INTERPOLAZIONE POLINOMIALE

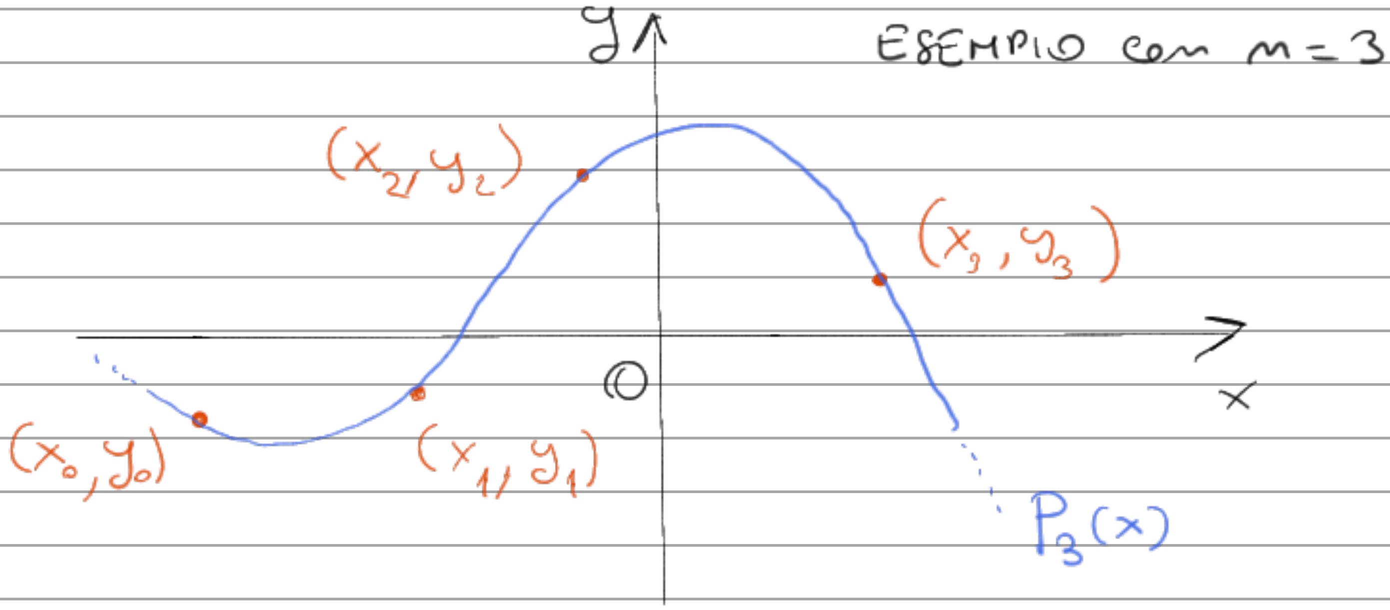
DEFINIZIONE DEL PROBLEMA:

Siano dati $n+1$ punti del piano cartesiano:

$$(x_i, y_i), \quad i=0,\dots,n$$

Determinare un polinomio $P_n(x)$, di grado al più n, il cui grafico passi per i punti assegnati, ovvero

(1) $$P_n(x_i)=y_i, \quad i=0,\dots,n$$



Mostriamo che, nelle ipotesi di ascisse x_i distinte, il polinomio che verifica (1) esiste ed è unico. In tal caso:

$P_n(x) \longrightarrow$ polinomio interpolante

Le condizioni $\begin{cases} P_n(x_i)=y_i \\ i=0,\dots,n \end{cases} \longrightarrow$ condizioni di interpolazione le ascisse x_i sono dette nodi

Proposizione

Siano $(x_i, y_i) \in \mathbb{R}^2$, $i=0, \dots, n$

con nodi distinti: $x_i \neq x_j$ se $i \neq j$.

Allora esiste ed è unico il polinomio $P_n(x)$

di grado al più n che soddisfa le

condizioni di interpolazione

$$P_n(x_i) = y_i \quad , i=0, \dots, n$$

DIM

Consideriamo il generico polinomio di

grado al più n scritto lungo la base delle

potenze:

$$P_n(x) = a_0 + a_1 x + \dots + a_n x^n$$

e imponiamo le condizioni di interpolazione:

$$\begin{cases} a_0 + x_0 a_1 + x_0^2 a_2 + \dots + x_0^n a_n = y_0 \\ a_0 + x_1 a_1 + x_1^2 a_2 + \dots + x_1^n a_n = y_1 \\ \vdots \\ a_0 + x_n a_1 + x_n^2 a_2 + \dots + x_n^n a_n = y_n \end{cases}$$

Abbiamo ottenuto un sistema lineare

di $n+1$ equazioni nelle $n+1$ incognite

$a_0, a_1, \dots, a_n$

La matrice dei coefficienti risulta:

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix}$$

V è detta matrice di Vandermonde.

Si può dimostrare che

$$\det(V) = \prod_{i>j} (x_i - x_j)$$

(omettiamo la dimostrazione)

Essendo i nodi distinti, osserviamo che $\det(V) \neq 0$.

Quindi il sistema lineare ammetterà soluzione unica. $\square$

ESERCIZIO

1) Scrivere una funzione Python che abbia in input un vettore $\underline{x}$ e restituisce in output la matrice **V** di Vandermonde

2) Scrivere una funzione python che abbia in input due vettori $\underline{x}$ e $\underline{y}$ della stessa lunghezza e restituisca, in output, i coefficienti del polinomio interpolante in un vettore $\underline{a}$

Purtroppo la matrice di Vandermonde diventa mal condizionata anche per valori non grandi di n (es: $n \simeq 10$).
Verificarlo in python mediante la funzione linalg.cond

Al fine di risolvere questo problema di mal condizionamento è possibile scrivere il polinomio $p_n(x)$ lungo una base diversa dalla base delle potenze. In letteratura si utilizzano le seguenti basi:

- BASE DI LAGRANGE
- BASE DI NEWTON

Discuteremo unicamente l'approccio che usa la base di Lagrange.

Polinomi di Lagrange

DEF Siano dati i nodi distinti dell'intervallo $[a,b]$: $x_0, x_1, \dots, x_n$.
Per $k = 0, \dots, n$, si definisce k-esimo polinomio cardinale di Lagrange, il seguente polinomio di grado n:

$$L_k(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}$$

In forma compatta:

$$L_k(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x-x_i}{x_k-x_i}$$

Proposizione $\{L_k(x)\}_{k=0,\dots,n}$ sono linearmente indipendenti (visti come elementi dello spazio vettoriale Π_n, spazio dei polinomi di grado al più n).

DIM Consideriamo una combinazione lineare nulla dei polinomi $L_k(x)$:

(*) $$\sum_{k=0}^{n} \alpha_k L_k(x) = 0, \qquad \forall x \in [a,b]$$

Vogliamo provare che $\alpha_0 = \alpha_1 = \dots = \alpha_n = 0$.

Premettiamo la seguente proprietà:

$$\forall i = 0, \dots, n \qquad L_k(x_i) = \begin{cases} 0, & \text{se } i \neq k, \\ 1, & \text{se } i = k. \end{cases}$$

Ricordando la def. del simbolo di Kronecker:

$$\delta_{ik} = \begin{cases} 0, & \text{se } i \neq k \\ 1, & \text{se } i = k \end{cases}$$

risulta quindi: $L_k(x_i) = \delta_{ik}$

Valutiamo ora l'espressione (*) nel nodo x_i: $x = x_i$

$$0 = \sum_{k=0}^{n} \alpha_k L_k(x_i) = \sum_{k=0}^{n} \alpha_k \delta_{ik} = \alpha_i \delta_{ii} = \alpha_i$$

$\square$

Essendo $\dim(\Pi_n) = n+1$, ne consegue che $\{L_k(x)\}_{k=0,\dots,n}$ formano una base di Π_n.

Polinomio interpolante di Lagrange

Dal risultato precedente, possiamo esprimere un generico polinomio $P_m \in \Pi_m$ come combinazione lineare degli elementi della base $\{L_k(x)\}$:

$$P_m(x) = \alpha_0 L_0(x) + \alpha_1 L_1(x) + \dots + \alpha_m L_m(x)$$
$$= \sum_{k=0}^{m} \alpha_k L_k(x)$$

Imponiamo le condizioni di interpolazione:

$$P_m(x_i) = y_i \,, \qquad i = 0, \dots, m$$

Risulta:

$$y_i = P_m(x_i) = \sum_{k=0}^{m} \alpha_k L_k(x_i) = \sum_{k=0}^{m} \alpha_k \delta_{ik} = \alpha_i$$

Quindi il polinomio interpolante è:

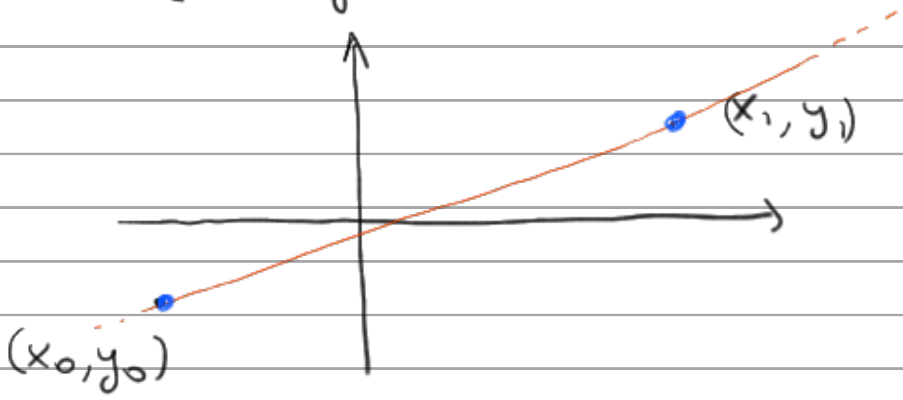
$$P_m(x) = \sum_{k=0}^{m} L_k(x) \cdot y_k$$

Polinomio interpolante di Lagrange

Per la sua costruzione non è richiesta la risoluzione di un sistema lineare.

Esempio 1 $\quad n=1$ (due punti)

x	y
x_0	y_0
x_1	y_1



$$L_0(x) = \frac{x-x_1}{x_0-x_1} \quad ; \quad L_1(x) = \frac{x-x_0}{x_1-x_0}$$

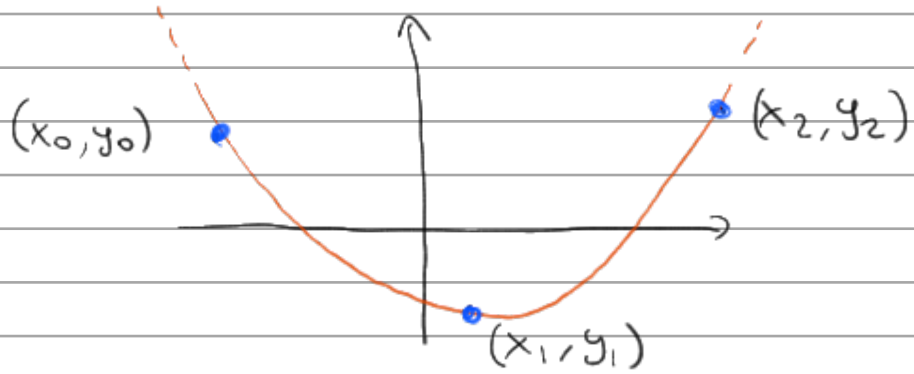
L'equazione della retta passante per i due punti è:

$$y = L_0(x)\cdot y_0 + L_1(x)\cdot y_1$$

$$= \frac{x-x_1}{x_0-x_1}\cdot y_0 + \frac{x-x_0}{x_1-x_0}\cdot y_1$$

Esempio 2 Determinare l'equazione della parabola passante per i tre punti $(x_0,y_0), (x_1,y_1), (x_2,y_2)$

x	y
x_0	y_0
x_1	y_1
x_2	y_2



$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}, \quad L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}$$

$$L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$$

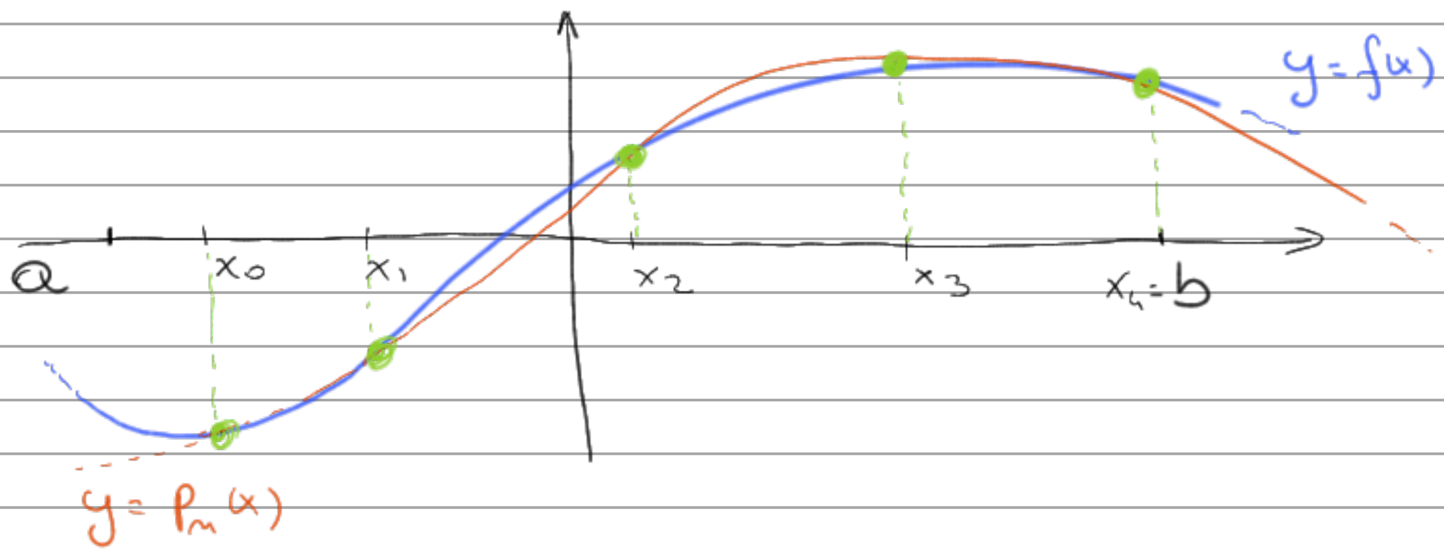
$$y = L_0(x)\cdot y_0 + L_1(x)\cdot y_1 + L_2(x)\cdot y_2$$

Interpolazione di una funzione mediante un polinomio.

Il polinomio interpolante può essere utilizzato per approssimare una funzione all'interno di un intervallo $[a,b]$.

Sia $f: [a,b] \to \mathbb{R}$ e siano $x_0, x_1, \dots, x_n$ nodi distinti di $[a,b]$.
Il polinomio che interpola la funzione f nei nodi x_i è il seguente:

$$P_n(x) = \sum_{k=0}^{n} L_k(x) f(x_k)$$



Supponiamo di lavorare con $n+1$ nodi equidistanti dell'intervallo $[a,b]$:

$$h = \frac{b-a}{n} ; \qquad \begin{cases} x_i = a + ih \\ i = 0, \dots, n \end{cases}$$

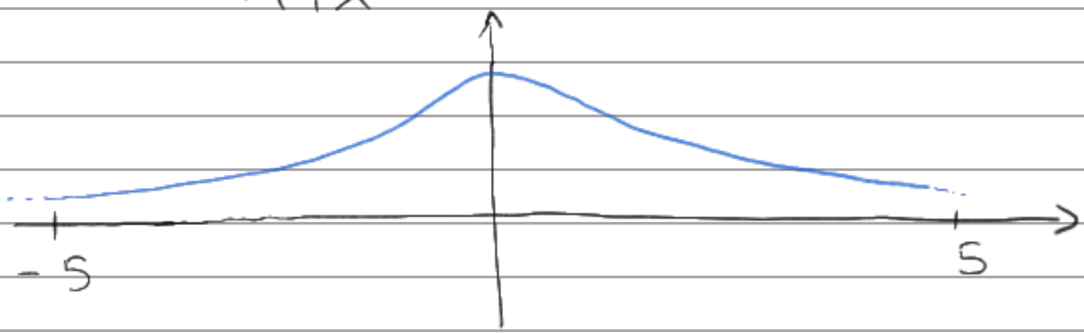
(quindi $x_0 = a$, $x_1 = a+h$, $x_2 = a+2h, \dots, x_n = b$)

Purtroppo, non è detto, in generale, che:

$$\lim_{n \to \infty} p_n(x) = f(x)$$

Controesempio (funzione di Runge)

$$f(x) = \frac{1}{1+x^2}, \qquad x \in [-5,5]$$



Studiamo ora l'errore che si commette quando una funzione è approssimata da un polinomio interpolante.

DEF. La norma infinito (o assoluta) nello spazio vettoriale $C([a,b])$ è così definita:

$$\|f\|_\infty = \max_{a \le x \le b} |f(x)|$$

Tale norma induce la distanza:

$$d(f,g) = \|f-g\|_\infty = \max_{a\le x\le b} |f(x)-g(x)|$$

Ci proponiamo di stimare la distanza (errore) tra una funzione e il suo polinomio interpolante.

Proposizione Sia $f \in C^{n+1}([a,b])$.
Allora, per ogni $x \in [a,b]$, esiste $\xi_x \in [a,b]$ tale che

$$f(x) - p_n(x) = \omega_n(x) \cdot \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \qquad (1)$$

dove:

$$\omega_n(x) = (x-x_0)(x-x_1)\cdots(x-x_n) = \prod_{i=0}^{n} (x-x_i)$$

(polinomio di grado $n+1$)

~~OSSERVAZIONE: facendo tendere~~ $\cancel{x_1, \dots, x_n}$ ~~al nodo~~ $\cancel{x_0}$,

$$\cancel{P_n(x) \longrightarrow \text{polinomio di Taylor sviluppato in un intorno del punto } x_0}$$

~~mentre il secondo membro di (1) diventa il resto di Lagrange.~~

DIM Se $x = x_i$, $i = 0, \dots, n$, la (1) è ovviamente vera. Pertanto fissiamo un valore $x \neq x_i$. Consideriamo la seguente funzione ausiliaria nella variabile $t \in [a,b]$:

$$g(t) = \omega_n(x)\left(f(t) - p_n(t)\right) - \omega_n(t)\left(f(x) - p_n(x)\right)$$

Risulta:

$$g(x_i) = 0, \quad \forall\, i = 0, \dots, n$$

$$g(x) = 0$$

La funzione g si annulla in $n+2$ punti distinti di $[a,b]$. Inoltre $g \in C^{n+1}([a,b])$.

Per il teorema di Rolle-Fermat:

$g'(t)$ si annulla in $n+1$ punti distinti di $[a,b]$

$\Downarrow$

$g''(t)$ si annulla in n punti distinti di $[a,b]$

$\vdots$

$g^{(n+1)}(t)$ si annulla in un punto di $[a,b]$

Cioè: $\exists\, \xi_x \in [a,b]$ t.c. $g^{(n+1)}(\xi_x) = 0$

Osserviamo che:

$$\frac{d^{n+1}}{dt^{n+1}} P_n(t) = 0 \qquad \left(\begin{array}{c} P_n(t) \text{ è un polinomio} \\ \text{di grado } n \end{array} \right)$$

Essendo poi $\omega_n(t) = t^{n+1} + q_n(t)$

polinomio di grado n

abbiamo:

$$\frac{d^{n+1}}{dt^{n+1}} \omega_n(t) = (n+1)!$$

Risulta quindi:

$$g^{(n+1)}(t) = \omega_n(x) \cdot f^{(n+1)}(t) - (n+1)! \left(f(x) - P_n(x) \right)$$

Per $t = \eta_x$ si ottiene quindi:

$$0 = g^{(n+1)}(\eta_x) = \omega_n(x) \cdot f^{(n+1)}(\eta_x) - (n+1)! \left(f(x) - P_n(x) \right)$$

da cui:

$$f(x) - P_n(x) = \omega_n(x) \cdot \frac{f^{(n+1)}(\eta_x)}{(n+1)!} \qquad \square$$

Esercizio in Python

1) Scrivere una function python che abbia:

INPUT:

x array dei nodi $(x_0, ---, x_n)$

y array di ordinate $(y_0, ---, y_n)$

xx array di ascisse

OUTPUT:

yy array che contiene le valutazioni del polinomio che interpola i dati x, y, nei punti dell'array xx.

In altri termini, indicato con $p_n(x)$ il polinomio interpolante $(p_n(x_i) = y_i)$

$$yy(i) = p_n(xx(i))$$

2) Utilizzare la funzione al punto precedente all'interno di una nuova funzione che abbia:

INPUT: funzione f

$a, b,$ estremi dell'intervallo

n: numero di nodi equidistanti dell'intervallo $[a, b]$

OUTPUT: grafico che rappresenti, nell'intervallo $[a, b]$,

- il grafico della funzione f,
- i punti (x_i, y_i), $i = 0, \dots, n$
- il grafico del polinomio $P_n(x)$

3) Eseguire la funzione al punto 2) sui seguenti esempi, e mostrare cosa succede al crescere di n.

(a) $$\begin{cases} f(x) = e^{-x} \cdot \sin(2x), \\ x \in [0, \pi]. \end{cases}$$

(b) $$\begin{cases} f(x) = \dfrac{1}{1+x^2}, \\ x \in [-5, 5]. \end{cases}$$ (funzione di Runge)