

08. Сплайн-интерполяция. Подпрограммы  
**SPLINE** и **SEVAL**. Интерполирование по  
Эрмиту. Обратная задача интерполирования.

Андрей Бареков      Ярослав Пылаев  
По лекциям Устинова С.М.

February 28, 2020

На практике интерполяционные полиномы высоких степеней строят редко, т.к. они очень чувствительны к погрешности в исходных данных. В таких случаях возможно *разбиение исходного промежутка на ряд участков*, на каждом из которых строится полином относительно невысокой степени.

На практике так поступают часто, однако в ряде приложений требуется дифференцируемость функции, а она отсутствует в точках сопряжения соседних полиномов. Решить проблему позволяет *сплайн-интерполяция*.

## 1 Сплайн-интерполяция

Имеется  $N$  точек и  $N - 1$  промежутков:

$$[x_1, x_2], [x_2, x_3], \dots, [x_k, x_{k+1}], \dots, [x_{N-1}, x_N]$$

На каждом промежутке  $[x_k, x_{k+1}]$  строится интерполяционный полином третьей степени:

$$[x_k, x_{k+1}] \rightarrow S_k(x) = \overbrace{a_k}^{=f_k} + b_k(x - x_k) + c_k(x - x_k)^2 + d_k(x - x_k)^3$$

Возникшая степень свободы используется для гладкого сопряжения соседних полиномов

Итак, имеем  $4 \times (N - 1) = 4N - 4$  параметра.

Потребуем, чтобы во всех *внутренних точках* совпадали соседние полиномы, их первые и вторые производные - это и будет условием гладкого сопряжения.

$$\begin{cases} S_k(x_{k+1}) = S_{k+1}(x_{k+1}) \\ S'_k(x_{k+1}) = S'_{k+1}(x_{k+1}) \\ S''_k(x_{k+1}) = S''_{k+1}(x_{k+1}) \end{cases}, \quad k = 1, 2, \dots, N - 2$$

Выходит  $3 \times (N - 2) = 3N - 6$  условий,  $+N$  условий интерполирования (совпадение полиномов и их функции в узлах)  $= 4N - 6$  условий.

$N$  уравнений отражают  
требование интер-ния:  $S_k(x_k) = f_k$   
 $k = 1, 2, \dots, N - 1$  (по  $N$  точкам  $N - 1$  ст.)  
Обратим внимание:  $S_{N-1}(x_N) = f_N$

Недостающие два уравнения чаще всего задаются на концах промежутка в точках  $x_1$  и  $x_N$ . Эти уравнения должны удовлетворять двум условиям одновременно:

1. Возникшая система уравнений должна как можно проще решаться.
2. Эти уравнения должны быть согласованы с характером поведения функции на концах промежутка.

В предлагаемом ПО строится интерполяционный полином третьей степени  $Q_3(x)$  по первым четырём точкам и приравняется:

$$S_1'''(x_1) = Q_3'''(x_1) \Rightarrow 6d_1 = \text{const} \text{ (дифференцировали)}$$

$$\begin{aligned} d_1(x-x_k)^3, \\ 3d_1(x-x_k)^2, \\ 6d_1(x-x_k), \\ 6d_1 = \text{const} \end{aligned}$$

Аналогично, по последним четырём точкам строится полином  $\tilde{Q}_3(x)$  и приравняется:

$$S_{N-1}'''(x_N) = \tilde{Q}_3'''(x_N)$$

## 2 Программное обеспечение

ПО состоит из двух программ:

### 2.1 SPLINE

$$SPLINE(\underbrace{N}_{\text{Кол-во точек}}, \underbrace{X_K}_{\text{Вектор с узлами}}, \underbrace{F_K}_{\text{Вектор с знач.}}, \underbrace{B, C, D}_{\text{Выходные параметры } b_k, c_k, d_k})$$

Решает систему уравнений относительно коэффициентов полиномов.

### 2.2 SEVAL

$$\underbrace{SEVAL}_{\text{function}}(N, X, X_K, F_K, \underbrace{B, C, D}_{\text{Получены из SPLINE}})$$

Вычисляет значение сплайна в некоторой точке  $X, X \in [x_k, x_{k+1}]$ .

$S_k(x_k) = a_k = f_k$  (поэтому ищем только  $b, c, d$ ).

Программа изначально определяет промежуток, на котором находится  $X$ , а потом высчитывает полином. Номер полинома находится двоичным поиском.

## 3 Интерполирование по Эрмиту

Так называется задача интерполирования, когда в таблице кроме значений функции присутствуют еще и производные.

$x$	$f(x)$	$f'(x)$	$f''(x)$
$x_0$	$f_0$	$f'_0$	
$x_1$	$f_1$	$f'_1$	$f''_1$
$x_2$	$f_2$		

$$\begin{cases} H(x_k) = f_k, k = 0, 1, 2 \\ H'(x_k) = f'_k, k = 0, 1 \\ H''(x_k) = f''_k, k = 1 \end{cases}$$

Степень полинома Эрмита равна общему количеству условий  $-1$ . Аналогично полиному Лагранжа можно записать полином Эрмита в готовом виде, не решая систему уравнений. Наиболее простой вид он имеет, когда количество условий во всех узлах одинаково.

### 3.1 Пример

Рассмотрим разложение функции в степенной ряд Тейлора в точке  $x_0$ :

$$f(x) = \underbrace{f(x_0) + \frac{x - x_0}{1!} f'(x_0) + \frac{(x - x_0)^2}{2!} f''(x_0) + \dots}_{P_2(x)}$$

Первые три слагаемых - частная сумма ряда Тейлора в  $x_0$ .

$$\begin{cases} P_2(x_0) = f(x_0) \\ P_2'(x_0) = f'(x_0) \\ P_2''(x_0) = f''(x_0) \end{cases}$$

Так как полином  $P(x)$  удовлетворяет условиям выше, то частная сумма ряда Тейлора - частный случай полинома Эрмита с одним узлом интерполирования.

## 4 Обратная задача интерполирования

$x$	$f(x)$	
$x_0$	$f(x_0)$	$x^* \Rightarrow f(x^*) = ?$ прямая задача интерполирования
$x_1$	$f(x_1)$	
$\dots$	$\dots$	<u><math>f(x^*) = f^* \Rightarrow x^* = ?</math> восстановить аргумент</u>
$x_m$	$f(x_m)$	

I. *Строим интерполяционный полином*, приравниваем значение функции, ищем корни:  $Q_m(x) = f^*$ . Возникшее уравнение решается либо аналитически, либо приближённо численными методами.

II. *Меняем местами столбцы таблицы*, строим интерполяционный полином для обратной функции, и вычисляем значение этого полинома в точке  $f^*$  и находим  $x^*$ .

**Ограничение:** Для существования обратной функции, исходная должна быть строго монотонной. В противном случае делим на промежутки монотонности.