac{x - x_1 + x_1 - x_2}{h} = \frac{x - x_1 - h}{h} = \frac{x - x_0 - 2h}{h} = t - 2$$

$$P_n(x) = P_n(x_0 + th) = y_0 + t\Delta y_0 \frac{t(t-1)}{2!}\Delta^2 y_0 + \dots + \frac{t(t-1)\dots(t-n+1)}{n!}\Delta^n y_0$$

Эта формула называется первой интерполяционной формулой Ньютона.
Она применяется для интерполирования в начале отрезка интерполяции, когда t мало по абсолютной величине. Поэтому ее называют формулой для интерполирования вперед.

Вторая интерполяционная формула Ньютона

Когда значение аргумента находится ближе к концу отрезка интерполяции, применять первую интерполяционную формулу становится невыгодно. В этом случае применяется вторая интерполяционная формула (формула для интерполирования назад).

$$P_n(x) = a_0 - a_1(x - x_n) - a_2(x - x_n)(x - x_{n-1}) - \dots$$
$$\dots - a_n(x - x_n)\dots(x - x_1) \tag{1'}$$

Вторая интерполяционная формула Ньютона

Как и для первой формулы Ньютона, коэффициенты $a_0, a_1, \dots, a_n$ находятся из условия совпадения значений функции и интерполяционного многочлена в узлах

$$a_k = \frac{\Delta^k y_{n-k}}{k! \cdot h^k} \tag{2'}$$

Подставляя выражение $(2')$ в формулу $(1')$ и переходя к переменной $t = \frac{x-x_n}{h}$ получим окончательный вид второй интерполяционной формулы Ньютона.

$$P_n(x) = P_n(x_n + th) = y_n + t\Delta y_{n-1} + \frac{t(t+1)}{2!}\Delta 2 y_{n-2} + \dots$$

$$\dots + \frac{t(t+1)\dots(t+n-1)}{n!}\Delta^n y_0.$$

Интерполяция сплайнами

Сплайн - это функция, которая на каждом отрезке $[x_{i-1}, x_i]$ $(i = 1, 2, \ldots, n)$ является алгебраическим многочленом, а на всем заданном отрезке $[x_0, x_n]$ непрерывна вместе с несколькими производными. Максимальная степень многочленов по всем отрезкам называется степенью сплайна.

Рассмотрим способ построения сплайнов третьей степени (кубических сплайнов).

Пусть функция $f(x)$ задана таблицей

x	x_0	x_1	$\ldots$	x_n
$f(x)$	y_0	y_1	$\ldots$	y_n

Длину частичного отрезка $[x_{i-1}, x_i]$ обозначим $h_i = x_i - x_{i-1}$ $(i = 1, 2, \ldots, n)$.

Будем искать кубический сплайн на каждом из частичных отрезков $[x_{i-1}, x_i]$ в виде

$$S(x) = a_i + b_i(x - x_{i-1}) + c_i(x - x_{i-1})^2 + d_i(x - x_{i-1})^3, \qquad (1)$$

$$(i = 1, 2, \ldots, n),$$

где a_i, b_i, c_i, d_i - $4n$ неизвестных.

Интерполяция сплайнами

Потребуем совпадения значений $S(x)$ в узлах с табличными значениями функции f :

$$S(x_{i-1}) = y_{i-1} = a_i,$$

$$S(x_i) = y_i = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3.$$

Число этих уравнений (2n) вдвое меньше числа неизвестных коэффициентов.
Чтобы получить дополнительные условия, потребуем непрерывности $S'(x)$ и $S''(x)$ во всех точках, включая узлы. Для этого следует приравнять левые и правые производные

$$S'(x-0), S'(x+0), S''(x-0), S''(x+0)$$

во внутреннем узле x_i. Сначала получим $S'(x)$ и $S''(x)$ во всех точках, включая узлы.

$$S'(x) = b_i + 2c_i(x - x_{i-1}) + 3d_i(x - x_{i-1})^2,$$

$$S''(x) = 2c_i + 6d_i(x - x_{i-1}),$$

Интерполяция сплайнами

$$S'(x_i - 0) = b_i + 2c_i h_i + 3d_i h_i^2,$$

$$S'(x_i + 0) = b_{i+1}, i = 1, 2, ..., n-1$$

При нахождении правой производной сначала заменяем i на $i+1$.

$$S''(x_i - 0) = 2c_i + 6d_i h_i,$$

$$S''(x_i + 0) = 2c_{i+1}.$$

Приравнивая левые и правые производные, получим

$$b_{i+1} = b_i + 2c_i h_i + 3d_i h_i^2, \tag{2}$$

$$2c_{i+1} = 2c_i + 6d_i h_i, (i = 1, 2, ..., n-1). \tag{3}$$

Интерполяция сплайнами

Уравнения (2) и (3) в совокупности дают еще 2(n-1) условий. Недостает еще два условия. Обычно в качестве таких условий берут требования к поведению сплайна в граничных точках x_0 и x_n. Если потребовать нулевой кривизны сплайна на концах (т.е. равенства нулю второй производной), то получим

$$c_1 = 0, c_n + 3d_n h_n = 0.$$

Перепишем все уравнения в виде системы

$$\begin{cases} y_{i-1} = a_i, & i = 1, 2, \ldots, n, \\ b_i h_i + c_i h_i^2 + d_i h_i^3 = y_i - a_i, & i = 1, 2, \ldots, n, \\ b_{i+1} - b_i - 2c_i h_i - 3d_i h_i^2 = 0, & i = 1, 2, \ldots, n-1, \\ c_{i+1} - c_i - 3d_i h_i = 0, & i = 1, 2, \ldots, n-1, \\ c_1 = 0, \\ c_n + 3d_n h_n = 0. \end{cases} \tag{4}$$

Решая эту систему, найдем a_i, b_i, c_i, d_i.

Интерполяция сплайнами

Пример. Построить сплайн третьей степени для функции, заданной таблицей

x	1	2	4	7
$f(x)$	0	3	1	5

Решение.
Здесь $n = 3,\ h_1 = 1,\ h_2 = 2,\ h_3 = 3.$
Для построения сплайна третьей степени

$$S_1(x) = a_1 + b_1(x-1) + c_1(x-1)^2 + d_1(x-1)^3, \quad 1 \leqslant x \leqslant 2;$$

$$S_2(x) = a_2 + b_2(x-2) + c_2(x-2)^2 + d_2(x-2)^3, \quad 2 \leqslant x \leqslant 4;$$

$$S_3(x) = a_3 + b_3(x-4) + c_3(x-4)^2 + d_3(x-4)^3, \quad 4 \leqslant x \leqslant 7$$

необходимо найти a_i, b_i, c_i, d_i $(i = 1, 2, 3)$.

Интерполяция сплайнами

Для этого составим систему вида (4)

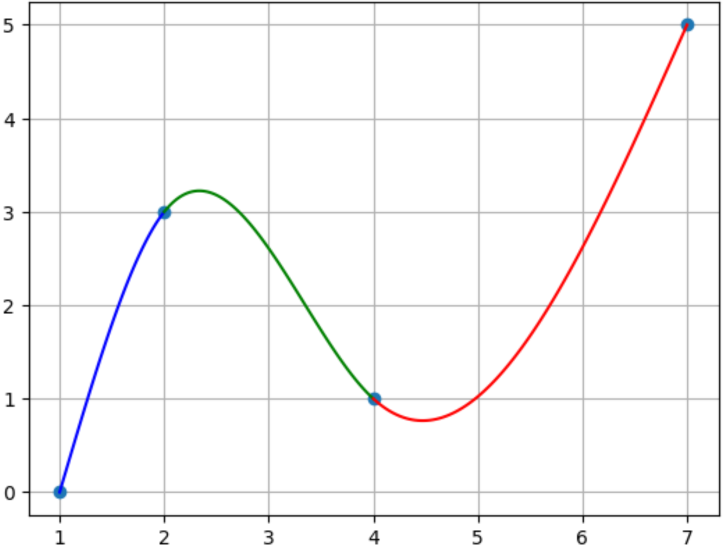
$$
\begin{cases}
a_1 = 0, \\
a_2 = 3, \\
a_3 = 1, \\
b_1 + c_1 + d_1 = 3 - a_1, \\
2b_2 + 4c_2 + 8d_2 = 1 - a_2, \\
3b_3 + 9c_3 + 27d_3 = 5 - a_3, \\
b_2 - b_1 - 2c_1 - 3d_1 = 0, \\
b_3 - b_2 - 4c_2 - 12d_2 = 0, \\
c_2 - c_1 - 3d_1 = 0, \\
c_3 - c_2 - 6d_2 = 0, \\
c_1 = 0, \\
c_3 + 9d_3 = 0.
\end{cases}
$$

Интерполяция сплайнами

Релизовать построение сплайна на Python можно следующим образом

```
import numpy as np
import matplotlib.pyplot as plt
x=np.array([1,2,4,7], dtype=float)
y=np.array([0,3,1,5], dtype=float)
A=np.matrix('\
1,1,1,0,0,0,0,0,0;\
0,0,0,2,4,8,0,0,0;\
0,0,0,0,0,0,3,9,27;\
-1,-2,-3,1,0,0,0,0,0;\
0,0,0,-1,-4,-12,1,0,0;\
0,-1,-3,0,1,0,0,0,0;\
0,0,0,0,-1,-6,0,1,0;\
0,1,0,0,0,0,0,0,0;\
0,0,0,0,0,0,0,1,9\
')
B=np.matrix('3;-2;4;0;0;0;0;0;0')
X=np.linalg.solve(A,B)
def spline1(x,y,X,t):
  s=y[0]+X[0]*(t-x[0])+X[1]*(t-x[0])**2+X[2]*(t-x[0])**3
  return float(s)
def spline2(x,y,X,t):
  s=y[1]+X[3]*(t-x[1])+X[4]*(t-x[1])**2+X[5]*(t-x[1])**3
  return float(s)
def spline3(x,y,X,t):
  s=y[2]+X[6]*(t-x[2])+X[7]*(t-x[2])**2+X[8]*(t-x[2])**3
  return float(s)
x1=np.linspace(1,2,100)
x2=np.linspace(2,4,100)
x3=np.linspace(4,7,100)
y1=[spline1(x,y,X,i) for i in x1]
y2=[spline2(x,y,X,i) for i in x2]
y3=[spline3(x,y,X,i) for i in x3]
plt.plot(x,y,'o',x1,y1,'b-',x2,y2,'g-',x3,y3,'r-')
plt.grid(True)
plt.show()
```

Интерполяция сплайнами



Погрешность многочленной интерполяции

Если известно аналитическое выражение интерполируемой функции $f(x)$, можно применять формулы для оценки погрешности интерполирования (погрешность метода).
Остаточный член интерполяционного многочлена $F_n(x)$ имеет вид:

$$R_n(x) = f(x) - F_n(x)$$

Решение вопроса о погрешности метода одинаково как для многочлена Лагранжа, так и для многочлена Ньютона.
Пусть $f(x)$ имеет все производные до $(n+1)$ - го порядка включительно. Введем вспомогательную функцию

$$u(x) = f(x) - F_n(x) - k \cdot \Pi_{n+1}(x),$$

где k — постоянный множитель.

Погрешность многочленной интерполяции

Как видно, функция $u(x)$ имеет $n+1$ корень (узлы интерполяции $x_0, x_1, ..., x_n$).
Подберем коэффициент k так, чтобы $u(x)$ имела $(n+2)$-й корень в любой точке $x^* \neq x_i$ $(i = 0, 1, \ldots, n)$

$$u(x^*) = 0 \Leftrightarrow f(x^*) - F(x^*) - k \cdot \Pi_{n+1}(x^*) = 0.$$

Для этого достаточно принять

$$k = \frac{f(x^*) - F(x^*)}{\Pi_{n+1}(x^*)} \tag{1}$$

При этом значении k функция $u(x)$ будет иметь $n+2$ корня на отрезке интерполяции и будет обращаться в нуль на концах каждого из $n+1$-го отрезков:

$$[x_0; x_1], [x_1; x_2], [x_i; x^*], [x^*; x_{i+1}], \ldots, [x_{n-1}; x_n].$$

Погрешность многочленной интерполяции

Теорема Ролля

Пусть функция $f(x)$ определена и непрерывна на отрезке $[a, b]$, существует конечная производная $f'(x)$, по крайней мере, на интервале (a, b) и на концах отрезка функция принимает равные значения ($f(a) = f(b)$.) Тогда между a и b найдется такая точка c $(a < c < b)$, что $f'(c) = 0$.

Применяя теорему Ролля к каждому из отрезков, убеждаемся в том, что
$u'(x)$ имеет не менее $n+1$ корней,
$u''(x)$ имеет не менее n корней,
$\vdots$
$u^{n+1}(x)$ имеет не менее 1 корня.

Погрешность многочленной интерполяции

Пусть α та точка, в которой $u^{(n+1)}(\alpha) = 0$.
Продифференцируем $u(x)$ $n+1$ раз.

$$u^{(n+1)}(x) = f^{(n+1)}(x) - k \cdot (n+1)!$$

Откуда

$$k = \frac{f^{(n+1)}(x) - u^{(n+1)}(x)}{(n+1)!}$$

При $x = \alpha$

$$k = \frac{f^{(n+1)}}{(n+1)!} \tag{2}$$

Сравнивая (1) и (2) имеем

$$f(x^*) - F_n(x^*) = \frac{f^{(n+1)}(\alpha)}{(n+1)!}\Pi_{n+1}(x)$$

Пусть $M_{n+1} = \max\limits_{x_0 \le x \le x_n} |f^{(n+1)}(x)|$ Тогда

$$|R_n(x)| \le \frac{M_{n+1}}{(n+1)!}|\Pi_{n+1}(x)| \tag{3}$$

Погрешность многочленной интерполяции

Пусть

$$M_{n+1} = \max_{x_0 \le x \le x_n} |f^{(n+1)}(x)|$$

Тогда

$$|R_n(x)| \le \frac{M_{n+1}}{(n+1)!} |\Pi_{n+1}(x)| \tag{3}$$

Последнее неравенство может быть использовано для оценки погрешности метода интерполирования по формуле Лагранжа. Используя подстановки $t = \frac{x-x_0}{h}$, $t = \frac{x-x_n}{h}$ и заменяя соответственно выражение для $\Pi_{n+1}(x)$ можно получить из (3) формулы оценки погрешностей интерполирования по формулам Ньютона.

$$|R_n(x)| \le \frac{h^{n+1} M_{n+1}}{(n+1)!} |t(t-1)(t-2)\dots(t-n)|$$

$$|R_n(x)| \le \frac{h^{n+1} M_{n+1}}{(n+1)!} |t(t+1)(t+2)\dots(t+n)|$$

Погрешность многочленной интерполяции

Используя подстановки $t = \frac{x-x_0}{h}$, $t = \frac{x-x_n}{h}$ и заменяя соответственно выражение для $\Pi_{n+1}(x)$ можно получить из (3) формулы оценки погрешностей интерполирования по формулам Ньютона.

$$|R_n(x)| \le \frac{h^{n+1}M_{n+1}}{(n+1)!}|t(t-1)(t-2)\dots(t-n)|$$

$$|R_n(x)| \le \frac{h^{n+1}M_{n+1}}{(n+1)!}|t(t+1)(t+2)\dots(t+n)|$$

При малых значениях h и при условии непрерывности $f^{(n+1)}(x)$ справедливо приближенное равенство

$$M_{n+1} \approx \frac{\Delta^{n+1}y}{h^{n+1}}$$

где

$$\Delta^{n+1}y = \max_{0\le m\le n}|\Delta^{n+1}y_m|.$$

При этом условии оценки принимают вид

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

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

Погрешность многочленной интерполяции

При малых значениях h и при условии непрерывности $f^{(n+1)}(x)$ справедливо приближенное равенство

$$M_{n+1} \approx \frac{\Delta^{n+1} y}{h^{n+1}}$$

где

$$\Delta^{n+1} y = \max_{0 \le m \le n} |\Delta^{n+1} y_m|.$$

При этом условии оценки принимают вид

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

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

Метод наименьших квадратов

Пусть функция $f(x)$ задана таблицей

x	x_0	x_1	$\ldots$	x_n
$f(x)$	y_0	y_1	$\ldots$	y_n

При нахождении аналитического выражения приближающей функции методом наименьших квадратов не требуется совпадения значений исходной и приближающей функций в точках x_i. Однако необходимо учитывать класс функций, к которому относится искомая функция. На практике это можно сделать, отметив на графике исходные точки и учитывая дополнительные условия задачи. Часто используют следующие функции:

1). $y = ax + b$ (линейная функция);
2). $y = ax^2 + bx + c$ (квадратичная функция);
3). $y = ax^m$ (степенная функция);
4). $y = ae^{mx}$ (показательная функция);
5). $y = \frac{ax+b}{cx+d}$ (дробно-линейная функция);
6). $y = a \cdot lnx + b$ (логарифмическая функция);
7). $y = \frac{a}{x} + b$ (гипербола).

Метод наименьших квадратов

Для нахождения параметров конкретной приближающей формулы в методе наименьших квадратов требуется, чтобы сумма квадратов отклонений значений приближающей функции от значений исходной функции была минимальной.
Будем рассматривать нахождение параметров $a_1, a_2, \ldots, a_n$ приближающей функции $F(x, a_1, a_2, \ldots, a_n)$.
Пусть

$$\sum_{i=0}^{n} \left(y_i - F(x_i, a_1, a_2, \ldots, a_n)\right)^2 = \Phi\left(a_1, a_2, \ldots, a_n\right).$$

Метод наименьших квадратов

При применении метода наименьших квадратов необходимо найти точки минимума функции $\Phi(a_1, a_2, \ldots, a_n)$, т.е. точки, в которых

$$\frac{\partial \Phi}{\partial a_1} = 0, \frac{\partial \Phi}{\partial a_2} = 0, \ldots, \frac{\partial \Phi}{\partial a_n} = 0.$$

Таким образом, получим систему n линейных алгебраических уравнений с n неизвестными

$$\begin{cases} \sum\limits_{i=0}^{n} (y_i - F(x_i, a_1, a_2, \ldots, a_n)) \cdot \frac{\partial F(x_i, a_1, a_2, \ldots, a_n)}{\partial a_1} = 0, \\ \sum\limits_{i=0}^{n} (y_i - F(x_i, a_1, a_2, \ldots, a_n)) \cdot \frac{\partial F(x_i, a_1, a_2, \ldots, a_n)}{\partial a_2} = 0, \\ \ldots \\ \sum\limits_{i=0}^{n} (y_i - F(x_i, a_1, a_2, \ldots, a_n)) \cdot \frac{\partial F(x_i, a_1, a_2, \ldots, a_n)}{\partial a_n} = 0. \end{cases} \tag{5}$$

Решая эту систему, найдем значения параметров $a_1, a_2, \ldots, a_n$.

Метод наименьших квадратов

Пример. Используя метод наименьших квадратов, найти приближающую функцию в виде линейной функции (линейная регрессия)

$$F\left(x,a,b\right)=ax+b.$$

Составим систему вида (5)

$$\begin{cases} \sum\limits_{i=0}^{n}\left(y_i-ax_i-b\right)\cdot x_i=0, \\ \sum\limits_{i=0}^{n}\left(y_i-ax_i-b\right)=0. \end{cases}$$

Метод наименьших квадратов

Преобразуем эту систему к равносильной

$$\begin{cases} \left(\frac{1}{n}\sum\limits_{i=0}^{n}(x_i)^2\right)\cdot a + \left(\frac{1}{n}\sum\limits_{i=0}^{n}x_i\right)\cdot b = \frac{1}{n}\sum\limits_{i=0}^{n}y_i x_i, \\ \left(\frac{1}{n}\sum\limits_{i=0}^{n}x_i\right)\cdot a + b = \frac{1}{n}\sum\limits_{i=0}^{n}y_i, \quad i = 1,\dots,n. \end{cases}$$

Решая эту систему, найдем значения параметров a и b.
Необходимым условием для выбора линейной функции в качестве искомой эмпирической является соотношение

$$y(\frac{x_1+x_n}{2} - \sqrt{y(x_1)y(x_n)} = 0.$$

Метод наименьших квадратов

Квадратичная функция (квадратичная регрессия).
Будем искать приближающую функцию в виде квадратного трехчлена.

$$F(x, a, b, c) = ax^2 + bx + c.$$

Находим частные производные:

$$F'_a = x^2, F'_b = x, F'_c = 1.$$

Составим систему вида

$$\begin{cases} \sum\limits_i (y_i - a(x_i)^2 - bx_i - c) \cdot (x_i)^2 = 0, \\ \sum\limits_i (y_i - a(x_i)^2 - bx_i - c) \cdot x_i = 0, \\ \sum\limits_i (y_i - a(x_i)^2 - bx_i - c) = 0, i = 1, \ldots, n. \end{cases}$$

После несложных преобразований получается система трех линейных уравнений с тремя неизвестными a, b, c.
Квадратичная регрессия применяется, если все выражения вида $y_2 - 2y_1 + y_0, y_3 - 2y_2 + y_1, y_4 - 2y_3 + y_2$ и т.д. мало отличаются друг от друга.

Метод наименьших квадратов

Степенная функция (геометрическая регрессия).
Будем искать приближающую функцию в виде

$$F(x,a,m) = ax^m.(1)$$

Предполагая, что в исходной таблице значения аргумента и значения функции положительны, прологарифмировав последнее равенство при условии $a > 0$, получим

$$lnF = lna + m \cdot lnx(2)$$

Так как функция F является приближающей для функции f, функция lnF будет приближающей для функции lnf. Введем новую переменную $u = lnx$. Тогда, lnF будет функцией от $u : \Phi(u)$.
Обозначим

$$m = A, lna = B.(3)$$

Тогда

$$\Phi(u,A,B) = Au + B.(4)$$

Таким образом, задача сводится к отысканию приближающей функции в виде линейной.

Метод наименьших квадратов

Степенная функция (геометрическая регрессия).

На практике, для нахождения искомой приближающей функции в виде степенной, необходимо проделать следующее:

1. По данной таблице составить новую таблицу, прологарифмировав значения x и y в исходной таблице;
2. По новой таблице найти параметры A и B приближающей функции вида (4);
3. Использовав обозначения (3), найти значения параметров a и m и подставить их в выражение (1).

Необходимым условием для выбора степенной функции в качестве искомой эмпирической формулы является соотношение:

$$y(\sqrt{x_1 x_n}) - \sqrt{y(x_1) y(x_n)} = 0.$$

Метод наименьших квадратов

Показательная функция

Пусть исходная таблица такова, что приближающую функцию целесообразно искать в виде показательной функции

$$F(x, a, m) = a \cdot e^{mx}, a > 0.(5)$$

Прологарифмировав равенство (5), получим

$$lnF = lna + mx(6)$$

Приняв обозначения (3), перепишем (6) в виде

$$lnF = Ax + B.(7)$$

Метод наименьших квадратов

Показательная функция

Таким образом, для нахождения приближающей функции в виде (5) нужно прологарифмировать значения функции в исходной таблице (1) и, рассматривая их совместно с исходными значениями аргумента, построить для новой таблицы приближающую функцию вида (7). Вслед за этим в соотвествии с обозначениями (3) остается получить значения искомых параметров a и b и подставить их в формулу (5).

Необходимым условием для выбора показательной функции в качестве искомой эмпирической формулы является соотношение

$$y(\frac{x_1 + x_n}{2}) - \sqrt{y(x_1)y(x_n)} = 0.$$

Метод наименьших квадратов

Дробно-линейная функция
Будем искать приближающую функцию в виде

$$F(x, a, b) = \frac{1}{ax + b}.(8)$$

Равенство (8) перепишем следующим образом:

$$\frac{1}{F(x, a, b)} = ax + b.$$

Метод наименьших квадратов

Дробно-линейная функция

Из последнего равенства следует, что для нахождения значений параметров a и b по заданной таблице нужно составить новую таблицу, у которой значения аргумента оставить прежними, а значения функции заменить обратными числами, после чего для полученной таблицы найти приближающую функцию вида $ax + b$. Найденные значения параметров a и b подставить в формулу (8).

Необходимым условием для выбора дробно-линейной функции в качестве искомой эмпирической формулы является соотношение

$$y\left(\frac{x_1 + x_n}{2}\right) - \frac{2y(x_1)y(x_n)}{y(x_1) + y(x_n)} = 0.$$

Метод наименьших квадратов

Логарифмическая функция

Пусть приближающая функция имеет вид:

$$F(x, a, b) = a \cdot lnx + b.(9)$$

Легко видеть, что для перехода к линейной функции достаточно сделать подстановку $lnx = u$. Отсюда следует, что для нахождения зна чений a и b ужно прологарифмировать значения аргумента в исходной таблице и, рассматривая полученные значения в совокупности с исходными значениями функции, найти для полученной таким образом новой таблицы приближающую функцию в виде линейной. Коэффициенты a и b найденной функции затем нужно подставить в формулу (9).

Необходимым условием для выбора логарифмической функции в качестве искомой эмпирической формулы является соотношение.

$$y(\sqrt{x_1 x_n}) - \frac{y(x_1) + y(x_n)}{2} = 0.$$

Метод наименьших квадратов

Гипербола

Если точечный график, построенный по таблице, дает ветвь гиперболы, приближающуя функцию можно искать в виде:

$$F(x, a, b) = \frac{a}{x} + b.(10)$$

Для перехода к линейной функции сделаем подстановку $u = \frac{1}{x}$. Тогда

$$\Phi(u, a, b) = au + b(11)$$

Практически перед нахождением приближающей функции вида (10) значения аргумента в исходной таблице следует заменить обратными числами и найти для новой таблицы приближающую функцию в виде линейной вида (11). Полученные значения параметров a и b нужно подставить в формулу (1).

Необходимым условием для выбора уравнения гиперболы функции в качестве искомой эмпирической формулы является соотношение.

$$y(\frac{2x_1x_n}{x_1 + x_n}) - \frac{y(x_1) + y(x_n)}{2} = 0.$$

Метод наименьших квадратов

Дробно-рациональная функция

Пусть приближающая функция находится в виде

$$F(x,a,b) = \frac{x}{ax+b}.(12)$$

Тогда

$$\frac{1}{F(x,a,b)} = a + \frac{b}{x}.$$

Таким образом, задача сводится к случаю, рассмотренному в предыдущем пункте. Действительно, если в исходной таблице заменить значения x и y их обратными величинами по формулам $z = \frac{1}{x}$ и $u = \frac{1}{y}$ и искать для новой таблицы приближающую функцию вида $u = bz + a$, то найденные значения a и b будут искомыми для (12). Необходимым условием для выбора дробно-рациональной функции в качестве искомой эмпирической формулы является соотношение

$$y\left(\frac{2x_1x_n}{x_1+x_n}\right) - \frac{2y(x_1)y(x_n)}{y(x_1)+y(x_n)} = 0.$$