

Оглавление

| | | |
|-----|---|---|
| 1 | Интерполяция функции 1 переменной | 2 |
| 1.1 | Линейная интерполяция | 2 |
| 1.2 | Полиномиальная интерполяция | 4 |
| 1.3 | Кубический сплайн | 5 |
| 1.4 | В-сплайн | 7 |

1 Интерполяция функции 1 переменной

Рассмотрим набор попарно различных точек $\{x_i\}_{i=0}^n, x_i \in [a, b]$. Пусть $\{y_i\}_{i=0}^n$ - значения некоторой функции $f: [a, b] \rightarrow \mathbb{R}$: в этих точках: $y_i = f(x_i)$. Предполагается, что сама функция f не известна, а известны только её значения в точках x_i . Задача интерполяции функции 1 переменной - построить функцию $\varphi: [a, b] \rightarrow \mathbb{R}$, такую что выполняются следующие условия: $\varphi(x_i) = y_i$. Т.е. построенная функция φ должна совпадать с неизвестной функцией f в заданном наборе узлов. Далее будут рассмотрены 3 способа построения функции φ : линейный, полиномиальный и кубическая интерполяция. Также будут рассмотрены В-сплайны, которые не интерполируют функцию, а приближают её.

1.1 Линейная интерполяция

Линейная интерполяция - наиболее простой способ интерполяция, при котором φ является кусочно-линейной функцией. При таком способе интерполяции соседние узлы соединены прямой линией. Интерполирующая функция φ имеет следующий вид:

$$\varphi(x) = y_i + (y_{i+1} - y_i) \frac{x - x_i}{x_{i+1} - x_i}, x \in [x_i, x_{i+1}] \quad (1)$$

Преимущества:

- простота реализации;
- высокая скорость построения φ ;
- высокая скорость вычисления $\varphi(x)$;

Недостатки:

- φ не является непрерывно-дифференцируемой;

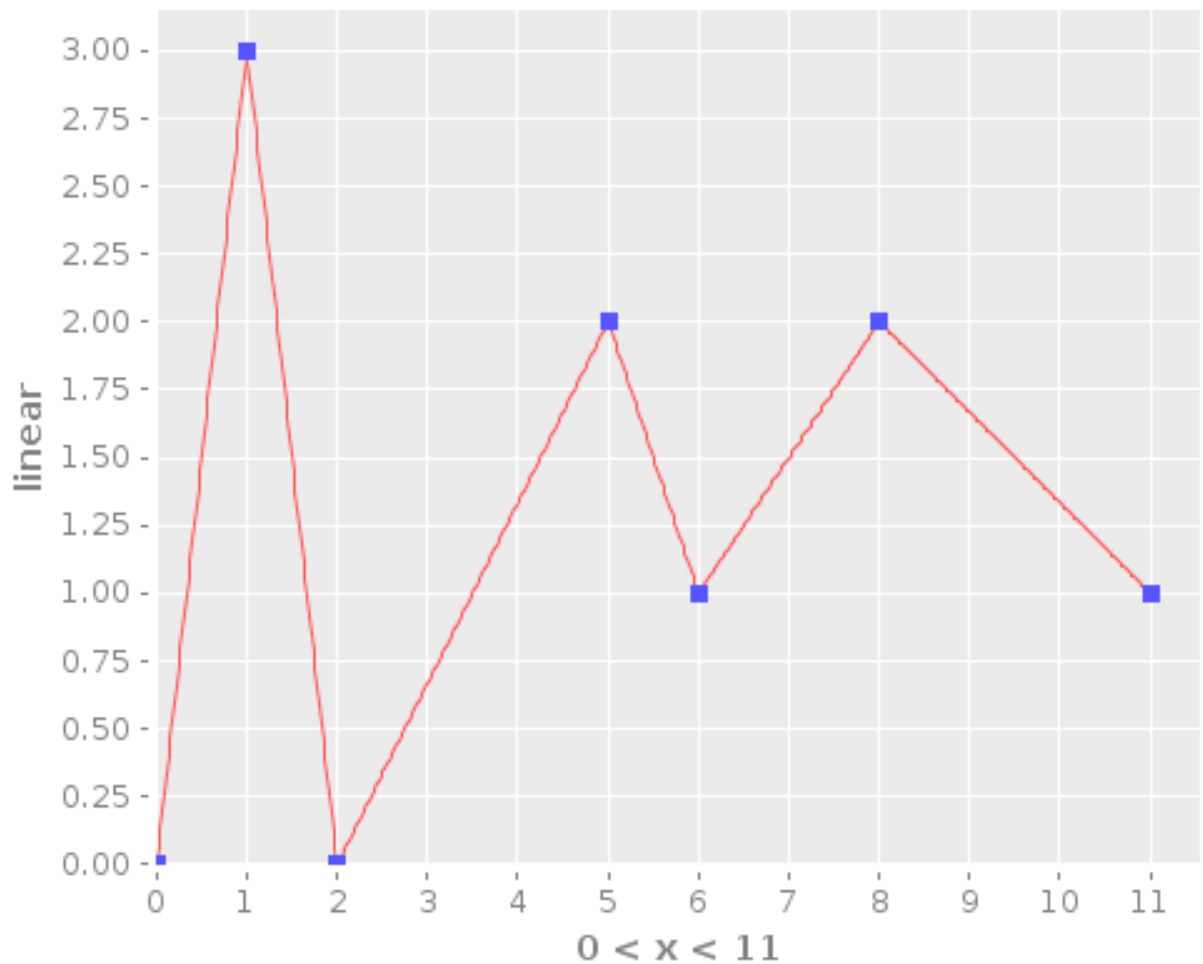
Временная сложность метода:

Построение φ : $O(n \log n)$ - требуется отсортировать все узлы.

Вычисление $\varphi(x)$: $O(\log n)$ - поиск соответствующего узла.

Пример:

```
; define points  
(def points [[0 0] [1 3] [2 0] [5 2] [6 1] [8 2] [11 1]])  
  
; build interpolation function  
(def lin (interpolate points :linear))  
  
; view plot on [0, 11]  
(view (function-plot lin 0 11))
```



1.2 Полиномиальная интерполяция

В данном способе интерполяции φ является многочленом степени n . Для построения φ используется формула Ньютона с разделёнными разностями. Вычисления производятся по следующим формулам:

$$\begin{aligned}\varphi(x) &= \sum_{i=0}^n f[x_0, \dots, x_i] \omega_i(x) \\ \omega_i(x) &= (x - x_0)(x - x_1) \dots (x - x_i) \\ f[x_0, \dots, x_i] &= \frac{f[x_1, \dots, x_i] - f[x_0, \dots, x_{i-1}]}{x_i - x_0}, \forall i \geq 1 \\ f[x_j] &= f(x_j), \forall j\end{aligned}\tag{2}$$

Преимущества:

- φ имеет производную любого порядка;

Недостатки:

- при больших n интерполяционный многочлен будет иметь большую погрешность интерполирования;
- низкая скорость построения φ ;
- низкая скорость вычисления $\varphi(x)$;

Временная сложность метода:

Построение φ : $O(n^2)$

Вычисление $\varphi(x)$: $O(n)$

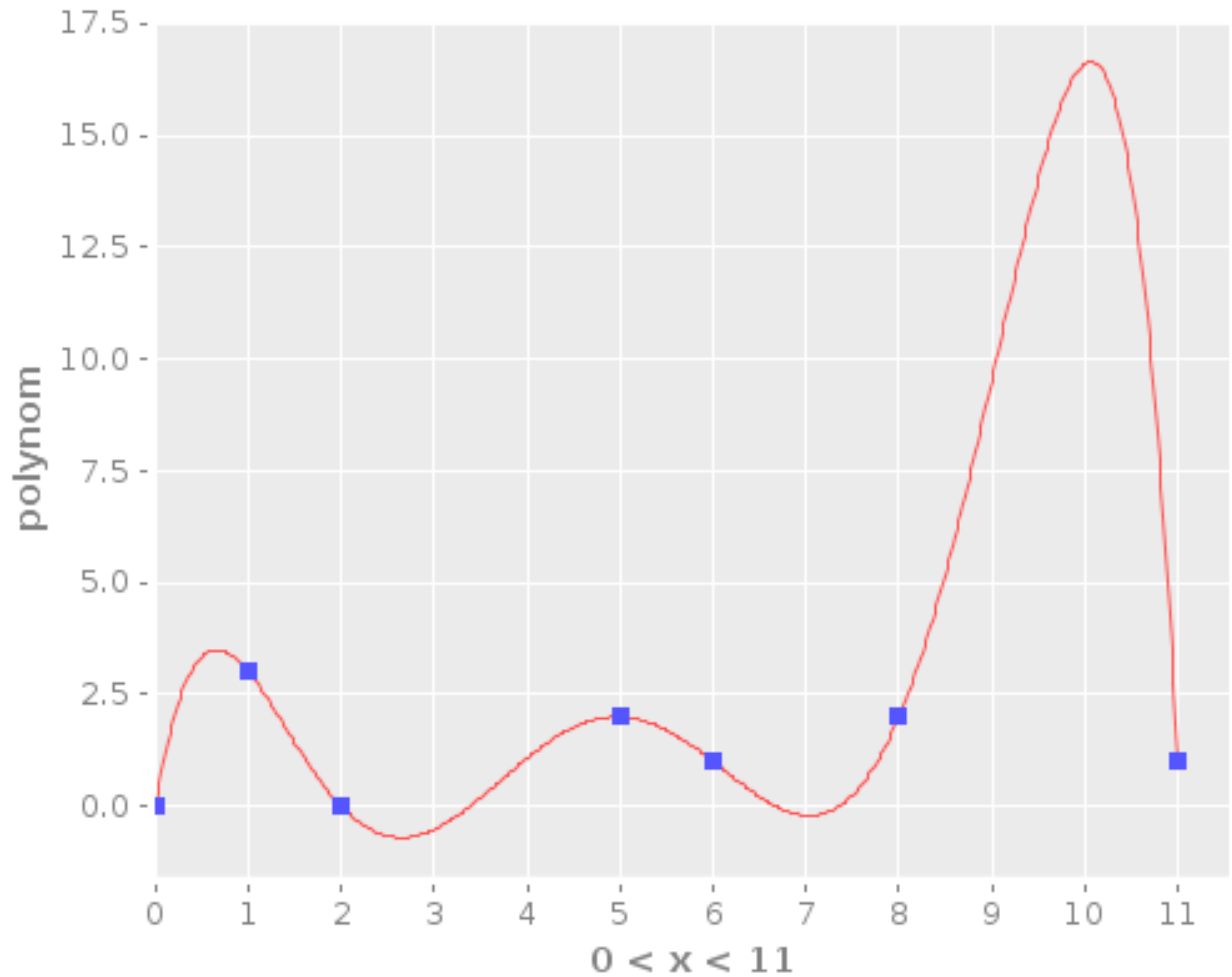
Пример:

```
; define points
(def points [[0 0] [1 3] [2 0] [5 2] [6 1] [8 2] [11 1]])

; build interpolation function
(def polynom (interpolate points :polynomial))
```

```
; view plot on [0, 11]
```

```
(view (function-plot polynom 0 11))
```



1.3 Кубический сплайн

Функция φ является кусочной и на каждом отрезке $[x_{i-1}, x_i], i = \overline{1, n}$ задаётся отдельным кубическим многочленом:

$$\varphi(x) = s_i(x) = \alpha_i + \beta_i(x - x_i) + \frac{\delta_i}{2}(x - x_i)^2 + \frac{\delta_i}{6}(x - x_i)^3, x \in [x_{i-1}, x_i] \quad (3)$$

Также накладываются требования наличия непрерывной первой и второй производной φ , из чего получаем дополнительные условия:

$$\begin{aligned} s'_{i-1}(x_{i-1}) &= s'_i(x_{i-1}), i = \overline{2, n} \\ s''_{i-1}(x_{i-1}) &= s''_i(x_{i-1}), i = \overline{2, n} \end{aligned} \quad (4)$$

Используя эти условия и условия интерполяции получаем следующие формулы для вычисления коэффициентов $\{\alpha_i\}$, $\{\beta_i\}$, $\{\delta_i\}$:

$$\begin{aligned} h_i &= x_i - x_{i-1}, i = \overline{1, n} \\ \alpha_i &= y_i, i = \overline{0, n} \\ \beta_i &= \frac{y_i - y_{i-1}}{h_i} + \frac{2\gamma_i + \gamma_{i-1}}{6}, i = \overline{1, n} \\ \delta_i &= \frac{\gamma_i - \gamma_{i-1}}{h_i}, i = \overline{2, n} \end{aligned} \quad (5)$$

Коэффициенты $\{\delta_i\}$ можно получить, решив следующую 3-диагональную систему линейных уравнений:

$$h_i \gamma_{i-1} + 2(h_i + h_{i+1}) \gamma_i + h_{i+1} \gamma_{i+1} = 6 \left(\frac{y_{i+1} - y_i}{h_{i-1}} - \frac{y_i - y_{i-1}}{h_i} \right), i = \overline{i, n-1} \quad (6)$$

Данная система имеет $n-2$ уравнений и n неизвестных. Задавая различные граничные условия можно получить недостающие уравнения и решить систему, тем самым получая коэффициенты для кубического сплайна. Были реализованы 2 вида граничных условий:

1. Естественные граничные условия. Полагают $\varphi''(x_0) = \varphi''(x_n) = 0$.
2. Периодические (замкнутые) граничные условия. Полагают $\varphi'(x_0) = \varphi'(x_n)$, $\varphi''(x_0) = \varphi''(x_n)$.

Временная сложность метода:

Построение φ : $O(n)$

Вычисление $\varphi(x)$: $O(\log n)$ - поиск соответствующего промежутка.

Пример:

```
; define points
(def points [[0 0] [1 3] [2 0] [5 2] [6 1] [8 2] [11 1]])

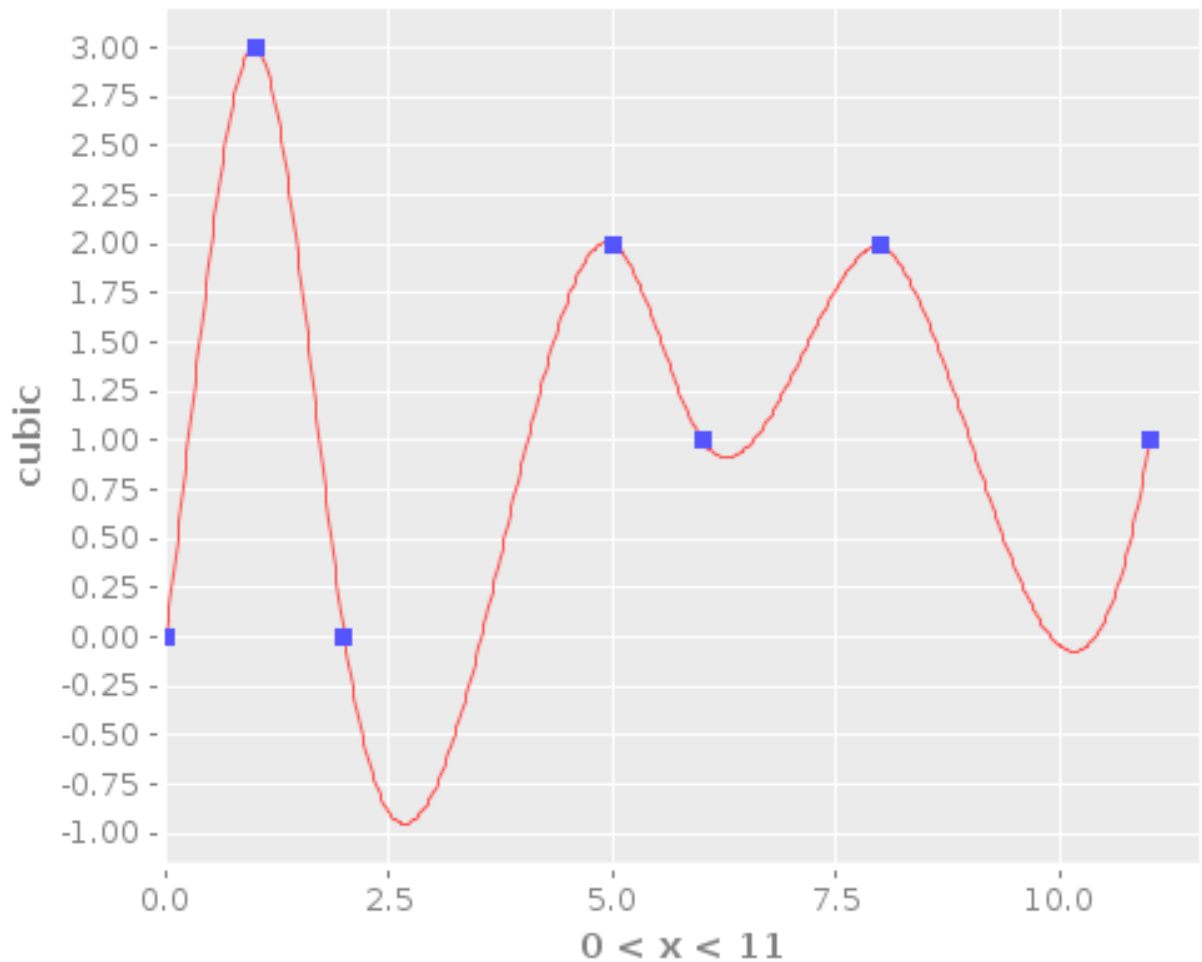
; build interpolation function
```

```

(def cubic (interpolate points :cubic-spline :boundaries :closed))

; view plot on [0, 11]
(view (function-plot cubic 0 11))

```



1.4 В-сплайн

В-сплайны не интерполируют узлы, как ранее рассмотренные методы, а приближают их. При построении В-сплайновой кривой узлы $\{x_i\}$ обычно не задаются, они вычисляются в процессе построения кривой. Задаются только значения $\{y_i\}$.

Временная сложность метода:

Построение φ : $O(n)$

Вычисление $\varphi(t)$: $O(d^2)$, где d - степень сплайна

```

; define points
(def points [[0 0] [1 3] [2 0] [5 2] [6 1] [8 2] [11 1]])

; build approximation function
(def b-spline (approximate points :degree 3))

; view plot on [0, 1]
(view (parametric-plot cubic 0 1))

```

