INTERPOLAZIONE CON METODO DI NEWTON "DIFFERENZE DIVISE"

Il grande vantaggio di questo "tipo di interpolazione" è che può sfruttare l'algoritmo di Horner per essere ~~calcolato~~ valutato in punto (NB: è sempre una interpolazione polinomiale).
Invece che la base canonica dei monomi per lo spazio dei polinomi utilizziamo una base così definita:

$$B = \{(x-x_0), (x-x_0)(x-x_1), \dots, (x-x_0)(x-x_1)\dots(x-x_n)\}$$

Quindi il polinomio sono della seguente forma

$$p_n(x) = a_0 + a_1(x-x_0) + \dots + a_n(x-x_0)\dots(x-x_n)$$

Come al solito dobbiamo imporre il passaggio per i punti dati:

$$y_i = f(x_i) \quad \text{per } i = 1, \dots, n$$

Successivamente imponiamo le condizioni di interpolazione
Partiamo da x_0:

- $f(x_0) = y_0 = p_n(x_0)$
  tutti i termini che contengono x_0 si annullano lasciando solamente a_0, quindi $a_0 = y_0$.

Consideriamo ora x_1!

- $p_n(x_1) = y_1 = f(x_1)$
  allo stesso modo si eliminano tutti i termini che contengono x_1 lasciando solamente:

$$p_n(x_1) = y_1 = a_1(x-x_0) + a_0 \Rightarrow a_1 = \frac{y_1 - y_0}{x_1 - x_0} = f[x_0, x_1]$$

praticamente i coefficienti del polinomio sono i rapporti delle differenze, da qui cui il nome del metodo

Il ragionamento prosegue considerando polinomi di grado sempre maggiore fino ad un ordine prefissato n.
Consideriamo ora una differenza divisa di ordine d:

$$f[x_k, x_{k+1}, \dots, x_{k+d}] = \frac{f[x_{k+1}, \dots, x_{k+d}] - f[x_k, \dots, x_{k+d-1}]}{x_{k+d} - x_k}$$

Ricordando che:

$$f[x_k] = y_k = f(x_k)$$

$$f[x_k, x_{k+1}] = \frac{f[x_{k+1}] - f[x_k]}{x_{k+1} - x_k}$$

Possiamo ricavare tutte le diff. divise che ci occorrono grazie ad una matrice

$$\begin{array}{llllll}
x_0 \to & \boxed{y_0} & & & & \\
 & & \boxed{f[x_0, x_1]} & & & \\
x_1 \to & y_1 & & \boxed{f[x_0, x_1, x_2]} & & \\
 & & f[x_1, x_2] & & \boxed{f[x_0, x_1, x_2, x_3]} & \\
x_2 \to & y_2 & & f[x_1, x_2, x_3] & & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \boxed{f[x_0, \dots, x_n]} \\
 & & f[x_{n-1}, x_n] & & & \\
x_n \to & y_n & & & &
\end{array}$$

I termini che ci occorrono sono solo quelli sulla diagonale principale

NB: il costo computazionale del polinomio di Lagrange e delle differenze divise è lo stesso ($O(N^2)$), quindi non vi è un metodo preferito rispetto ad un altro, quindi la scelta tra queste 2 tecniche va fatta in base al contesto in cui occorrono.

TIPS: Si può utilizzare la distribuzione di Chebyshev per una "migliore approssimazione".

DISTRIBUZIONE DEI NODI

Si potrebbe pensare che la miglior distribuzione di nodi sia una equidistante, in realtà non si hanno particolari svantaggi o vantaggi, ma esistono "distribuzioni" migliori.

Esempio sui punti equispaziati:

$x_i = x_0 \cdot h i$ con $i=0,\dots,n$

$\omega_0(x) = (x-x_0)(x-x_1)$

Come possiamo maggiorare questa quantità?

Osserviamo che in $x = \frac{x_0+x_1}{2}$ $|\omega_2(x)|$ è massimo

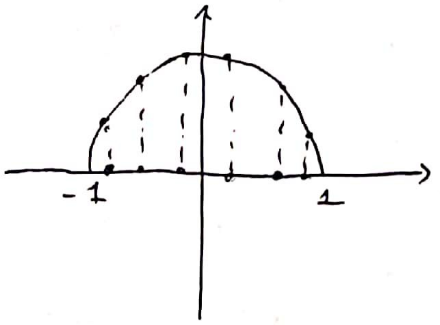
$$|\omega_2(x)| = \frac{(x_1-x_0)^2}{\cancel{2}4} = \frac{h^2}{4}$$

Se consideriamo n punti equidistanti:

$$\underbrace{|\omega_{n+1}(x)| \le n!\, \frac{h^{n+1}}{4}}$$

(Si può dimostrare per induzione)

DISTRIBUZIONE DI CHEBYSHEV



Prendiamo dei punti equidistanti su una semicirconferenza e infine prendiamo come nodi le proiezioni di quest'ultimi sulla asse delle ascisse.

$$x_k = \frac{a+b}{2} + \frac{b-a}{2}\cos\left(\frac{2k+1}{2(n+1)}\pi\right) \qquad k=0,\dots,n$$

↳ Riduce l'errore sull'oscillazione dei nodi chebyshev.m | divided_difference.m | divided_difference_multi.m

home > antonio > elaborati_matlab > differenze divise > divided_difference_multi.m > function [f] = divided_difference_multi(xdata, ydata, z)

```matlab
function [f] = divided_difference_multi(xdata, ydata, z)
    n = length(xdata);
    A = size(n);
    A = [ydata'];
    for i=2:n
        for j=2:i
            A(i, j) = (A(i,j-1)-A(i-1,j-1))/(xdata(i)-xdata(i-j+1));
        end
    end
    m = length(z);
    f = zeros(m, 1);
    %praticamente questa e' una implementazione dell'algoritmo di horner
    for k=1:m
        f(k) = A(n,n);
        for i=n-1:-1:1
            f(k) = f(k)*(z(k)-xdata(i))+A(i,i);
        end
    end
end

```

⊗ 0 ⚠ 0   Ln 1, Col 1   Spaces: 4   UTF-8   LF   MATLAB chebyshev.m   divided_difference.m   divided_difference_multi.m

home > antonio > elaborati_matlab > differenze divise > divided_difference.m > function [f_in_x0] = divided_difference(xdata, ydata, x0)

```
function [f_in_x0] = divided_difference(xdata, ydata, x0)
    n = length(xdata);
    A = size(n);
    A = [ydata'];
    for i=2:n
        for j=2:i
            A(i, j) = (A(i,j-1)-A(i-1,j-1))/(xdata(i)-xdata(i-j+1));
        end
    end
    f_in_x0 = A(n,n);
    for i=n-1:-1:1
        f_in_x0 = f_in_x0*(x0-xdata(i))+A(i,i);
    end
end
```

ⓧ 0 ⚠ 0   Ln 1, Col 1   Spaces: 4   UTF-8   LF   MATLAB chebyshev.m | divided_difference.m | divided_difference_multi.m

home > antonio > elaborati_matlab > polinomio di Lagrange > chebyshev.m > ...

```matlab
function [distribution] = chebyshev(a, b, k)
    distribution = zeros(k, 1);
    for i=1:k
        distribution(i) = (a+b)/2 + (b-a)*cos((2*i*pi + pi)/(2*k + 2))/2;
    end
end

```

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