

Interpolación con splines cuadráticos

Métodos Numéricos

Rubén
Morales

Pablo
Baeyens

Francisco
Morales

Pablo
Medina

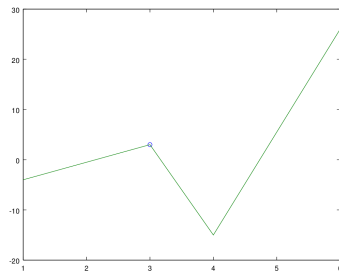
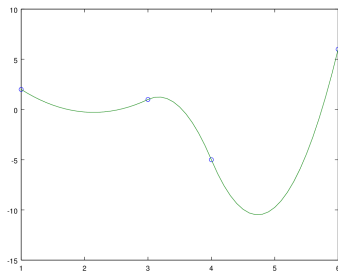
Miguel
Anguita

10 de Junio de 2015

Splines cuadráticos

Definición

Un **spline cuadrático** es una función $s \in S_2^1(P)$, para cierta partición P sobre un intervalo.



Dimensión del espacio

Proposición

Sea $[a, b]$ intervalo, $P = \{x_i\}_{i=0 \dots n} \in \mathcal{P}([a, b])$, entonces $\dim(S_2(P)) = n + 2$.

Demostración.

Podemos deducirlo a partir de la fórmula general:

$$\dim(S_k^r(P)) = (k - r)n + r + 1$$

Así:

$$\dim(S_2^1(P)) = (2 - 1)n + 1 + 1 = 2$$



Dimensión del espacio

Proposición

Sea $[a, b]$ intervalo, $P = \{x_i\}_{i=0\dots n} \in \mathcal{P}([a, b])$, entonces $\dim(S_2(P)) = n + 2$.

Demostración.

Podemos deducirlo a partir de la fórmula general:

$$\dim(S_k^r(P)) = (k - r)n + r + 1$$

Así:

$$\dim(S_2^1(P)) = (2 - 1)n + 1 + 1 = 2$$



Dimensión del espacio

Proposición

Sea $[a, b]$ intervalo, $P = \{x_i\}_{i=0\dots n} \in \mathcal{P}([a, b])$, entonces $\dim(S_2(P)) = n + 2$.

Demostración.

Podemos deducirlo a partir de la fórmula general:

$$\dim(S_k^r(P)) = (k - r)n + r + 1$$

Así:

$$\dim(S_2^1(P)) = (2 - 1)n + 1 + 1 = 2$$



Dimensión del espacio

Proposición

Sea $[a, b]$ intervalo, $P = \{x_i\}_{i=0\dots n} \in \mathcal{P}([a, b])$, entonces $\dim(S_2(P)) = n + 2$.

Demostración.

Podemos deducirlo a partir de la fórmula general:

$$\dim(S_k^r(P)) = (k - r)n + r + 1$$

Así:

$$\dim(S_2^1(P)) = (2 - 1)n + 1 + 1 = 2$$



Dimensión del espacio

Proposición

Sea $[a, b]$ intervalo, $P = \{x_i\}_{i=0\dots n} \in \mathcal{P}([a, b])$, entonces $\dim(S_2(P)) = n + 2$.

Demostración.

Podemos deducirlo a partir de la fórmula general:

$$\dim(S_k^r(P)) = (k - r)n + r + 1$$

Así:

$$\dim(S_2^1(P)) = (2 - 1)n + 1 + 1 = 2$$



Base del espacio

Base

Conocida la dimensión podemos establecer una base:

$$\{1, x, x^2, (x - x_1)_+^2, \dots, (x - x_{n-1})_+^2\}$$

Método local

Dada la derivada en el nodo k , podemos calcular mediante diferencias divididas los trozos a ambos lados:

Si d_k queda a la **derecha** del trozo:

$$\begin{array}{ccc}
 x_{k-1} & y_{k-1} & \\
 x_k & y_k & p_k \\
 x_k & y_k & d_k \quad \frac{d_k - p_k}{h_k}
 \end{array}$$

$$s_k(x) = y_{k-1} + p_k(x - x_{k-1}) + \frac{d_k - p_k}{h_k}(x - x_{k-1})(x - x_k) \quad (1)$$

Método local

Dada la derivada en el nodo k , podemos calcular mediante diferencias divididas los trozos a ambos lados:

Si d_k queda a la **izquierda** del trozo:

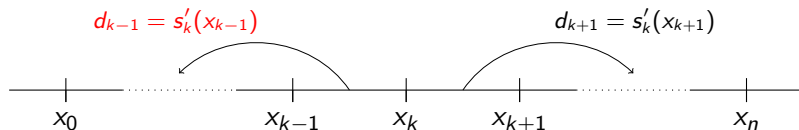
$$\begin{array}{cccc}
 x_k & y_k & & \\
 x_k & y_k & d_k & \\
 x_{k+1} & y_{k+1} & p_{k+1} & \frac{p_{k+1} - d_k}{h_k}
 \end{array}$$

$$s_{k+1}(x) = y_k + d_k(x - x_k) + \frac{p_{k+1} - d_k}{h_{k+1}}(x - x_k)(x - x_k) \quad (1)$$

Método local

Repetimos este proceso actualizando la derivada:

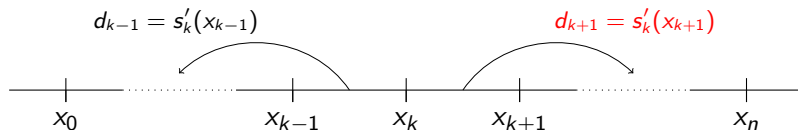
- Hacia la **izquierda**, aplicando la primera fórmula.
- Hacia la derecha, aplicando la segunda fórmula.



Método local

Repetimos este proceso actualizando la derivada:

- Hacia la izquierda, aplicando la primera fórmula.
- Hacia la **derecha**, aplicando la segunda fórmula.



Método global

Resolvemos el sistema, añadiendo la condición para la derivada:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & 0 & \cdots & 0 \\ 1 & x_1 & x_1^2 & (x_1 - x_1)_+^2 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \cdots & \vdots \\ \vdots & & \vdots & \vdots & \cdots & \vdots \\ 1 & x_n & x_n^2 & (x_n - x_1)_+^2 & \cdots & (x_n - x_{n-1})_+^2 \\ 0 & 1 & 2x_k & 2(x_k - x_1)_+ & \cdots & 2(x_k - x_{n-1})_+ \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ \alpha_1 \\ \vdots \\ \alpha_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ \vdots \\ y_n \\ d_k \end{pmatrix}$$

$$s(x) = a + bx + cx^2 + \alpha_1(x - x_1)_+^2 + \cdots + \alpha_{n-1}(x - x_{n-1})_+^2$$

Método global

Resolvemos el sistema, añadiendo la condición para la derivada:

$$\begin{pmatrix}
 1 & x_0 & x_0^2 & 0 & \cdots & 0 \\
 1 & x_1 & x_1^2 & (x_1 - x_1)_+^2 & \cdots & 0 \\
 \vdots & & \vdots & \vdots & \cdots & \vdots \\
 \vdots & & \vdots & \vdots & \cdots & \vdots \\
 1 & x_n & x_n^2 & (x_n - x_1)_+^2 & \cdots & (x_n - x_{n-1})_+^2 \\
 0 & 1 & 2x_k & 2(x_k - x_1)_+ & \cdots & 2(x_k - x_{n-1})_+
 \end{pmatrix}
 \begin{pmatrix}
 a \\
 b \\
 c \\
 \alpha_1 \\
 \vdots \\
 \alpha_{n-1}
 \end{pmatrix}
 =
 \begin{pmatrix}
 y_0 \\
 y_1 \\
 \vdots \\
 \vdots \\
 y_n \\
 d_k
 \end{pmatrix}$$

$$s(x) = a + bx + cx^2 + \alpha_1(x - x_1)_+^2 + \cdots + \alpha_{n-1}(x - x_{n-1})_+^2$$

Método global

Resolvemos el sistema, añadiendo la condición para la derivada:

$$\begin{pmatrix} 1 & x_0 & x_0^2 & 0 & \cdots & 0 \\ 1 & x_1 & x_1^2 & (x_1 - x_1)_+^2 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \cdots & \vdots \\ \vdots & & \vdots & \vdots & \cdots & \vdots \\ 1 & x_n & x_n^2 & (x_n - x_1)_+^2 & \cdots & (x_n - x_{n-1})_+^2 \\ 0 & 1 & 2x_k & 2(x_k - x_1)_+ & \cdots & 2(x_k - x_{n-1})_+ \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ \alpha_1 \\ \vdots \\ \alpha_{n-1} \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ \vdots \\ y_n \\ d_k \end{pmatrix}$$

$$s(x) = a + bx + cx^2 + \alpha_1(x - x_1)_+^2 + \cdots + \alpha_{n-1}(x - x_{n-1})_+^2$$

SplineCuad

Implementamos la función con el **método global**.

- 1 Calculamos la matriz de coeficientes:

```
n = length(x) - 1;  
  
A(:,1) = [ones(n+1,1); 0];  
A(:,2) = [x'           ; 1];  
A(:,3) = [x'.^2         ; 2.*x(k+1)];  
  
for j = 4 : n + 2  
    t = @(s) (s > x(j-2)) .* (s - x(j-2));  
    A(:, j) = [t(x') .^2; 2.*t(x(k+1))];  
end
```


SplineCuad

Implementamos la función con el **método global**.

② Y resolvemos el sistema:

```
sol = A \ [y' ; d_k];

for k = 1:n
    p = sol(3:-1:1)';

    for l = 2:k
        p += sol(l+2).*[1, -2.*x(l), x(l).^2];
    end

    B(k, :) = polyaffine(p,[-x(k) 1]);
end

s = mkpp(x,B);
```

SplineCuadLocal

- 1 Recorremos todos los nodos de $k+1$ en adelante:

```
for i = (k+1):(length(x)-1)
    p = (y(i+1)-y(i))/(x(i+1)-x(i));
    q = (p-d)/(x(i+1)-x(i));
    v = [x(i) x(i)];
    s(i,:) = [0, 0, y(i)]+[0, d, -d*x(i)]+q*poly(v);
    d = polyval(polyder(s(i,:)),x(i+1));
end
d = d_k;
```

SplineCuadLocal

- ② Recorremos todos los nodos desde k hasta el 1:

```
for j = k:-1:1
    p = (y(j+1)-y(j))/(x(j+1)-x(j))
    q = (d-p)/(x(j+1)-x(j))
    v = [x(j) x(j+1)]
    s(j,:) = [0 0 y(j)]+[0 p -p*x(j)]+q*poly(v);
    d = polyval(polyder(s(j,:)), x(j))
end
```

SplineCuadLocal

- ③ Y cambiamos de base usando polyaffine:

```
for i = 1:length(s)
    s(i,:) = polyaffine(s(i,:), [-x(i), 1]);
end
```