CUBIC SPLINE (NATURAL)

L'interpolazione tramite Spline consiste nel dividere un intervallo in varie "porzioni"/"segmenti" e, in ognuno di questi segmenti, approssimiamo i dati, o la funzione, con un polinomio di 3° grado in cui imponiamo alcune condizioni.

Condizioni da imporre per una spline cubica:

1) $S(x) \in \mathbb{P}^3$, $S_J(x)$ rappr. un polinomio di grado 3 sull'intervallo $[x_J, x_{J+1}]$ (per $J=0,\dots,n-1$)

2) $S_J(x_J) = f(x_J)$ e $S_J(x_{J+1}) = f(x_{J+1})$ (per $J=0,\dots,n-1$)

3) $S_{J+1}(x_{J+1}) = S_J(x_{J+1})$ (per $J=0,\dots,n-1$)

4) $S'_{J+1}(x_{J+1}) = S'_J(x_{J+1})$ (per $J=0,\dots,n-1$)

5) $S''_{J+1}(x_{J+1}) = S'_J(x_{J+1})$ (per $J=0,\dots,n-1$)

6) a) $S''(x_0) = S''(x_n) = 0$ Natural Spline

XOR b) $S'(x_0) = f'(x_0) \wedge S'(x_n) = f'(x_n)$

Queste condizioni fanno sì che il polinomio $S(x)$ sia "smooth" ed anche che approssimi correttamente la funzione/dati in analisi.

Come possiamo costruire una Spline?

Prendiamo un S_J generico:

$$S_J(x) \in \mathbb{P}^3 := a_J + b_J(x-x_J) + c_J(x-x_J)^2 + d_J(x-x_J)^3$$

per $J=0,\dots,n-1$

Notiamo subito che $S_J(x_J) = a_J$ e per la condizione 2) si ha che $S_J(x_J) = a_J = f(x_J)$ inoltre combinato alla condizione 3) otteniamo

$a_{j+1} = S_{j+1}(x_{j+1}) = S_j(x_{j+1}) = a_j + b_j(x_{j+1} - x_j) + c_j(x_{j+1} - x_j)^2 + d_j(x_{j+1} - x_j)^3$

per $j = 0, \dots, n-2$

Dato che il termine $x_{j+1} - x_j$ ricorrerà spesso per brevità lo indicheremo con h_j.

Definiamo ora $a_n = f(x_n)$

$$a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3 \quad (\text{per } j = 0, \dots, n-1)$$

In modo analogo possiamo ricavare un'equazione per b_n

Consideriamo $S_j(x) = a_j + b_j x + c_j x^2 + d_j x^3$

$$\Downarrow$$

$$S_j'(x) = b_j + 2c_j x + 3d_j x^2$$

Calcolando $S_j'(x)$ in x_j otteniamo

$$S_j'(x_j) = b_j \; 2c_j(x_j - x_j) + 3d_j(x_j - x_j)$$

$$S_j'(x_j) = b_j \quad (\text{per } j = 0, 1, \dots, n-1)$$

Imponendo ora la clausola 4 otteniamo

$$b_{j+1} = b_j + 2c_j h_j + 3d_j h_j^2 \quad (\text{per } j = 0, \dots, n-1)$$

Definiamo ora $c_n = S''(x_n)/2$ ed imponendo la condizione 5) otteniamo, per $j = 0, 1, \dots, n-1$

$$c_{j+1} = c_j + 3d_j h_j$$

risolvendo rispetto a d_j otteniamo

$$\frac{c_{j+1} - c_j}{3h_j} = d_j \qquad (NB: d_j \neq 0 \quad \forall j = 0, \dots, n-1$$

Ora questa nuova informazione la possiamo reinserire nelle formule già trovate per "semplificarle".

$$a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h^3$$

$\downarrow$ sost. d_j

$$a_{j+1} = a_j + b_j h_j + c_j h_j^2 + h_j^{\cancel{3}\,2} \left(\frac{c_{j+1} - c_j}{3 \cancel{h_j}} \right)$$

$$a_{j+1} = a_j + b_j h_j + h_j^2 \left(c_j + \frac{c_{j+1} - c_j}{3} \right)$$

$$a_{j+1} = a_j + b_j h_j + \frac{h_j^2}{3} (2 c_j + c_{j+1})$$

$$b_{j+1} = b_j + 2 c_j h_j + 3 d_j h_j^2$$

$$b_{j+1} = b_j + 2 c_j h_j + \cancel{3} h_j^{\cancel{2}} \left(\frac{c_{j+1} - c_j}{\cancel{3} \cancel{h_j}} \right)$$

$$b_{j+1} = b_j + h_j (2 c_j + c_{j+1} - c_j)$$

$$b_{j+1} = b_j + h_j (c_{j+1} + c_j) \quad \#$$

Inoltre dalla prima equazione di sopra possiamo anche ricavare una equazione per b_j

$$a_{j+1} = a_j + b_j h_j + \frac{h_j^2}{3} (2 c_j + c_{j+1})$$

$\downarrow$

$$b_j = \frac{a_{j+1} - a_j}{h_j} - \frac{h_j}{3} (2 c_j + c_{j+1})$$

Ora possiamo effettuare una riduzione degli indici per trovare un'equazione per b_{j-1}:

$$b_{j-1} = \frac{a_j - a_{j-1}}{h_{j-1}} - \frac{h_{j-1}}{3} (2 c_{j-1} + c_j)$$

Ora uguagliando questa equazione a quella trovata in (#), ovviamente diminuendo l'indice di 1, otteniamo un sistema lineare, valido per $j = 1, 2, \dots, n-1$ e che quindi può essere visto "come un vettore", in cui l'unico valore sconosciuto è $\{c_j\}_{j=0}^{n}$

Esplicito "le equazioni del sistema"

$$\underbrace{b_{j-1}}_{} + h_{j-1}(c_{j-1} + c_j) = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1})$$

$$\frac{1}{h_{j-1}}(a_j - a_{j-1}) - \frac{h_{j-1}}{3}(2c_{j-1} + c_j) + h_{j-1}(c_{j-1} + c_j) = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1})$$

Da qui possiamo ricavare

$$\frac{h_j}{3}(2c_j + c_{j+1}) - \frac{h_{j-1}}{3}(2c_{j-1} + c_j) + h_{j-1}(c_{j-1} + c_j) = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{1}{h_{j-1}}(a_j - a_{j-1})$$

$$h_j(2c_j + c_{j+1}) - h_{j-1}(2c_{j-1} + c_j) + 3h_{j-1}(c_{j-1} + c_j) = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1})$$

Consideriamo ora solo il membro sulla sx, quello a dx è espresso solo in termini di h ed a che sono noti:

$$2h_j c_j + h_j c_{j+1} - 2h_{j-1} c_{j-1} - h_{j-1} c_j + 3h_{j-1} c_{j-1} + 3h_{j-1} c_j = \ldots$$

$$2h_j c_j + h_j c_{j+1} + h_{j-1} c_{j-1} + 2h_{j-1} c_j = \ldots$$

$$h_j c_{j+1} + h_{j-1} c_{j-1} + c_j(2h_j + 2h_{j-1}) = \ldots$$

$$h_j c_{j+1} + h_{j-1} c_{j-1} + 2c_j(h_j + h_{j-1}) = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1})$$

Possiamo ora individuare tutti i coefficienti della spline cubica, in pochi semplici passaggi.

NB! se abbiamo n intervalli occorrono 4n (circa) condizioni da imporre per ottenere la Spline.

Per ottenere i coefficienti ~~delle~~ "a" basta prendere le valutazioni ~~della~~ f nei punti dati.
Per ricavare i coeffici "c" consideriamo la seguente matrice

$$A = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 & 0 \\ h_0 & 2(h_0+h_1) & h_1 & & & 0 & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \ddots & & & \\ 0 & 0 & \vdots & & \ddots & 0 & - \\ \vdots & \vdots & \vdots & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

il vettore dei termini noti

$$d = \begin{bmatrix} 0 \\ \frac{3}{h_1}(a_2-a_1) - \frac{3}{h_0}(a_1-a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n-a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1}-a_{n-2}) \\ 0 \end{bmatrix}$$

ed il vettore delle incognite

$$x = \begin{bmatrix} c_1 \\ \vdots \\ c_n \end{bmatrix}$$

Risolvendo:

$Ax = d$, rispetto ad x ovviamente
otteniamo il vettore dei coefficienti "c".

NB: In questo caso stiamo considerando come condizioni di contorno $S''(x_0) = S''(x_n) = 0$, da cui

$0 = S''(x_0) = 2c_0 + 6d_0(x_0 - x_0) \Rightarrow c_0 = 0$

Stesso dicasi per $c_n (= 0)$!

Ora conoscendo h, c ed a possiamo ricavarci b e d.

$$b_j = \frac{(a_{j+1} - a_j)}{h_j} - \frac{h_j}{3}(2c_j + c_{j+1})$$

$$d_j = \frac{c_{j+1} - c_j}{3h_j}$$

Una volta ottenuti tutti i valori possiamo valutare la spline in ~~[illegible]~~ qualsiasi punto interno all'intervallo ~~[illegible]~~ $[x_0, x_n]$

TIPS: per capire quali coefficienti usare basta vedere in quale sottointervallo è contenuto il valore da evalutare e possiamo farlo utilizzando una ricerca binaria $(O(\log n))$!

CLAMPED SPLINE!

Come abbiamo visto la Spline naturale si distingue dal fatto che la derivata seconda agli estremi vale 0

$S''(x_0) = S''(x_n) = 0$

Invece le "CLAMPED SPLINE" si distinguono per le condizioni imposte sulle derivate prime agli estremi x_0 e x_n, ovvero $S_0'(x_0) = f'(x_0)$ e $S_{n-1}'(x_n) = f'(x_n)$

Per ottenere questa "nuova" spline basta apportare delle modifiche alla matrice A usata per individuare i coefficienti c!

Osserviamo che

$f'(x_0) = S_0'(x_0) = b_0$ ciò implica che

$$f'(x_0) = \frac{1}{h_0}(a_1 - a_0) - \frac{h_0}{3}(2c_0 + c_1)$$

Da cui

$$2h_0 c_0 + h_0 c_1 = \frac{3}{h_0}(a_1 - a_0) - 3\ell'(x_0)$$

Allo stesso modo otteniamo per x_n:

$$\ell'(x_n) = b_n = b_{n-1} + h_{n-1}(c_{n-1} + c_n)$$

$$= \frac{a_n - a_{n-1}}{h_{n-1}} - \frac{h_{n-1}}{3}(2c_{n-1} + c_n) + h_{n-1}(c_{n-1} + c_n)$$

$$= \frac{a_n - a_{n-1}}{h_{n-1}} + \frac{h_{n-1}}{3}(c_{n-1} + 2c_n)$$

Da cui:

$$h_{n-1} c_{n-1} + 2h_{n-1} c_n = 3\ell'(x_n) - \frac{3}{h_{n-1}}(a_n - a_{n-1})$$

In rosso sono riportati i nuovi termini per la matrice A, rispettivamente per la prima e ultima riga.

$$A = \begin{bmatrix} 2h_0 & h_0 & \cdots & \cdots & 0 & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \cdots & & & \vdots \\ \vdots & \ddots & \ddots & \ddots & & & \vdots \\ \vdots & & \ddots & h_{n-2} & & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & \cdots & \cdots & & h_{n-1} & 2h_{n-1} \end{bmatrix}$$

In blu sono evidenziati la prima componente, e anche l'ultima, del vettore d usato per trovare c ($c = A/d$).

NOT AKNOT SPLINE

Non è altro che una variante della spline naturale in cui in più si aggiunge la richiesta che la derivata terza della spline $S(x)$ sia continua in x_0 e x_n, ovvero che $S_0'''(x_0)$ e $S_{n-1}'''(x_n)$ siano continue.

Per fare ciò solitamente si assegna un valore specifico.

Come facciamo ad essere sicuri che S_0''' e S_{n-1}''' siano continue?

$$S_j(x) = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3$$

$$S_j'(x) = b_j + 2c_j h_j + 3 d_j h_j^2$$

$$S_j''(x) = 2c_j + 6 d_j h_j$$

$$S_j'''(x) = 6 d_j$$

Basta imporre che $6 d_j$ sia definita.

Solitamente una delle condizioni più usate è

$$S_0'''(x_1) = S_1'''(x_1) \quad \text{e} \quad S_{n-2}'''(x_{n-1}) = S_{n-2}'''(x_{n-1})$$

e ciò può essere semplicemente calcolato computando una spline naturale ~~...~~ e successivamente impostando $d_1 = d_2$ e $d_{n-1} = d_{n-2}$.

TIPS: non si ha nessun vantaggio particolare nell'usare la distribuzione di Chebyshev per i nodi.

È di gran lunga preferibile concentrare la distribuzione dei nodi ~~...~~ nei punti in cui la funzione cresce più velocemente oppure ha dei cambi di convessità.

binary_search.m   cubic_spline.m

home > antonio > elaborati_matlab > cubic spline > cubic_spline.m > function [f] = cubic_spline(xdata, ydata, z, type, der_x0, der_xn)

```matlab
function [f] = cubic_spline(xdata, ydata, z, type, der_x0, der_xn)
    %type indica quale tio di spline viene utilizzata basta indicare il tipo
    %con una stringa
    %der_x0 e der_xn devono contenere il valore della derivata prima in x0 e xn
    %questi due valori vengono utilizzati solo nel caso di Clamped Spline
    %Sj(x) = aj + bj(x-xj) + cj(x-xj)^2 + dj(x-xj)^3
    %a = ydata; coefficienti del termine noto
    n = length(xdata);
    h = zeros(n-1, 1);%vettore che contiene gli intervalli
    aus = zeros(1, n);%vettore ausiliario usato per trovare le costanti c
    d = zeros(1, n);%coefficiente del termine cubico
    A = zeros(n);%matrice usata per trovare le costanti c
    for i=1:n-1
        h(i) = xdata(i+1)-xdata(i);
    end
    for i=2:n-1
        aus(i) = 3*(ydata(i+1) - ydata(i))/h(i) + 3*(ydata(i)-ydata(i-1))/h(i-1);
    end
    if strcmp(type, 'Clamped')
        aus(1) = (3/h(1))*(ydata(2)-ydata(1))-3*der_x0;
        aus(n) = 3*der_xn - (3/h(n-1))*(ydata(n)-ydata(n-1));
        A(1, 1) = 2*h(1);
        A(1, 2) = h(1);
        A(n, n) = 2*h(n-1);
        A(n, n-1) = h(n-1);
    else
        aus(1) = 0;
        aus(n) = 0;
        A(1, 1) = 1;
        A(n, n) = 1;
    end
    A(2:1+n:n*n-2*n-1) = h(1:n-2);
    A(2*n+2:1+n:n*n-1) = h(2:n-1);
    A(2+n:1+n:n*n-(n+1)) = 2*(h(1:n-2)+h(2:n-1));
    c = A/aus;%vettore dei coefficienti
    for i=1:n-1
        d(i) = (c(i+1)-c(i))/3*h(i);
    end
    b = zeros(n,1);
    for i=1:n-1
        b(i) = (1/h(i))*(ydata(i+1)-ydata(i)) - (1/3*h(i))*(2*c(i)+c(i+1));
    end
    m = length(z);%numero di punti in cui valutare la spline
    f = zeros(m, 1);%vettore che conterra' le valutazioni della spline
    if strcmp(type, 'Not-a-Knot')
        d(1) = d(2);
        d(n-1) = d(n-2);
    end
    for i=1:m
        index = binary_search(xdata, z(i));
        hj = (z(i)-xdata(index));%computa x-xj
        f(i) = ydata(index) + b(index)*hj + c(index)*hj^2 + d(index)*hj^3;
    end
end
```

0 0   Ln 1, Col 1   Spaces: 4   UTF-8   LF   MATLAB   1

binary_search.m × cubic_spline.m

home > antonio > elaborati_matlab > cubic spline > binary_search.m > function [index] = binary_search(vector, value)

```matlab
function [index] = binary_search(vector, value)
    up = length(vector);
    low = 1;
    if up==1
        index = 1;
    else
        while up ~= low
            index = (low + up)/2;
            if value <= vector(ceil(index)) && value >= vector(floor(index))
               index = floor(index);
               break;
            elseif value < vector(floor(index))
                    up = floor(index);
            else
                low = ceil(index);
            end
        end
    end
end
```