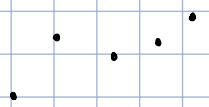


Laurea Triennale Informatica
Approssimazione di dati e funzioni

R. Vermiglio

DIPARTIMENTO DI SCIENZE MATEMATICHE, INFORMATICHE E FISICHE
Università degli Studi di Udine

Ho i vari dati (che sono dei punti, ad es. un campionamento) e voglio legare assieme queste informaz.



ho due approcci:

- **INTERPOLAZIONE**:



ovvero costruire una funzione che passi per quei punti

↳ **INTERPOLANTE**

le funzioni semplici, da considerare per approssimare, sono:

infatti l'**INTERPOLANTE** è complessa (molte volte) e perciò
io cerco una funzione ϕ che la approssimi.



→ interpolazione se ho pochi DATI e NON AFFETTI DA ERRORE

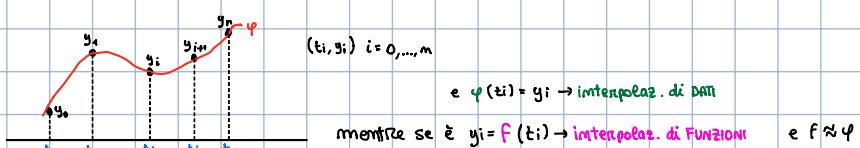
↳ legata ad un sist. QUADRATO

Interpolazione **polinomiale**

Interpolazione **polinomiale a tratti**

Interpolazione **spline**

Interpolazione **trigonometrica e FFT**



e $p(t_i) = y_i \rightarrow$ interpolaz. di DATI

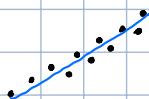
mentre se è $y_i = F(t_i) \rightarrow$ interpolaz. di FUNZIONI

e $F \approx p$

con p **INTERPOLANTE** e t_i modo di interpolaz.

- **MIGLIOR APPROSSIMAZIONE**:

quando ho tanti dati



spesso conviene cercare una funzione ϕ che passa "il più vicino possibile" a tutti quei dati.

→ migliore approssimaz. se ho TANTI DATI affetti da ERRORE

↳ legata ad un sist. SOVRADET.

Interpolazione di dati o funzioni

Assegnati i dati $(t_i, y_i), i = 0, \dots, n$, con $t_0 < t_1 < \dots < t_n$ determinare una funzione φ tale che

$$\varphi(t_i) = y_i, i = 0, \dots, n$$

- φ si chiama **funzione interpolante** o **interpolante**
 - $t_i, i = 0, \dots, n$ sono i **nodi di interpolazione**

In alcune applicazioni è utile imporre ulteriori condizioni alla φ quali, per esempio, avere una certa pendenza in alcuni punti assegnati, o essere regolare, monotona, convessa o periodica.

Se i dati risultano da un campionamento di una funzione f , i.e.

$$y_i = f(t_i), i = 0, \dots, n,$$

l'interpolazione permette di approssimare la funzione f con una funzione φ .

Interpolazione: applicazioni

L'interpolazione si applica

- alla ricostruzione di curve sufficientemente lisce che passano attraverso dei punti assegnati
 - valutazione facile e veloce di una funzione f
 - sostituire una funzione "complicata" con una "semplice"
 - per ottenere dei valori in punti t diversi dai dati osservati t_i
 - per approssimare le derivate di una funzione f
 - per approssimare gli integrali di una funzione f

L'interpolazione **non** è una tecnica appropriata quando i dati del problema sono affetti da errori significativi: in questo caso la tecnica della **miglior approssimazione** è da preferire.

Interpolazione: considerazioni generali

Come scegliere la funzione φ ? → interpolante

ho $m+1$ dati, ovvero ho $m+1$ vincoli

La scelta più opportuna è suggerita dal contesto applicativo e dalle proprietà che si vogliono ottenere, come mantenere la periodicità o la monotonia dei dati, proprietà di lisciezza, convessità,...

La caratteristica fondamentale dell'interpolante è che si esprima come segue

$$\varphi(t) = \sum_{j=0}^n \underbrace{x_j \phi_j(t)}_{\substack{\text{coeff.} \\ \text{fumz. base}}} \rightarrow \text{COMBINAZ. LINEARE di fumz. semplici} \downarrow \text{ha } m+1 \text{ fumz. base}$$

con $\phi_0(t), \dots, \phi_n(t)$ **funzioni base**. Le condizioni di interpolazione portano al seguente **sistema di equazioni lineari** per la determinazione dei coefficienti x_i che definiscono l'interpolante

$$\left\{ \begin{array}{l} \text{h0} \quad \varphi(t) = \sum_{j=0}^m x_j \phi_j(t) \quad \text{e sostituisco } t \text{ con } t_i \rightarrow \varphi(t_i) = \sum_{j=0}^m x_j \phi_j(t_i) = y_i \\ \sum_{j=0}^n x_j \phi_j(t_i) = y_i, \quad i = 0, \dots, n \end{array} \right.$$

Interpolazione, continua

Introducendo la matrice

$$A = \begin{pmatrix} \phi_0(t_0) & \phi_1(t_0) & \cdots & \phi_n(t_0) \\ \phi_0(t_1) & \phi_1(t_1) & \cdots & \phi_n(t_1) \\ \vdots & \ddots & \ddots & \vdots \\ \phi_0(t_n) & \phi_1(t_n) & \cdots & \phi_n(t_n) \end{pmatrix}$$

ed i vettori

$$y = (y_0, \dots, y_n)^T, \quad x = (x_0, \dots, x_n)^T$$

le **equazioni** si scrivono in forma matriciale

$$Ax = y$$

Le proprietà e la struttura della matrice A dipendono sia dalla scelta delle funzioni base $\phi_j, j = 0, \dots, n$ che dai nodi di interpolazione.

Interpolazione, continua

Le famiglie di funzioni φ solitamente usate sono:

- Polinomi \rightsquigarrow Interpolazione polinomiale
- Polinomi a tratti \rightsquigarrow Interpolazione polinomiale a tratti
- Splines \rightsquigarrow Interpolazione splines \rightarrow Polinomi a tratti con condizioni
- Polinomi trigonometrici \rightsquigarrow Interpolazione trigonometrica
- Funzioni razionali \rightsquigarrow Interpolazione razionale

In sostanza decido di usare come INTERPOLANTE un polinomio

se tutti i punti sono DISTINTI allora il polinomio è UNICO.

Decido di rappresentare il polinomio in forma MATRICIALE

$$\begin{pmatrix} 1 & t_0 & \dots & t_0^n \\ 1 & t_1 & \dots & t_1^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & \dots & t_n^n \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

detta MATRICE di VANDERMONDE

il problema è che questa matrice risulta MAL CONDIZIONATA al crescere del grado del polinomio

Rappresentazioni più efficienti : - LAGRANGE : rappresentazione classica che può essere migliorata con la RAPPRESENTAZIONE BARICENTRICA
↳ per il calcolo dei coefficienti

- NEWTON : sceglie delle funzioni base che fanno sì che la matrice dei coeff. del sist. lineare che vado a risolvere risulti TRIANGOLARE INFERIORE
↳ calcolo dei coeff. con SOSTITZ. IN AVANTI
ed è più efficiente nel caso di aggiunta di un nuovo punto

- RAPPRESENTAZ. DIVISE (fa parte dei Newton)

Interpolazione polinomiale

Scelgo come **INTERPOLANTE** un **POLINOMIO** di grado m (quindi con $m+1$ coeff) \rightarrow Se i punti sono distanti tra loro allora il polinomio è unico

Con le funzioni base $\phi_i(t) = t^i$, $i = 0, \dots, n$ (**monomi**) il **polinomio interpolante di grado n** si rappresenta

$$p_n(t) = \underbrace{x_0}_{\text{coeff}} + x_1 t + \dots + x_n t^n$$

e i coefficienti sono ottenuti risolvendo il sistema

$$\begin{pmatrix} 1 & t_0 & \dots & t_0^n \\ 1 & t_1 & \dots & t_1^n \\ \vdots & \vdots & \ddots & \vdots \\ 1 & t_n & \dots & t_n^n \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}.$$

La matrice del sistema si chiama **di Vandermonde** e la indichiamo con V .

Se i nodi di interpolazione sono distinti, V è non singolare e quindi
 \hookrightarrow e quindi **INVERTIBILE**

esiste ed è unico il polinomio $p_n(t)$ di grado al più n che interpola i dati (t_i, y_i) , $i = 0, \dots, n$.
 \hookrightarrow il grado potrebbe essere anche $< n$

Per calcolare il polinomio in un punto, uso l'algo. di HORNER

$$P_m(t) = x_m t^m + \dots + x_0 \\ = ((\dots (x_m t + x_{m-1}) t + x_{m-2}) t \dots) t + x_0$$

se ho t dato e voglio calcolare $\bar{p} = p_m(t) \Rightarrow \bar{p} = x_m \rightarrow$ algoritmo EFFICIENTE per calcolare il polinomio dato un punto
for $k=1..m$

$$\begin{cases} \bar{p} = \bar{p} \cdot t + x_{m-k} \\ m \text{ operaz + m somme} = 2m \text{ op. arit} \rightarrow \text{STABILE} \end{cases}$$

end

può essere che, dati m punti, il polinomio abbia grado $g < m$, es. $m=3$

ma il polinomio che vi passa è
una parabola di grado 2

oppure

che ha grado 1



Interpolazione polinomiale, continua

Per determinare numericamente i coefficienti x_i del polinomio interpolante, dobbiamo calcolare la soluzione del sistema lineare con $O(n^3)$ operazioni.

→ perché la calcolo

La matrice V risulta malcondizionata, specialmente per n grandi, → i coefficienti x_i possono risultare poco accurati. → porta imprecisione nella ricostruzione dell'interpolante.

Example

Siano $t_i = i \times 0.2$, $i = 0, \dots, 10$. La stima del condizionamento della matrice di Vandermonde in Matlab è $cond(V) = 3.6980e + 08$, $\underbrace{8}_{10^8}$

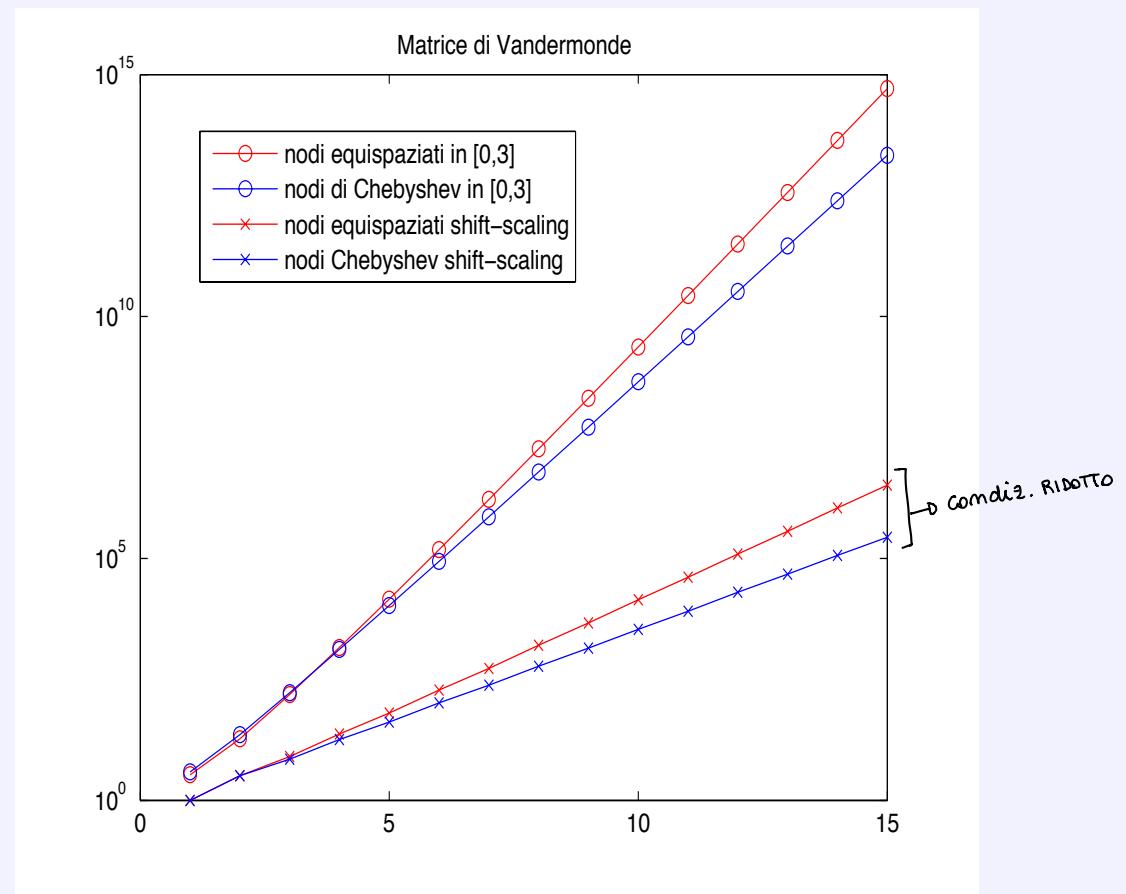
Il condizionamento può essere leggermente migliorato considerando

$$\phi_i(t) = \left(\frac{t - c}{d}\right)^i, i = 0, \dots, n$$

con $c = (t_0 + t_n)/2$ (*shift*) e $d = (t_n - t_0)/2$ (*scaling*). Tali funzioni base aiutano ad evitare overflow e/o underflow. ↳ così riportato i modi in $[1; 1]$

Example

Siano $t_i = i \times 0.2$, $i = 0, \dots, 10$. Risulta $c = 1$ e $d = 1$. La stima del condizionamento della matrice d'interpolazione A relativa alle funzioni con *shift* in Matlab è $cond(A) = 1.3952e + 04$.



Condizionamento matrice di Vandermonde con nodi equispaziati in $[a, b]$, $t_i = a + i \frac{b-a}{n}$, $i = 0, \dots, n$, nodi di Chebyshev in $[a, b]$, $t_i = \frac{1}{2}((a+b) + (b-a) \cos(\frac{(2i+1)\pi}{2n+2}))$, $i = 0, \dots, n$, e con la tecnica *shift-scaling*.

Rappresentazione di Lagrange

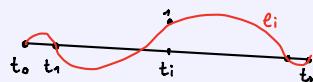
→ il POLINOMIO è sempre lo stesso, lo scrivo solo in maniera diversa

Le funzioni base di Lagrange $\ell_i(t), i = 0, \dots, n$ sono definite da

$$\ell_i(t) = \prod_{j=0, j \neq i}^n \frac{t - t_j}{t_i - t_j}, \quad i = 0, \dots, n$$

e verificano

$$\ell_i(t_j) = \begin{cases} 0 & \text{se } i \neq j \\ 1 & \text{se } i = j \end{cases}$$



Il polinomio interpolante, **polinomio di Lagrange**, si scrive

$$p_n(t) = \sum_{i=0}^n \ell_i(t) y_i.$$

Tale rappresentazione risulta utile nello studio del condizionamento, nell'analisi della convergenza e nelle applicazioni per la derivazione numerica con tecniche pseudospettrali e nell'integrazione numerica.

La valutazione efficiente del polinomio in un punto si basa sulla formulazione **di Lagrange baricentrica**.

Prima rappresentazione di Lagrange baricentrica

Introducendo il polinomio $\ell(t) = \prod_{j=0}^n t - t_j$, si può riscrivere il polinomio di Lagrange come segue

$$p_n(t) = \sum_{i=0}^n \ell_i(t) y_i \Rightarrow p_n(t) = \ell(t) \sum_{i=0}^n \frac{w_i}{t - t_i} y_i$$

calcolo più oneroso

dove $w_i = \frac{1}{\prod_{j=0, j \neq i}^n t_i - t_j} = \frac{1}{\ell'_i(t_j)}$, $i = 0, \dots, n$, rappresentano i **pesi** della formula.

Questa nuova rappresentazione richiede $O(n^2)$ operazioni per calcolare le quantità w_i che non dipendono da t , seguite da $O(n)$ operazioni per la valutazione di $p_n(t)$.

L'introduzione di un nuovo nodo di interpolazione t_{n+1} richiede

- la divisione di ogni w_i , $i = 0, \dots, n$, per $t_i - t_{n+1}$,
- il calcolo di w_{n+1} ,

e quindi in totale $O(n)$ operazioni.

$$\sum_{i=0}^m \frac{l_i(t)(t-t_i)}{(t-t_i)} y_i = l(t) \sum_{i=0}^m \frac{w_i \cdot y_i}{t-t_i}$$

Misurata $l_i(t)(t-t_i) = \frac{l(t)}{\prod_{j=0, j \neq i}^m (t_i - t_j)} = w_i l(t)$

$$(t - t_i) \quad i=0..m$$

$$l(t) \in O(m)$$

$$w_i \text{ richiede } O(m^2)$$

Seconda rappresentazione di Lagrange baricentrica

Dall'uguaglianza

$$1 = \ell(t) \sum_{i=0}^n \frac{w_i}{t - t_i},$$

si ottiene la seguente **seconda formula baricentrica**

$$p_n(t) = \frac{\sum_{i=0}^n \frac{w_i}{t - t_i} y_i}{\sum_{i=0}^n \frac{w_i}{t - t_i}},$$

Per alcune scelte dei nodi di interpolazione t_i i pesi w_i possono essere calcolati esplicitamente:

- $w_i = 2^n(b-a)^{-n}(-1)^i \binom{n}{i}$ per i nodi equidistanti,
- $w_i = 2^n(b-a)^{-n}(-1)^i \sin\left(\frac{(2i+1)\pi}{2n+2}\right)$ per i nodi di Chebyshev,
- $w_i = 2^n(b-a)^{-n}(-1)^i \begin{cases} 1/2 & i = 0, i = n \\ 1 & \text{altrimenti} \end{cases}$ per i nodi estremali di Chebyshev.

Rappresentazione di Newton

Le funzioni base di Newton sono definite da

$$\phi_0(t) = 1, \quad \phi_i(t) = \prod_{j=0}^{i-1} (t - t_j), \quad i = 1, \dots, n.$$

La matrice del sistema risulta triangolare inferiore ed i coefficienti x_i del polinomio interpolante, **polinomio di Newton**,

$$p_n(t) = x_0 + x_1(t - t_0) + x_2(t - t_0)(t - t_1) + \dots + x_n(t - t_0) \cdots (t - t_{n-1})$$

possono essere calcolati con l'algoritmo di sostituzione in avanti con $O(n^2)$ operazioni.

Il polinomio non dipende dall'ordinamento dei nodi di interpolazione ma il condizionamento della matrice risente dall'ordine dei nodi.

Spesso l'ordinamento naturale crescente non risulta il migliore e il condizionamento migliora ordinando i punti in funzione della distanza dal punto medio oppure dal punto t in cui si vuole valutare il polinomio.

RAPPRESENTAZIONE DI NEWTON

$$\phi_0(t) = 1$$

$$\phi_1(t) = t - t_0$$

⋮

$$\phi_m(t) = (t - t_0) \dots (t - t_{m-1})$$

$$\begin{pmatrix} 1 & 0 & & & \\ \vdots & (t_1 - t_0) & & & \\ \vdots & & \ddots & & \\ 1 & (t_n - t_0) & \cdots & (t_n - t_{m-1}) & \end{pmatrix} \begin{pmatrix} x_0 \\ \vdots \\ x_m \end{pmatrix} = \begin{pmatrix} y_0 \\ \vdots \\ y_m \end{pmatrix}$$

Lo risolvo con $O(m^2)$ operaz. perché è TRIANGOLARE

il polinomio interpolante $p_m(t) = x_0 + x_1(t - t_0) + \dots + x_m(t - t_{m-1}) \dots (t - t_0) \rightarrow$ che è quasi nella forma Horner

$$= (x_m(t - t_{m-1}) + x_{m-1})(t - t_{m-2}) + x_{m-2}) \dots (t - t_0) + x_0$$

$$p = x_m$$

for $k = m-1 \dots 0$

$$p = p \cdot (t - t_k) + x_k$$

end for

Polinomio di Newton, continua

Il polinomio di Newton presenta notevoli vantaggi computazionali

- la valutazione del polinomio di Newton in uno o più punti può essere ottenuta mediante una variante dell'algoritmo di Horner;
- l'inserimento di un ulteriore nodo di interpolazione t_{n+1} richiede il calcolo del solo coefficiente x_{n+1} del nuovo polinomio

$$p_{n+1}(t) = p_n(t) + x_{n+1}(t - t_0) \cdots (t - t_n)$$

che si ottiene dalla condizione di interpolazione

$$x_{n+1} = \frac{y_{n+1} - p_n(t_{n+1})}{(t_{n+1} - t_0) \cdots (t_{n+1} - t_n)}$$

Per ottenere i coefficienti del polinomio di Newton c'è un altro approccio che utilizza le **differenze divise**.

Differenze divise e polinomio di Newton

Approccio alternativo per calcolare i COEFFICIENTI PRINCIPALI, calcolando appunto le differenze divise.

Assegnati i dati (t_i, y_i) , $i = 0, \dots, n$, la **differenza divisa** $f[t_i, \dots, t_{i+k}]$ di ordine $k \geq 0$ viene definita ricorsivamente come segue

$$f[t_i] = y_i \quad \text{per } k = 0, i = 0, \dots, n$$

$$f[t_i, \dots, t_{i+k}] = \frac{f[t_{i+1}, \dots, t_{i+k}] - f[t_i, \dots, t_{i+k-1}]}{(t_{i+k} - t_i)}, \quad k > 0, i = 0, \dots, n - k.$$

La differenza divisa di ordine k è invariante rispetto a qualsiasi permutazione degli argomenti.

Si può verificare che per i coefficienti del polinomio interpolante di Newton vale

$$x_i = f[t_0, \dots, t_i], i = 0, \dots, n.$$

Il calcolo delle differenze divise utilizza una tabella e richiede $O(n^2)$ operazioni, ma è meno sensibile a underflow o overflow rispetto al calcolo degli elementi della matrice triangolare.

$$\text{POLINOMIO} \rightarrow p_m(t) = x_m(t-t_0) \dots (t-t_{m-1}) + x_{m-1}(t-t_0) \dots (t-t_{m-2}) + \dots + x_0$$

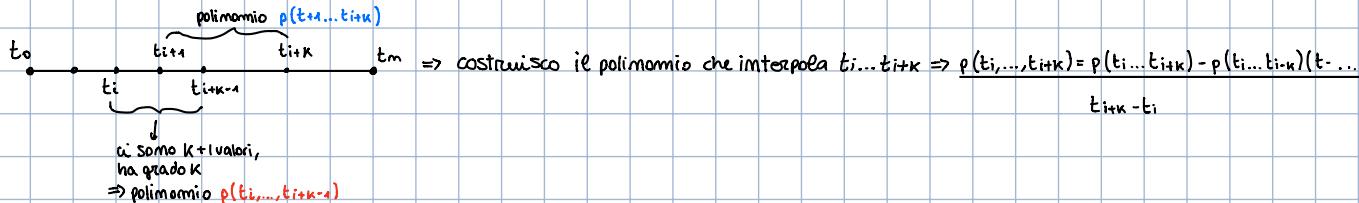
$$= x_m t^m + \dots$$

↓
polinomio di grado m
che interpola i dati t_0, \dots, t_m

↓
polinomio di grado $m-1$
che interpola i dati t_0, \dots, t_{m-1}

x_m è il coefficiente principale del polinomio $p_m(t)$

Consideriamo p .



esempio di calcolo con le rapp. divise:

faccio la tabella:

t_0	y_0	x_0	$f[t_0, t_1]$	x_1	$f[t_0, t_1, t_2]$	x_2	\dots	$f[t_0, t_1, \dots, t_m]$	x_m
t_1	y_1	>	$f[t_0, t_1]$	>	$f[t_0, t_1, t_2]$	>	>	>	>
:	:	:	:	:	:	:	:	:	:
t_m	y_m	>	$f[t_{m-1}, t_m]$	>	$f[t_{m-2}, t_{m-1}, t_m]$	>	>	>	>

e una volta che ottengo $f[t_0, \dots, t_m]$ so che i vari coefficienti sono (red) \rightarrow costa $O(m^2)$

esercizio:

- Calcolare il polinomio che interpola i punti $(0,1), (1,0), (2,3) \Rightarrow t_0=0, y_0=1$ usando tutte le rappresentaz. fatte

$$t_1=1, y_1=0$$

$$t_2=2, y_2=3$$

LAGRANGE: FUNZIONI

$$l_0(t) = \frac{(t-t_1)(t-t_2)}{(t_0-t_1)(t_0-t_2)} = \frac{(t-1)(t-2)}{-1 \cdot -2} = \frac{(t-1)(t-2)}{2}$$

$$l_1(t) = \frac{(t-t_0)(t-t_2)}{(t_1-t_0)(t_1-t_2)} = \frac{t(t-2)}{-1} = -t(t-2)$$

$$l_2(t) = \frac{(t-t_0)(t-t_1)}{(t_2-t_0)(t_2-t_1)} = \frac{t(t-1)}{2}$$

e questi sono i 3 coefficienti, ora p si scrive $p_2(t) = \frac{(t-1)(t-2)}{2} \cdot y_0 - t(t-2) y_1 + \frac{t(t-1)}{2} y_2$
è una parabola (perchè ha 3 var.)

$$\Rightarrow p_2(t) = \frac{(t-1)(t-2)}{2} + \frac{3t(t-1)}{2}$$

ora sostituisco gli y_0, y_1, y_2

NEWTON:

$$(0,1), (1,0), (2,3) \Rightarrow t_0 = 0 \quad y_0 = 1$$

$$t_1 = 1 \quad y_1 = 0$$

$$t_2 = 2 \quad y_2 = 3$$

FORMA $\rightarrow p_2(t) = x_0 + x_1(t-t_0) + x_2(t-t_0)(t-t_1)$ e devo calcolare x_0, x_1, x_2
e sostituire

costruisco la matrice:

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & t_1-t_0 & 0 \\ 1 & t_1-t_0 & t_2-t_0 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 3 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 2 & 2 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 3 \end{pmatrix}$$

uso la sostituz in avanti $\rightarrow x_0 = 1$

$$x_0 + x_1 = 0 \Rightarrow x_1 = -1$$

$$x_0 + 2x_1 + 2x_2 = 3 \Rightarrow x_2 = 2$$

ora sostituisco gli $x_i \Rightarrow p_2(t) = 1 - (t-t_0) + 2(t-t_0)(t-t_1)$ e sostituisco i $t_i \Rightarrow p_2(t) = 1 - t + 2t(t-1)$

e per verificare se è il polinomio interpolante: $p_2(0) = 1 \quad p_2(1) = 0 \quad p_2(3) = 3$

RAPPRESENTAZIONI DIVISE: tabella $t_i \quad y_i$

0	1
1	0
2	3

$$f[t_0, t_1] = \frac{0-1}{1-0} = -1$$
$$f[t_1, t_2] = \frac{1-3}{2-1} = -2$$
$$f[t_0, t_1, t_2] = \frac{0-1}{2-0} = \frac{1}{2} = 0.5$$

e i coeff. sono •

Errore interpolazione polinomiale di una funzione f

→ i dati siamo le campionamenti di una funzione f nei nodi di interpolazione \Rightarrow ora stiamo cercando di **APPROSSIMARE UNA FUNZIONE**

Se $y_i = f(t_i)$, $i = 0, \dots, n$, è importante studiare l'accuratezza dell'approssimazione $p_n(t)$ di $f(t)$ fornita dal polinomio interpolante per $t \in [a, b]$, $a \leq t_0 < t_1 < \dots < t_n \leq b$, diversi dai nodi.

→ f ha le derivate continue fino all'ordine $m+1$
Sia $f \in C^{m+1}[a, b]$ allora

→ c'è un errore, tra due funzioni, è a sua volta una funzione

$$e_n(t) = f(t) - p_n(t) = \frac{f^{n+1}(\xi)}{(n+1)!} (t - t_0) \cdots (t - t_n).$$

Poichè ξ non è noto tale formula non è molto utile in pratica ma ci fornisce comunque utili indicazioni.

→ massimo errore commesso nell'intervallo \Rightarrow è un numero, mentre l'errore è una funzione
Sia $D_{n+1} := \max_{a \leq t \leq b} |f^{n+1}(t)|$. Vale la seguente maggiorazione uniforme dell'errore

$$\begin{aligned} \max_{a \leq t \leq b} |e_n(t)| &\leq \frac{D_{n+1}}{(n+1)!} \max_{a \leq t \leq b} |(t - t_0) \cdots (t - t_n)| \\ &\leq \frac{D_{n+1}}{(n+1)!} |b - a|^{n+1} \end{aligned}$$

Convergenza uniforme

Se $f \in C^\infty[a, b]$ possiamo utilizzare tale formula per analizzare la convergenza per $n \rightarrow \infty$.

Se per ogni $n \geq 0$ risulta $D_{n+1} \leq K^{n+1}$, con K costante positiva, allora l'errore uniforme tende a zero per $n \rightarrow \infty$.

Example

Sia $f(t) = e^t$, $t \in [a, b]$. Si ottiene

$$\max_{a \leq t \leq b} |e_n(t)| \leq \frac{e^b}{(n+1)!} (b-a)^{n+1} \rightarrow 0 \text{ per } n \rightarrow \infty$$

Il polinomio interpolante converge uniformemente alla funzione esponenziale per qualsiasi scelta dei nodi di interpolazione.

Ponendo $a = t_0$ e $b = t_n$ si ottiene la seguente maggiorazione dell'errore

$$\max_{a \leq t \leq b} |e_n(t)| \leq \frac{D_{n+1}}{4(n+1)} h^{n+1},$$

dove $h = \max_{i=0, \dots, n-1} |t_{i+1} - t_i|$.

Errore e differenze divise

Utilizzando le differenze divise si ottiene la seguente espressione dell'errore

$$e_n(t) = f[t_0, \dots, t_n, t](t - t_0) \cdots (t - t_n).$$

e se $f \in C^{n+1}[a, b]$ vale

$$f[t_0, \dots, t_n, t] = \frac{f^{n+1}(\xi)}{(n+1)!}.$$

Ciò permette di estendere la definizione delle differenze divise di ordine k anche con nodi coincidenti come segue

$$f[t, \dots, t] = \frac{f^k(t)}{(k)!}, k \geq 0$$

\uparrow
 $k+1$ volte

Utilizzando la tabella delle differenze divise si possono facilmente costruire polinomi interpolanti con condizioni anche sulle derivate.

es. consideriamo il polinomio che interpola la funzione nei punti t_0, \dots, t_m, t

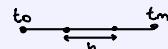
vae. qualsiasi

$$p_m(s; t_0, \dots, t_m) + f[t_0, \dots, t_m, t] (s-t_0) \dots (s-t_m)$$

\uparrow
nuova
variabile

Nodi equidistanti

i modi dove interpolare la funzione li sceglio io



Consideriamo i **nodi equidistanti** $t_i = a + ih$, $i = 0, \dots, n$, $h = \frac{b-a}{n}$, allora

l'ampiezza tra intervalli

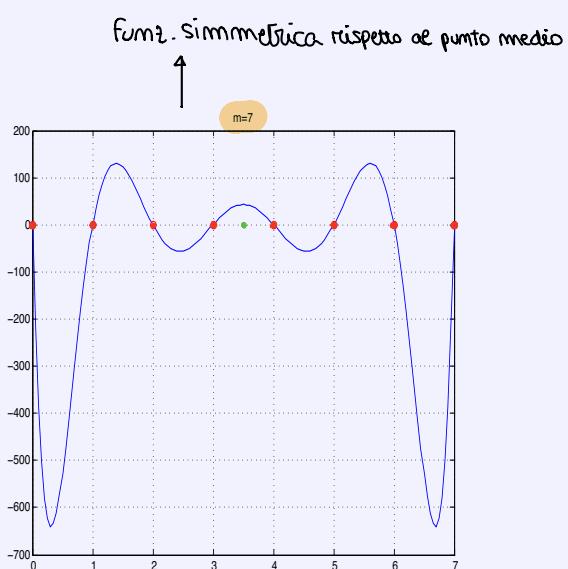
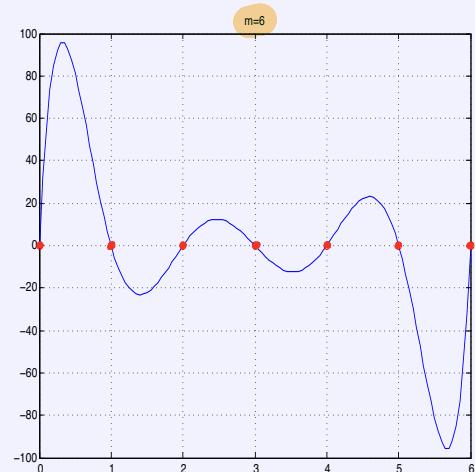
$$e_n(t_0 + sh) = \frac{f^{n+1}(\xi)}{(n+1)!} s(s-1) \cdots (s-n) h^{n+1}, \quad s \in [0, n].$$

Se f^{n+1} non varia molto nell'intervallo $[a, b]$, l'errore si comporterà come $\pi_n(s) = s(s-1) \cdots (s-n)$, $s \in [0, n]$.

- $\pi_n(s + \frac{n}{2}) = (-1)^{n+1} \pi_n(\frac{n}{2} - s)$, $s \in [0, \frac{n}{2}]$
- in ogni intervallo $(i, i+1)$, $i = 0, \dots, n$ c'è un punto di massimo e/o minimo il cui valore in modulo cresce simmetricamente allontanandosi dal centro dell'intervallo verso gli estremi dell'intervallo

Fissato n l'approssimazione sarà migliore nella parte centrale dell'intervallo. Inoltre le oscillazioni di π_n saranno sempre più ampie al crescere di n

Nodi equidistanti, continua



Man mano che ci allontaniamo dal punto centrale le oscillazioni aumentano
 ⇒ la ricostruzione è più accurata nella parte centrale dell'intervallo

La funzione di Runge

Il polinomio interpolante su nodi equidistanti può non convergere al crescere di m , come succede per la funzione di Runge

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

C'è una **scelta ottimale** dei nodi di interpolazione: i **nodi di Chebyshev** relativi all'intervallo $[-5, 5]$

$$t_i = -5 \cos\left(\frac{(2i+1)\pi}{(2n+2)}\right), \quad i = 0, \dots, n.$$

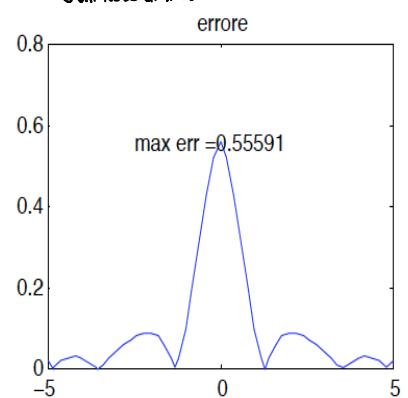
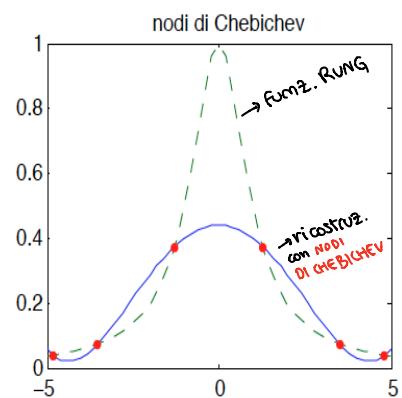
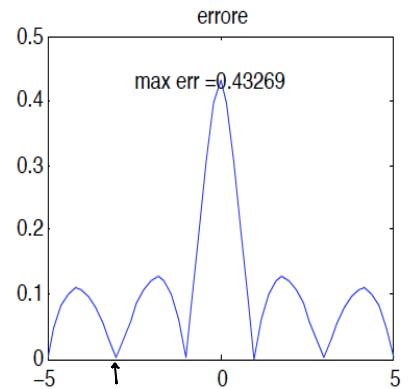
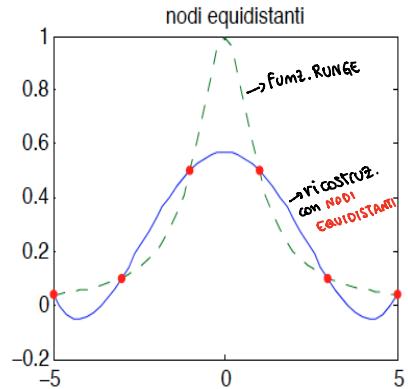
ci assicuriamo la
convergenza

Non sono equispaziati e si "addensano" agli estremi dell'intervallo $[-5, 5]$.

Con tale scelta dei nodi il polinomio interpolante converge uniformemente alla funzione di Runge.

La funzione di Runge e l'errore per $n = 5$

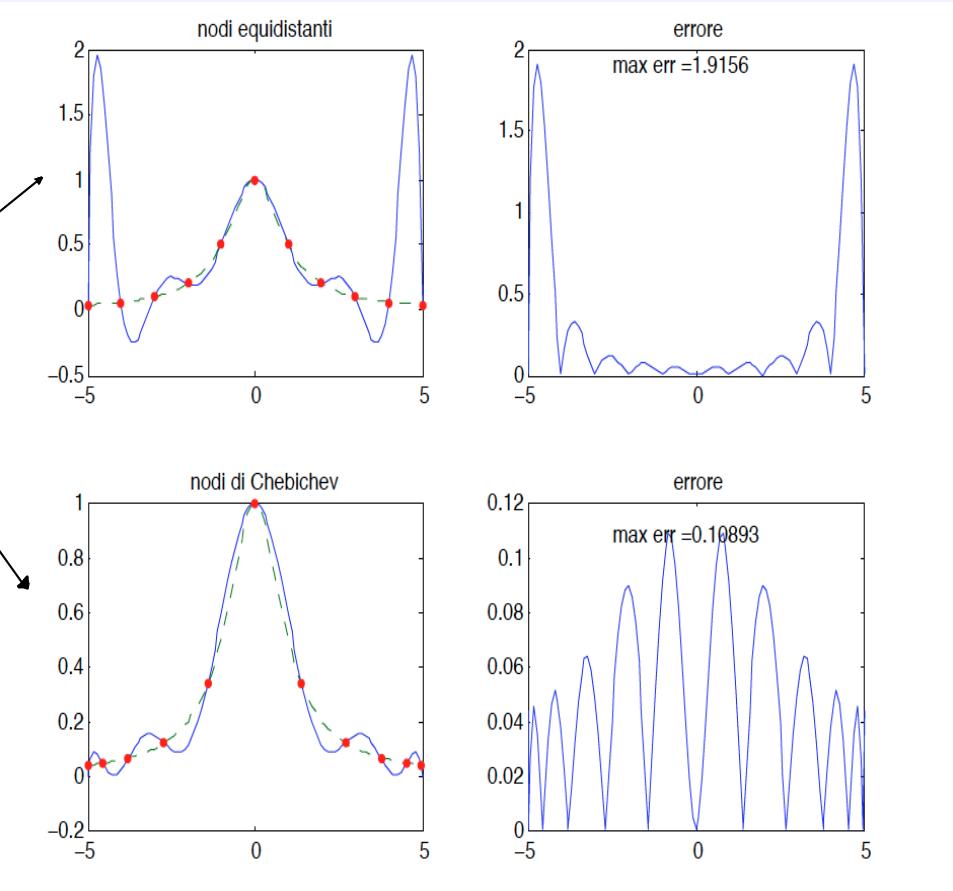
esempio con 5 nodi



La funzione di Runge e l'errore per $n = 10$

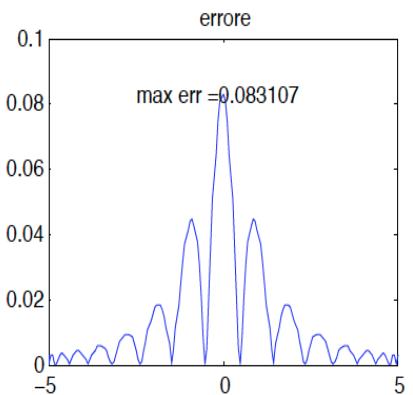
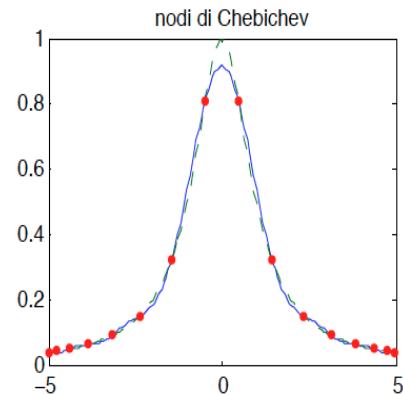
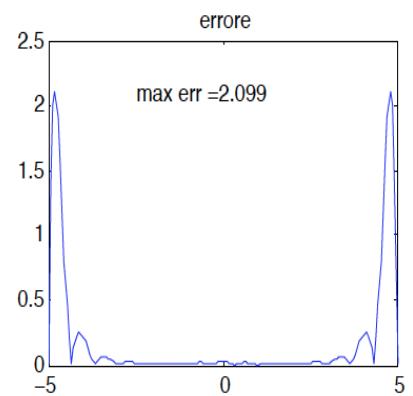
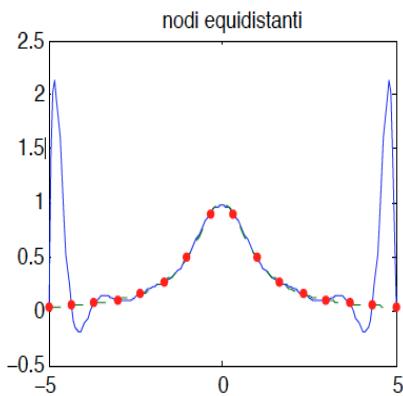
esempio con 10 nodi

scale diverse



La funzione di Runge e l'errore per $n = 15$

esempio 15 modi



Convergenza conclusioni

Per la convergenza del polinomio interpolante si deve avere:

- f continua
- f ha le derivate continue

- Se $f \in C[a, b]$ non è garantita la convergenza del polinomio interpolante nemmeno sui nodi di Chebyshev.
 $\Rightarrow f$ è solo continua
- Per ogni $f \in C^1[a, b]$, è garantita la convergenza del polinomio interpolante sui nodi di Chebyshev.
 \Rightarrow ha le derivate continue

Condizionamento

Siano $p_n(t)$ e $\tilde{p}_n(t)$ i polinomi che interpolano i dati (t_i, y_i) e (t_i, \tilde{y}_i) , $i = 0, \dots, n$ rispettivamente.

↑ interpola i dati
↑ NON perturbati
↗ interpola i dati
↗ PERTURBATI

Usando la rappresentazione di Lagrange dei polinomi si ottiene

$$\max_{t \in [a, b]} |p_n(t) - \tilde{p}_n(t)| \leq \max_{i=0, \dots, n} |y_i - \tilde{y}_i| \Lambda_n(t_0, \dots, t_n)$$

dove

$$\Lambda_n(t_0, \dots, t_n) := \max_{t \in [a, b]} \sum_{i=0}^n |\ell_i(t)|$$

è la **costante di Lebesgue** e misura il condizionamento del problema dell'interpolazione polinomiale.

$$p_m(t) = \sum_{i=0}^m l_i(t) g_i$$

$$\tilde{p}_m(t) = \sum_{i=0}^m l_i(t) \cdot \tilde{g}_i$$

$$\begin{aligned} |p_m(t) - \tilde{p}_m(t)| &= \left| \sum_{i=0}^m l_i(t) (g_i - \tilde{g}_i) \right| \\ &\leq \sum_{i=0}^m |l_i(t)| \cdot \underbrace{|g_i - \tilde{g}_i|}_{\leq \max_{i=0 \dots m} |g_i - \tilde{g}_i|} & \leq \max_{i=0 \dots m} |g_i - \tilde{g}_i| \cdot \sum_{i=0}^m |l_i(t)| \end{aligned}$$

Condizionamento, continua

La costante di Lebesgue dipende dalla scelta dei nodi e risulta

- nodi equidistanti, i.e. $t_i = a + i \frac{b-a}{n}, i = 0, \dots, n, \rightsquigarrow$

$$\Lambda_n(t_0, \dots, t_n) \approx \frac{2^{n+1}}{e \times n \times \log(n)}$$

da cui

$$\Lambda_n(t_0, \dots, t_n) \leq \begin{cases} 6.5 \times 10^3 & n \leq 20 \\ 10^{27} & n \leq 100 \end{cases}$$

- nodi di Chebyshev, i.e. $t_i = \frac{a+b}{2} + \frac{b-a}{2} \cos\left(\frac{(2i+1)\pi}{2n+2}\right), i = 0, \dots, n, \rightsquigarrow$

$$\Lambda_n(t_0, \dots, t_n) \begin{cases} \leq 3 & n \leq 20 \\ \leq 4 & n \leq 100 \\ \approx \frac{2\log(n)}{\pi} & n \text{ grandi} \end{cases}$$

I nodi di Chebyshev rappresentano la scelta ottimale anche per il condizionamento.

Polinomio di Hermite

Nell'interpolazione di **Hermite** vengono assegnati nei nodi $t_i, i = 0, \dots, n$ non solo i valori y_i ma anche dei valori per le derivate dy_i . Il polinomio interpolante di Hermite, H_{2n+1} , verifica

$$\begin{aligned} H_{2n+1}(t_i) &= y_i, \quad i = 0, \dots, n \\ H'_{2n+1}(t_i) &= dy_i, \quad i = 0, \dots, n \end{aligned}.$$

Tale polinomio è unico.

Di tale polinomio si può fornire una rappresentazione "tipo" Lagrange

$$H_{2n+1}(t) = \sum_{i=0}^n (h_i(t)y_i + \bar{h}_i(t)dy_i)$$

dove

$$\begin{aligned} h_i(t) &= \ell_i^2(t)(1 - 2\ell'_i(t_i)(t - t_i)), \quad i = 0, \dots, n \\ \bar{h}_i(t) &= \ell_i^2(t)(t - t_i), \quad i = 0, \dots, n \end{aligned}$$

Polinomio di Hermite e differenze divise

Utilizzando le differenze divise

$$f[t_i] = y_i, f[t_i, t_i] = dy_i, i = 0, \dots, n$$

si può facilmente ottenere una rappresentazione del polinomio di Hermite "tipo" Newton.

Consideriamo, per esempio, il polinomio **cubico di Hermite**, H_3 che soddisfa alle seguenti condizioni di interpolazione:

$$H_3(t_0) = y_0, \quad H_3(t_1) = y_1, \quad H'_3(t_0) = dy_0, \quad H'_3(t_1) = dy_1.$$

Si rappresenta nella forma di Newton come segue:

$$H_3(t) = f[t_0] + f[t_0, t_0](t - t_0) + f[t_0, t_0, t_1](t - t_0)^2 + f[t_0, t_0, t_1, t_1](t - t_0)^2(t - t_1).$$

Polinomio di Hermite ed errore

Se i dati sono il campionamento di una funzione $f \in C^{2n+2}[a, b]$ e $dy_i = f'(t_i)$, $i = 0, \dots, n$ si ha la seguente stima dell'errore puntuale

$$E_{2n+1}(t) = f(t) - H_{2n+1}(t) = \frac{f^{2n+2}(\xi)}{(2n+2)!} ((t - t_0) \cdots (t - t_n))^2$$

Dalla stima puntuale si ricava la stima uniforme dell'errore

$$\max_{a \leq t_0 \leq t \leq t_n \leq b} |E_{2n+1}(t)| \leq \frac{D_{2n+2}}{(2n+2)!} \max_{a \leq t \leq b} ((t - t_0) \cdots (t - t_n))^2$$

Se $a = t_0$ e $b = t_n$ si ha infine

$$\max_{a=t_0 \leq t \leq t_n=b} |E_{2n+1}(t)| \leq \frac{D_{2n+2}(n!)^2}{16(2n+2)!} h^{2n+2},$$

dove $h = \max_{i=0, \dots, n-1} |t_{i+1} - t_i|$.

Polinomi a tratti

L'interpolazione polinomiale a tratti rappresenta una valida alternativa pratica e teorica alle difficoltà che sorgono nell' interpolazione con polinomi di grado m elevato. → invece che aumentarne il grado, si usa un polinomio di grado basso, riducendo l'intervallo e dividendo l'intervallo iniziale in m intervallini.

Data una suddivisione (anche uniforme!)

$$\Delta_m = \{a = t_0 < t_1 < \cdots < t_m = b\}$$

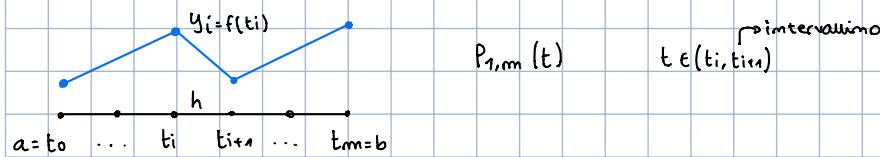
dell'intervallo $[a, b]$, un **polinomio a tratti di grado n** , $P_{n,m}$, è una funzione continua tale che

$$P_{n,m} |_{I_i} \in \Pi_n$$

dove $I_i := [t_i, t_{i+1}]$, $i = 0, \dots, m - 1$.

Nell'interpolazione polinomiale a tratti si usano polinomi di grado n basso raccordati con continuità.

$P_{n,m}$ è descritto da $m(n + 1) - (m - 1) = mn + 1$ parametri.



$$f(t) - P_{1,m}(t) = \frac{|F^{(2)}(\xi_t)| |(t - t_i)(t - t_{i+1})|}{2} \leq \frac{(t_{i+1} - t_i)}{8} \cdot \max_{t \in [a,b]} |f^{(2)}(t)| \leq h^2 \frac{D_2}{8} \xrightarrow{h \rightarrow 0} 0$$

Interpolazione lineare a tratti

Assegnati $(t_i, y_i), i = 0, \dots, m$, il polinomio **lineare** a tratti, si costruisce raccordando le rette che interpolano $(t_i, y_i), (t_{i+1}, y_{i+1})$ in ogni intervallino $I_i, i = 0, \dots, m - 1$.

Se i dati sono il campionamento di una funzione $f \in C^2[a, b]$ si ottiene la seguente stima uniforme dell'errore

$$\max_{a \leq t \leq b} |E_{1,m}(t)| \leq \frac{D_2}{8} h^2$$

dove $E_{1,m} = f - P_{1,m}$ e $h := \max_{i=0, \dots, m-1} |t_{i+1} - t_i|$, che ci assicura la **convergenza uniforme** per $h \rightarrow 0$ di ordine **2**.

Abbiamo sacrificato la "liscezza" dell'interpolante ma abbiamo ottenuto la convergenza per qualsiasi distribuzione dei nodi di interpolazione!

Interpolazione quadratica a tratti

Esercizio

Definisci il polinomio a tratti $P_{2,m}$ interpolante ed analizza la convergenza.

Interpolazione cubica di Hermite a tratti

Per migliorare la "liscezza" dell'interpolante possiamo utilizzare il polinomio cubico di Hermite a tratti, $H_{3,m}$, che risulta una funzione $C^1[a, b]$, imponendo nei punti di raccordo non solo la continuità ma anche la continuità della derivata.

I parametri che descrivono $H_{3,m}$ sono $4m - 2(m - 1) = 2m + 2$ e pertanto risulta univocamente determinato dalle condizioni di interpolazione $H_{3,m}(t_i) = y_i, H'_{3,m}(t_i) = dy_i, i = 0, \dots, m$.

Se i dati sono il campionamento di una funzione $f \in C^4[a, b]$ e $dy_i = f'(t_i), i = 0, \dots, m$, si ottiene la seguente stima uniforme dell'errore

$$\max_{a=t_0 \leq t \leq t_m=b} |f(t) - H_{3,m}(t)| \leq \frac{D_4}{16(4!)} h^4,$$

dove $h =: \max_{i=0, \dots, m-1} |t_{i+1} - t_i|$, che ci assicura la **convergenza uniforme** per $h \rightarrow 0$ di ordine 4.

Interpolazione cubica di Hermite a tratti, continua

Se non si conoscono i valori $f'(t_i), i = 0, \dots, m$ si possono costruire delle stime, valutando per esempio (C. Moler)

per $i = 0, \dots, m - 1$:

$$r_i = \frac{y_{i+1} - y_i}{h_i};$$

per $i = 1, \dots, m - 1$:

$dy_i = 0$ se r_i, r_{i-1} hanno segno opposto o sono nulli, altrimenti

$$w_{i,1} = 2h_i + h_{i-1}, \quad w_{i,2} = h_i + 2h_{i-1},$$

$$dy_i = \frac{w_{i,1} + w_{i,2}}{\frac{w_{i,1}}{r_{i-1}} + \frac{w_{i,2}}{r_i}};$$

$$dy_0 = \frac{(2h_0 + h_1)r_0 - h_0 r_1}{h_0 + h_1},$$

$dy_0 = 0$ se d_0, r_0 hanno segno opposto;

$$dy_m = \frac{(2h_{m-1} + h_{m-2})r_{m-1} - h_{m-1}r_{m-2}}{h_{m-1} + h_{m-2}},$$

$dy_m = 0$ se d_m, r_{m-1} hanno segno opposto.

Funzioni spline

Si definisce **spline di grado n** relativa alla suddivisione Δ_m la funzione $S_{n,m} : [a, b] \rightarrow \mathbb{R}$ tale che

i) $S_{n,m} \in \overset{\text{continua fino all'ordine } n-1}{C^{n-1}}([a, b])$

ii) $S_{n,m}|_{I_i} \in \prod_n, i = 0, \dots, m - 1.$

Una spline $S_{n,m}$ è quindi una funzione polinomiale a tratti di grado n continua con le sue derivate su $[a, b]$ fino all'ordine $n - 1$.

Lo spazio delle funzioni splines $S_{n,m}$ ha dimensione $m + n$, ciò significa che la funzione $S_{n,m}$ è descritta da $m + n$ parametri.

Oss Il polinomio lineare a tratti è la spline $S_{1,m}$

Interpolazione spline cubiche

Consideriamo le **splines cubiche**, che indichiamo per brevità S_3 , in quanto particolarmente interessanti per le applicazioni. Le splines cubiche S_3 sono polinomi a tratti di grado $n = 3$ tali che $S_3 \in C^2([a, b])$.

po derivate 1 e 2 continue

Lo spazio ha dimensione $m+3$ e pertanto sono necessarie $m+3$ condizioni indipendenti per determinare univocamente una funzione spline.

Scegliendo come nodi di interpolazione gli $m+1$ punti della suddivisione Δ_m si hanno $m+1$ condizioni di interpolazione che non sono sufficienti a determinare univocamente la funzione S_3 interpolante.

Interpolazione spline cubiche, continua

Le due ulteriori condizioni da imporre mi permettono di definire delle particolari classi di funzioni splines:

- *naturali*: $S_3''(t_0) = S_3''(t_m) = 0$;
- *periodiche*: $S_3'(t_0) = S_3'(t_m)$ e $S_3''(t_0) = S_3''(t_m)$;
- *vincolate*: $S_3'(t_0) = dy_0$ e $S_3'(t_m) = dy_m$;
- *not-a-knot*: $S_3'''(t)$ continua in t_1 e t_{m-1} .

La scelta della particolare classe dipende dal problema e dalle proprietà che si vogliono garantire all'interpolante.

Spline cubica interpolante : come calcolarla?

$$S_3|_{I_i} \in \Pi_3 \rightsquigarrow S_3''|_{I_i} \in \Pi_1, i = 0, \dots, m-1$$

Definendo i **momenti**

$$S_3''(t_i) = M_i, \quad i = 0, \dots, m$$

otteniamo

$$S_3''(t) = M_{i+1} \frac{t - t_i}{h_i} - M_i \frac{t - t_{i+1}}{h_i}, \quad t \in I_i, \quad i = 0, \dots, m-1$$

dove $h_i = t_{i+1} - t_i$, $i = 0, \dots, m-1$, sono le ampiezze degli intervalli I_i

Interpolazione spline cubiche: come calcolarla?

Integrando

$$S'_3(t) = M_{i+1} \frac{(t - t_i)^2}{2h_i} - M_i \frac{(t - t_{i+1})^2}{2h_i} + \alpha_i, \quad t \in I_i, \quad i = 0, \dots, m-1$$

ed integrando nuovamente

$$S_3(t) = M_{i+1} \frac{(t - t_i)^3}{6h_i} - M_i \frac{(t - t_{i+1})^3}{6h_i} + \alpha_i(t - t_i) + \beta_i, \quad t \in I_i, \quad i = 0, \dots, m-1$$

I parametri $M_i, i = 0, \dots, m$, $\alpha_i, \beta_i, i = 0, \dots, m-1$, si determinano imponendo

- ① le condizioni di interpolazione
- ② le condizioni di continuità sulla derivata prima
- ③ scegliendo una particolare spline.

Condizioni di interpolazione

Imponiamo le $m + 1$ condizioni di interpolazione che ci assicurano anche la continuità di S_3 .

$$\left\{ \begin{array}{l} S_3|_{I_i}(t_i) = y_i \rightsquigarrow M_i \frac{h_i^2}{6} + \beta_i = y_i, \quad i = 0, \dots, m-1 \\ S_3|_{I_i}(t_{i+1}) = y_{i+1} \rightsquigarrow M_{i+1} \frac{h_i^2}{6} + \alpha_i h_i + \beta_i = y_{i+1}, \quad i = 0, \dots, m-1 \end{array} \right.$$

Tali relazioni ci permettono di esprimere α_i e β_i , $i = 0, \dots, m-1$, in funzione dei momenti M_i , $i = 0, \dots, m$, come segue

$$\begin{cases} \alpha_i = \frac{y_{i+1} - y_i}{h_i} - (M_{i+1} - M_i) \frac{h_i}{6} \\ \beta_i = y_i - M_i \frac{h_i^2}{6} \end{cases}$$

Condizioni di interpolazione, continua

Abbiamo

$$\begin{aligned} S_3(t) &= M_{i+1} \frac{(t-t_i)^3}{6h_i} - M_i \frac{(t-t_{i+1})^3}{6h_i} + \\ &+ \left(\frac{y_{i+1}-y_i}{h_i} - (M_{i+1} - M_i) \frac{h_i}{6} \right) (t - t_i) + \\ &+ y_i - M_i \frac{h_i^2}{6}, \quad t \in I_i, \quad i = 0, \dots, m-1 \end{aligned}$$

e

$$\begin{aligned} S'_3(t) &= M_{i+1} \frac{(t-t_i)^2}{2h_i} - M_i \frac{(t-t_{i+1})^2}{2h_i} + \\ &+ \frac{y_{i+1}-y_i}{h_i} - (M_{i+1} - M_i) \frac{h_i}{6}, \quad t \in I_i, \quad i = 0, \dots, m-1 \end{aligned}$$

Condizioni di continuità della derivata prima

Le $m - 1$ condizioni di continuità della derivata prima

$$S'_3|_{I_i}(t_{i+1}) = S'_3|_{I_{i+1}}(t_{i+1}), \quad i = 0, \dots, m - 2$$

definiscono le equazioni lineari

$$\begin{cases} M_i h_i + 2M_{i+1}(h_i + h_{i+1}) + M_{i+2} h_{i+1} = 6 \left(\frac{y_{i+2} - y_{i+1}}{h_{i+1}} - \frac{y_{i+1} - y_i}{h_i} \right), \\ i = 0, \dots, m - 2 \end{cases}$$

Le due ulteriori equazioni si ottengono considerando le condizioni che definiscono le diverse classi di splines. Dopo aver calcolato i momenti, si determinano le costanti α_i e β_i , $i = 0, \dots, m - 1$ e quindi la spline interpolante è ricostruita.

Splines naturali interpolanti

Per le **splines naturali** vale $M_0 = 0$ e $M_m = 0 \rightsquigarrow$

$$\begin{pmatrix} h_{0,1} & h_1 & 0 & \cdots & 0 \\ h_1 & h_{1,2} & h_2 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & h_{m-3} & h_{m-3,m-2} & h_{m-2} \\ 0 & \cdots & 0 & h_{m-2} & h_{m-2,m-1} \end{pmatrix} \begin{pmatrix} M_1 \\ \vdots \\ M_{m-1} \end{pmatrix} = \begin{pmatrix} Y_{1,0} \\ \vdots \\ Y_{m-1,m-2} \end{pmatrix}$$

con

$$h_{i,j} = 2(h_i + h_j), \quad i, j = 0, \dots, m-1$$

$$Y_{i,j} = 6\left(\frac{y_{i+1}-y_i}{h_i} - \frac{y_{j+1}-y_j}{h_j}\right) \quad i, j = 0, \dots, m, \quad i \neq j = 1$$

La matrice è tridiagonale, simmetrica e definita positiva e il sistema può essere agevolmente risolto con algoritmi dedicati.

Splines periodiche interpolanti

Per le **splines periodiche**

- $S_3''(t_0) = S_3''(t_m) \rightsquigarrow M_0 = M_m$
- $S_3'(t_0) = S_3'(t_m) \rightsquigarrow$

$$2M_0(h_0 + h_{m-1}) + M_1h_0 + M_{m-1}h_{m-1} = 6 \left(\frac{y_1 - y_0}{h_0} - \frac{y_m - y_{m-1}}{h_{m-1}} \right)$$

da cui si ottiene il sistema lineare

$$\begin{pmatrix} h_{0,m-1} & h_0 & & h_{m-1} \\ h_0 & h_{0,1} & h_1 & \\ \ddots & \ddots & \ddots & \\ h_{m-1} & & h_{m-2} & h_{m-2,m-1} \end{pmatrix} \begin{pmatrix} M_0 \\ M_1 \\ \vdots \\ M_{m-1} \end{pmatrix} = \begin{pmatrix} Y_{0,m-1} \\ Y_{1,0} \\ \vdots \\ Y_{m-1,m-2} \end{pmatrix}$$

Splines vincolate interpolanti

Per le **splines vincolate**

- $S'_3(t_0) = dy_0 \rightsquigarrow$

$$2M_0h_0 + M_1h_1 = 6 \left(\frac{y_1 - y_0}{h_0} - dy_0 \right) = \bar{Y}_0$$

- $S'_3(t_m) = dy_m \rightsquigarrow$

$$2M_mh_{m-1} + M_{m-1}h_{m-1} = 6 \left(dy_m - \frac{y_m - y_{m-1}}{h_{m-1}} \right) = \bar{Y}_m$$

da cui si ottiene il sistema lineare tridiagonale

$$\begin{pmatrix} 2h_0 & h_0 & & & & \\ h_0 & h_{0,1} & h_1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & h_{m-2} & h_{m-2,m-1} & h_{m-1} \\ & & & & h_{m-1} & 2h_{m-1} \end{pmatrix} \begin{pmatrix} M_0 \\ M_1 \\ \vdots \\ M_{m-1} \\ M_m \end{pmatrix} = \begin{pmatrix} \bar{Y}_0 \\ Y_{1,0} \\ \vdots \\ Y_{m-1,m-2} \\ \bar{Y}_m \end{pmatrix}$$

Convergenza delle splines interpolanti

Theorem

Sia $f \in C^4([a, b])$ e sia $\Delta_m = \{t_0^m = a < t_1^m < \dots < t_m^m = b\}$ una successione di suddivisioni di $[a, b]$ tali che $\sup_{i,m} \frac{h^m}{|h_i^m|} \leq K < +\infty$, con $h_i^m := |t_i^m - t_{i-1}^m|$ e $h^m := \max_{i=0, \dots, m-1} h_i$. Allora

$$\max_{a \leq t \leq b} |f^{(k)}(t) - S_3^{(k)}(t)| \leq C_k K h^{4-k} D_4, \quad k = 0, 1, 2, 3.$$

con $C_k > 0$ opportune costanti indipendenti da Δ_m .

Proprietà di "minima curvatura" delle splines cubiche interpolanti

Theorem

Tra tutte le funzioni $g \in C^2[a, b]$ che interpolano (t_i, y_i) , $i = 0, \dots, m$, con $a = t_0$, $b = t_m$, le splines cubiche **naturali** sono quelle che minimizzano

$$\int_a^b (g''(t))^2 dt \quad (1)$$

Theorem

Tra tutte le funzioni $g \in C^2[a, b]$ che interpolano (t_i, y_i) , $i = 0, \dots, m$, con $a = t_0$, $b = t_m$, e $g'(t_0) = dy_0$, $g'(t_m) = dy_m$, le splines cubiche **vincolate** sono quelle che minimizzano (1)

Theorem

Tra tutte le funzioni $g \in C^2[a, b]$ periodiche che interpolano (t_i, y_i) , $i = 0, \dots, m$, con $a = t_0$, $b = t_m$, le splines **periodiche** sono quelle che minimizzano (1)

Splines cubiche e polinomi cubici di Hermite a tratti

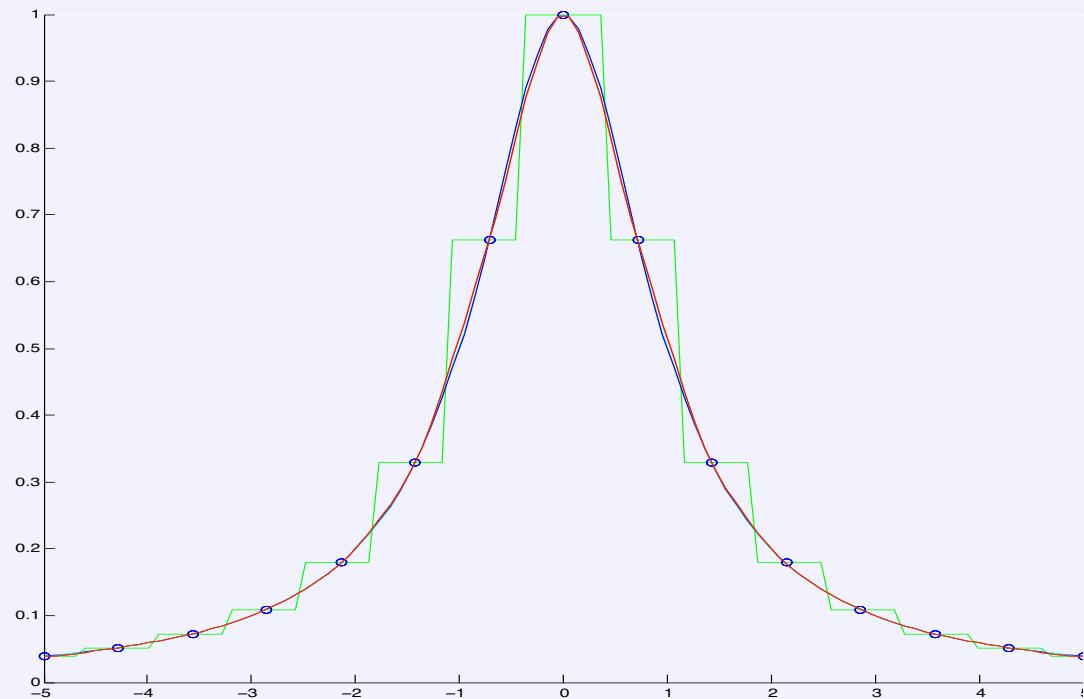
La scelta tra le funzioni spline cubiche ed i polinomi cubici di Hermite a tratti dipende dai dati da interpolare e dagli obiettivi che si vogliono raggiungere con l'interpolazione.

Se la liscezza è importante la scelta delle splines è appropriata.

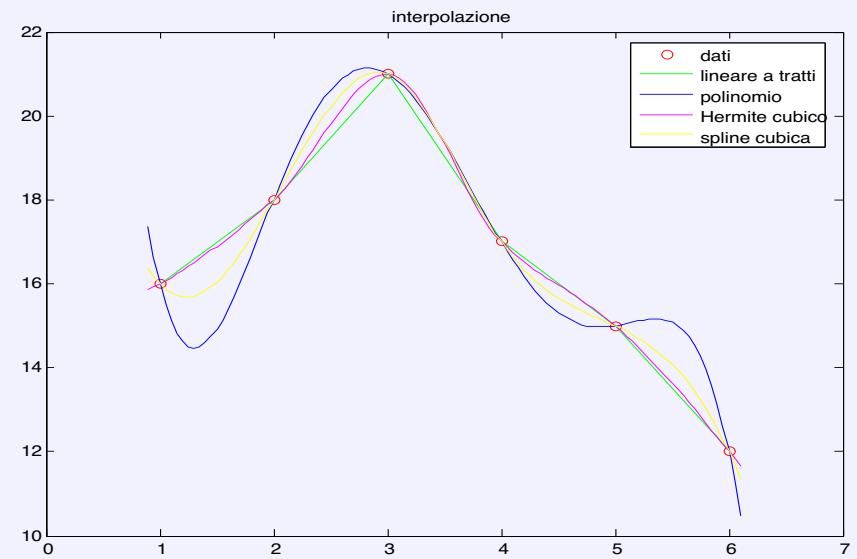
L'interpolante di Hermite a tratti risulta più flessibile nel preservare la monotonia dei dati originali.

Visualizzare i dati ed entrambe le interpolanti può aiutare nella scelta dell'interpolante che meglio descrive il comportamento dei dati originali.

Interpolazione della funzione di Runge



Risultati grafici



Le funzioni B-spline

Le funzioni **B-spline** di grado n forniscono una base per lo spazio delle funzioni spline $S_{m,n}$.

Possono essere definite attraverso la ricorsione, la convoluzione e le differenze divise. Analizziamo la ricorsione.

Estendiamo i nodi assegnati $t_i, i = 0, \dots, m$, scegliendo arbitrariamente degli ulteriori infiniti punti fuori dall'intervallo $[t_0, t_m]$ come segue

$$\dots < t_{-2} < t_{-1} < \mathbf{t_0} < \dots < \mathbf{t_m} < t_{m+1} < t_{m+2} < \dots$$

Definite le **B-splines** di grado $\mathbf{n} = 0$ come segue

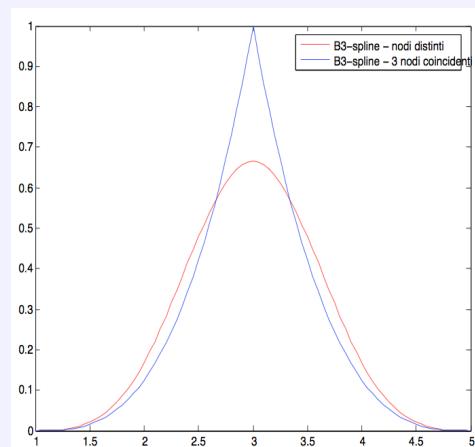
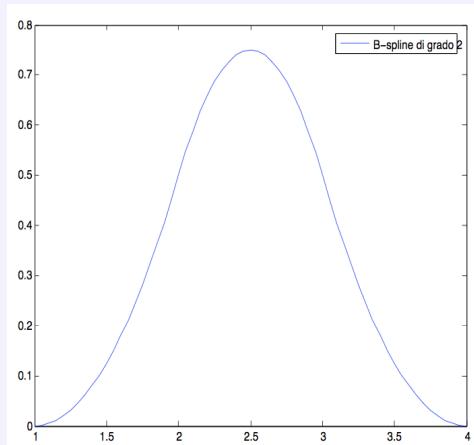
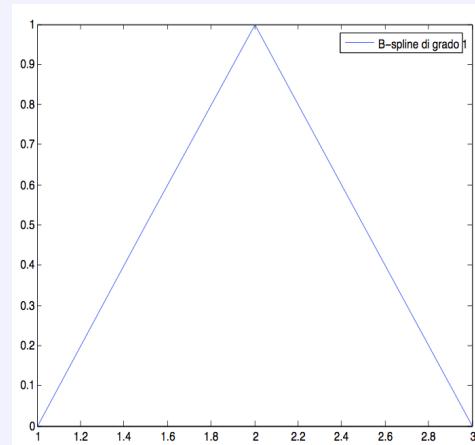
$$B_i^0(t) = \begin{cases} 1 & \text{se } t_i \leq t \leq t_{i+1} \\ 0 & \text{altrove} \end{cases}, \quad i = 0, 1, \dots, m-1,$$

quelle di grado $\mathbf{n} > 0$ sono ottenute ricorsivamente attraverso le relazioni:

$$B_i^n(t) = c_i^n(t)B_i^{n-1}(t) + (1 - c_{i+1}^n(t))B_{i+1}^{n-1}(t),$$

dove $c_i^n(t) = \frac{t - t_i}{t_{i+n} - t_i}$, e $i = -n, \dots, -1, 0, 1, \dots, m-1$.

B-splines, continua



B-splines, continua

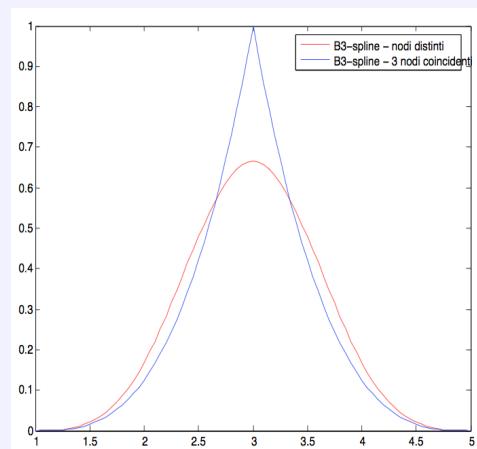
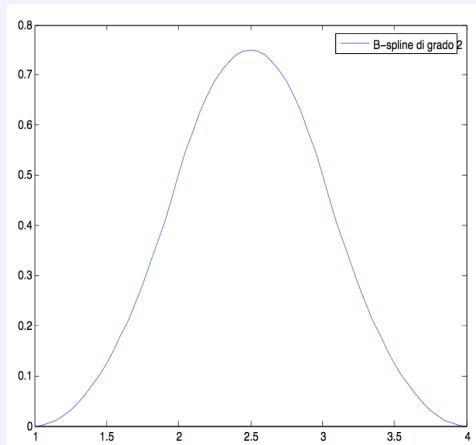
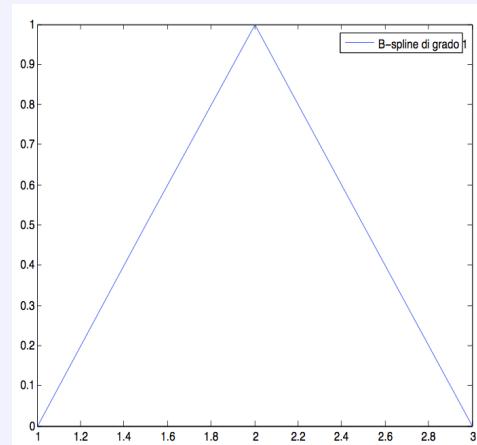
Le funzioni $B_i^n(t)$ soddisfano le seguenti proprietà:

- $B_i^n(t) \geq 0$, $t_i \leq t \leq t_{i+n+1}$ e $B_i^n(t) = 0$, altrove (**supporto compatto**)
- Per $n > 0$ le funzioni B_i^n hanno derivate continue fino all'ordine $n - 1 \rightsquigarrow$ sono splines!
- Le funzioni $B_{-n}^n(t), \dots, B_0^n(t), \dots, B_{m-1}^n$ sono **linearmente indipendenti** in $[t_0, t_m]$ e sono una **base** per lo spazio $S_{n,m}$.
- $\sum_{i=-\infty}^{+\infty} B_i^n(t) = 1, \forall t$ (**proprietà di normalizzazione**).

Usando le funzioni B-spline come base di rappresentazione, il sistema lineare da risolvere per calcolare i coefficienti della funzione spline interpolante risulta non singolare e con una struttura a banda.

In questo modo sia il calcolo che la valutazione delle funzioni spline interpolanti possono essere realizzate per mezzo di metodi efficienti e stabili.

B-splines, continua



Interpolazione parametrica

La funzione interpolante dipende dal sistema di coordinate usato per rappresentare i punti. Nella ricostruzione di curve che passano per dei punti assegnati $P_i = (x_i, y_i)$, $i = 0, \dots, m$ si vorrebbe che la ricostruzione fosse **indipendente** dal sistema di coordinate scelto.

Una soluzione è fornita dall'**interpolazione parametrica**.

- Consideriamo una curva piana in forma parametrica

$$P(t) = (x(t), y(t)), \quad t \in [0, T]$$

- Introduciamo una suddivisione $0 = t_0 < t_1 < \dots < t_m = T$ e consideriamo $P_i = (x_i, y_i) = (x(t_i), y(t_i))$, $i = 0, \dots, m$.
- Con i due insiemi di dati $\{t_i, x_i\}$ e $\{t_i, y_i\}$, $i = 0, \dots, m$ e le relative funzioni interpolanti

$$\begin{aligned} X_m(t) : \quad & X_m(t_i) = x_i, \quad i = 0, \dots, m, \\ Y_m(t) : \quad & Y_m(t_i) = y_i, \quad i = 0, \dots, m, \end{aligned}$$

si costruisce la **curva interpolante parametrica** di grado n

$$P_n(t) = (X_n(t), Y_n(t)), \quad t \in [0, T].$$

Interpolazione parametrica, continua

La partizione di $[0, T]$ può essere realizzata in diversi modi. Per esempio, si può porre $T = 1$ e $t_i = \frac{i}{m}$, $i = 0, \dots, m$. Una ragionevole scelta alternativa è la parametrizzazione basata sulla **lunghezza cumulativa** della linea a tratti che unisce i punti P_i , i.e.

$$t_0 = 0, \quad t_i = \sum_{k=1}^i \ell_k, \quad i = 1, \dots, m,$$

con

$$\ell_i = \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}, \quad i = 1, \dots, m.$$

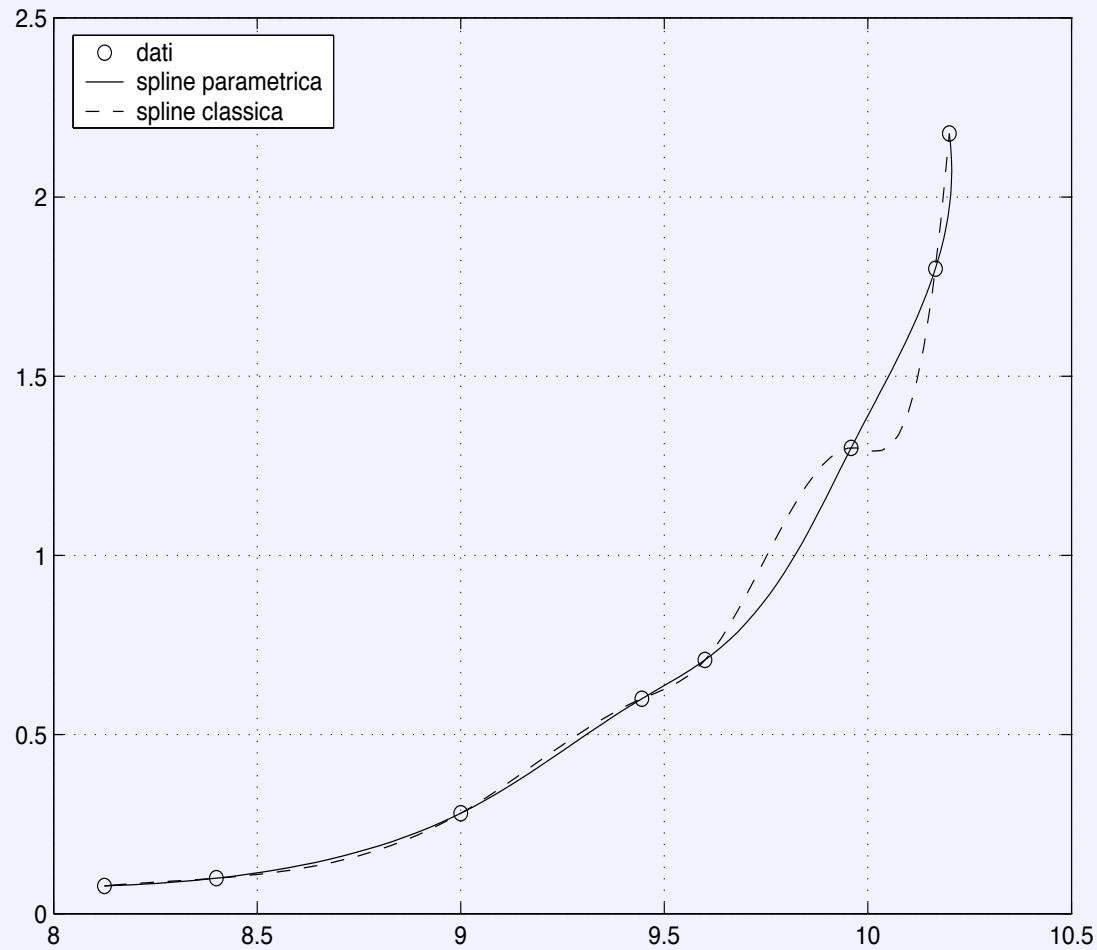
lunghezza del segmento $\overline{P_{i-1}P_i}$.

Le interpolanti parametriche sono geometricamente invarianti, ovvero indipendenti dal sistema di coordinate scelto per la rappresentazione.

La funzione viene chiamata interpolante *cumulative length* e approssima in modo soddisfacente quelle curve che presentano ampia curvatura.

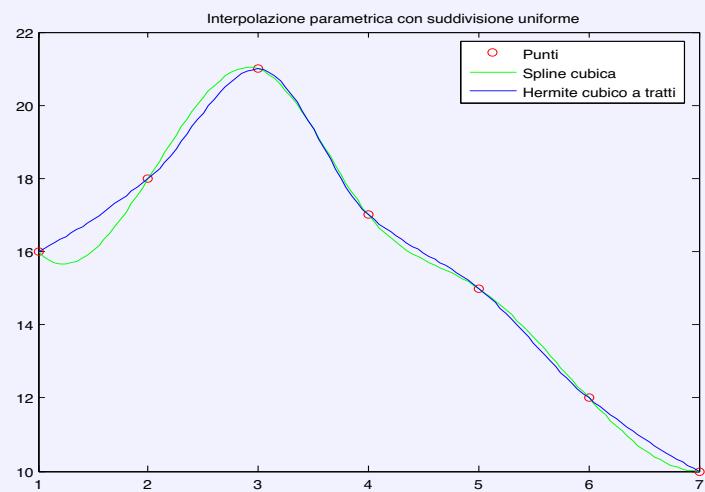
La funzione spline cubica interpolante approssima in modo soddisfacente quelle curve che presentano ampia curvatura.

Spline parametrica



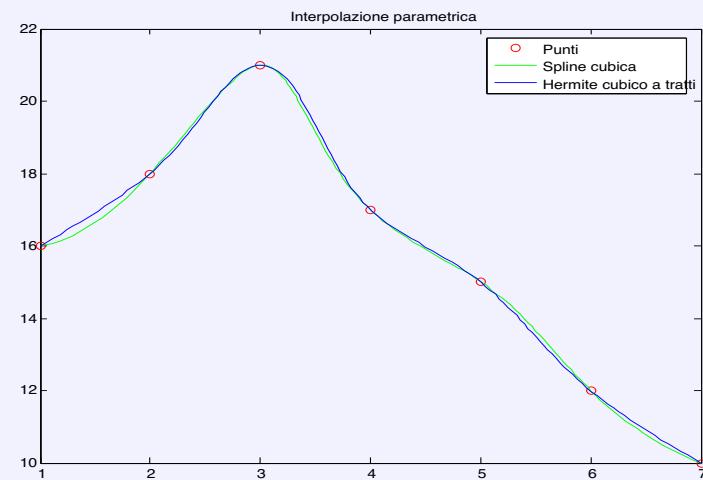
Risultati con suddivisione uniforme

$$x = [1, 2, 3, 4, 5, 6, 7], y = [16, 18, 21, 17, 15, 12, 10],$$



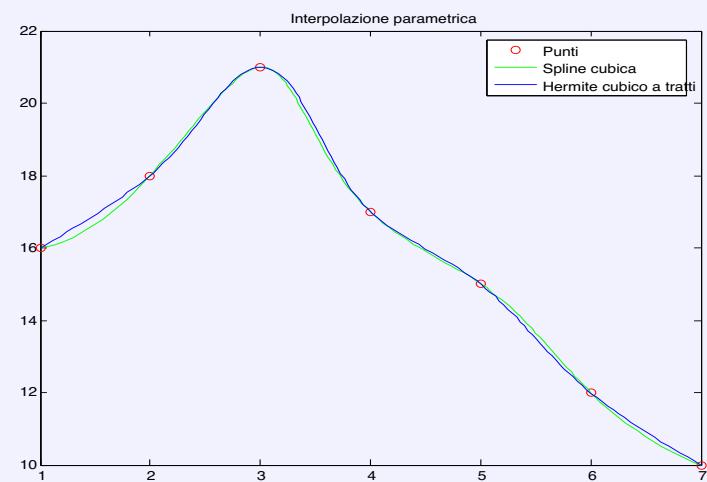
Risultati con lunghezza cumulativa

$$x = [1, 2, 3, 4, 5, 6, 7], y = [16, 18, 21, 17, 15, 12, 10],$$

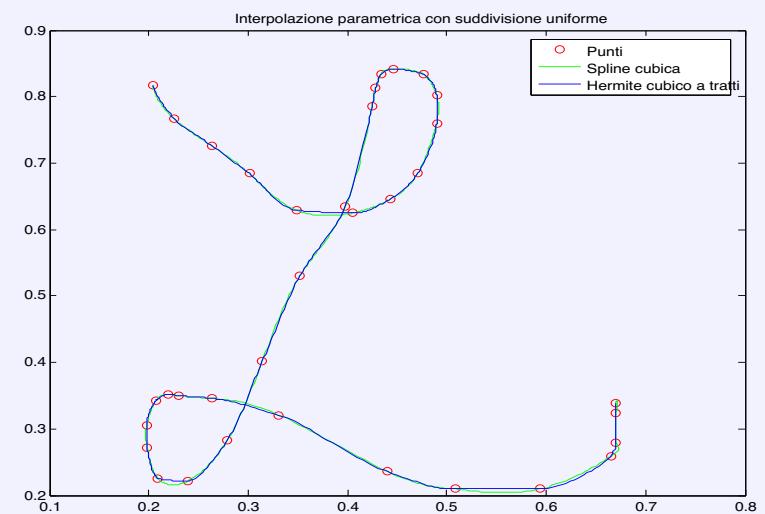


Risultati con lunghezza cumulativa

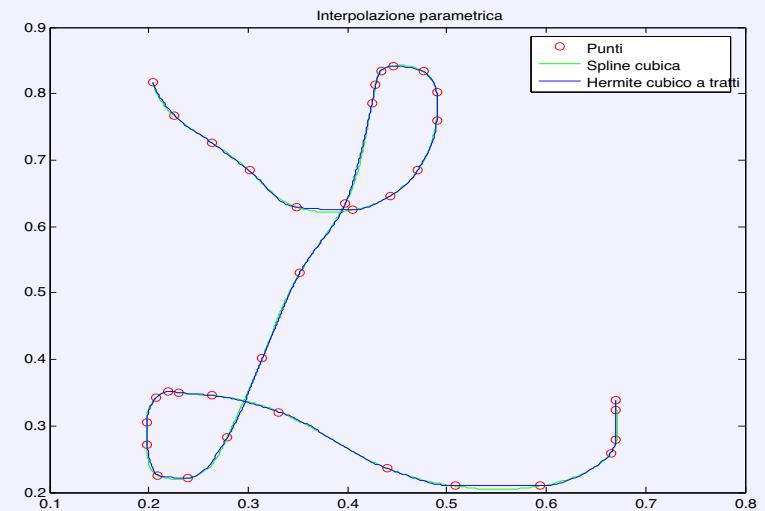
$$x = [1, 2, 3, 4, 5, 6, 7], y = [16, 18, 21, 17, 15, 12, 10],$$



Lettera con suddivisione uniforme



Lettera con lunghezza cumulativa



Curva di Bézier

Siano $P_i, i = 0, \dots, m, m + 1$ punti ordinati nel piano. Il poligono che essi determinano si chiama **poligono di Bézier** o **poligono caratteristico**.

I polinomi di **Bernstein** $b_k^m \in \Pi_m$ nell'intervallo $[0, 1]$ sono definiti da

$$b_k^m(t) = \frac{m!}{k!(m-k)!} t^k (1-t)^{m-k}, \quad k = 0, \dots, m, m = 0, 1, \dots,$$

e possono essere ottenuti attraverso la formula ricorsiva

$$\begin{cases} b_0^m(t) = (1-t)^m \\ b_k^m(t) = (1-t)b_k^{m-1}(t) + tb_{k-1}^{m-1}(t), \quad k = 1, \dots, m. \end{cases}$$

$\{b_k^m | k = 0, \dots, m\}$ rappresenta una base per lo spazio Π_m .

La **curva di Bézier** è definita come segue:

$$B_m(t; P_0, \dots, P_m) = \sum_{k=0}^m b_k^m(t) P_k, \quad t \in [0, 1].$$

Curva di Bézier, continua

Tali curve possono essere ottenute anche attraverso un approccio **geometrico** partendo dal poligono di Bézier.

Fissato $t \in [0, 1]$, i punti

$$P_{i,1}(t) = (1 - t)P_i + tP_{i+1}, i = 0, \dots, m - 1,$$

definiscono m vertici di un nuovo poligono. Possiamo ripetere il procedimento generando dei nuovi vertici

$$P_{i,2}(t) = (1 - t)P_{i,1} + tP_{i+1,1}, i = 0, \dots, m - 2,$$

e il poligono che li unisce e terminando quando il poligono risultante è costituito dai soli due vertici $P_{0,m-1}, P_{1,m-1}$.

Si può verificare che

$$P_{0,m}(t) = (1 - t)P_{0,m-1} + tP_{1,m-1} = B_m(t; P_0, \dots, P_m,)$$

Ripetendo il procedimento per diversi valori di t si trova la curva di Bézier.

Oss: Assegnata una configurazione di nodi si possono costruire diverse curve cambiando l'ordinamento dei punti. $B_m t; (P_0, \dots, P_m, t)$ coincide con $B_m(t; P_m, \dots, P_0)$ a parte l'orientamento.

Curva B-spline

Le curve di Bézier non forniscono una approssimazione sufficientemente accurata del poligono e per questa ragione sono più adeguate le **curve B-spline**.

Sia $n \leq m$. Scegliendo i nodi in $[0, 1]$ come segue

$$t_0 = t_1 = \dots = t_n = 0 < t_{n+1} < \dots < t_m < t_{m+1} = 1 = \dots = t_{m+n+1}$$

$m-n$ punti interni

e costruendo le relative funzioni B-spline di grado n , i.e B_k^n , $k = 0, \dots, m$, otteniamo la **curva B-spline** di grado n

$$S_n(t; P_0, \dots, P_m) = \sum_{k=0}^m B_k^n(t) P_k, \quad t \in [0, 1],$$

passante per i punti P_0 e P_m .

Oss: La scelta dei nodi equivale a chiedere che i primi n vertici del poligono e gli ultimi n coincidano rispettivamente con P_0 e P_m . In generale la curva B-spline di grado n interpola un punto specifico se si impone che n vertici del poligono siano coincidenti, mentre si ottiene localmente una retta prendendo $n+1$ vertici consecutivi allineati.

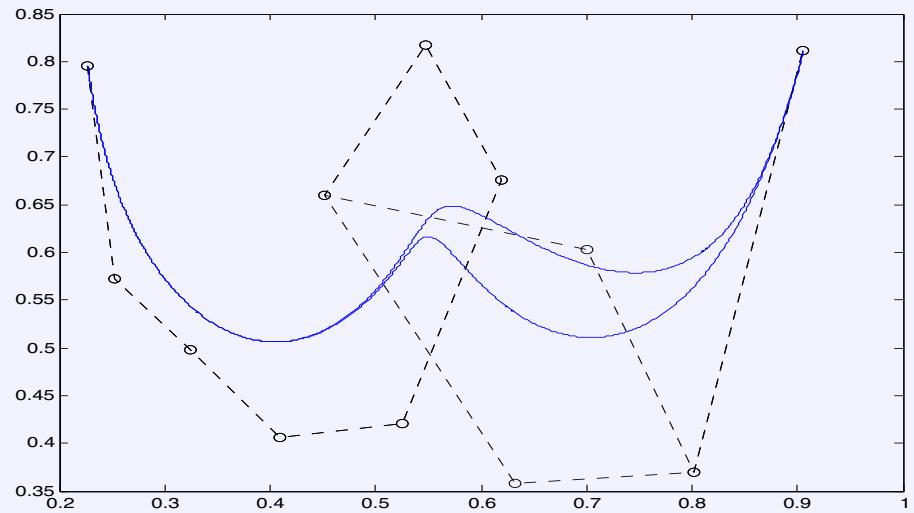
Curva B-spline, continua

Poichè il supporto di B_k^n è $[t_k, t_{k+n+1}]$, la modifica del solo vertice P_k del poligono caratteristico induce una perturbazione "locale" della curva B-spline attorno al vertice.

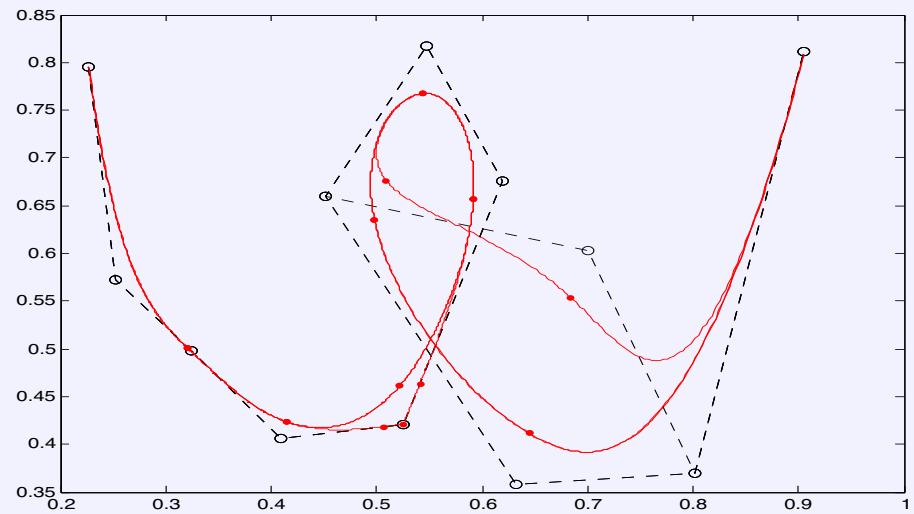
La curva B-spline approssima il poligono di controllo meglio della corrispondente curva di Bézier.

Le curve di Bézier e le curve B-spline trovano importanti applicazioni in diversi ambiti ed, in particolare, nella "computer graphics".

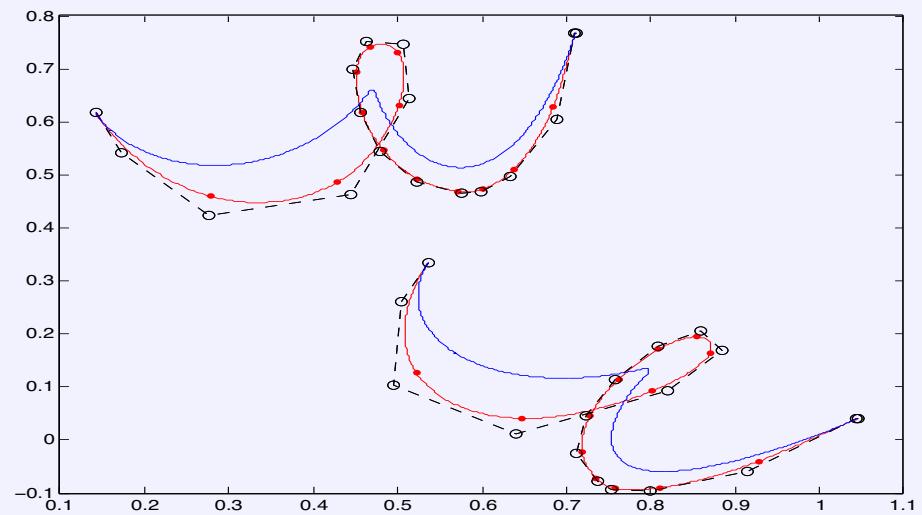
Curva di Bezier: modifica di un punto



Curva B-spline: modifica di un punto



Curva di Bezier e curva B-spline: invarianza per rotazione



Polinomio trigonometrico e interpolazione

Sia $f : [0, 2\pi] \rightarrow \mathbb{R}$ una funzione **periodica**, i.e. $f(0) = f(2\pi)$ e supponiamo dati gli $m + 1 = 2M + 1$ valori $y_j = f(t_j), j = 0, \dots, m$ nei nodi $t_j = \frac{2\pi j}{m+1}, j = 0, \dots, m$.

Si cerca il **polinomio trigonometrico**

$$T_{2M+1}(t) = \frac{a_0}{2} + \sum_{k=1}^M (a_k \cos(kt) + b_k \sin(kt))$$

che soddisfa le condizioni di interpolazione

$$T_{2M+1}(t_j) = y_j, \quad j = 0, \dots, m.$$

Oss Se dati hanno m dispari, i.e. $m = 2M + 1$ si considera un polinomio trigonometrico della seguente forma

$$T_{2M+2}(t) = \frac{a_0}{2} + \sum_{k=1}^M (a_k \cos(kt) + b_k \sin(kt)) + a_{M+1} \cos((M+1)t)$$

Introducendo

$$e^{\imath t} = \cos(t) + \imath \sin(t)$$

con \imath l'unità immaginaria, il polinomio trigonometrico può essere rappresentato

$$T_{2M+1}(t) = \sum_{k=-M}^{M} c_k e^{\imath kt}$$

dove

$$a_k = c_k + c_{-k}, \quad b_k = \imath(c_k - c_{-k}), \quad k = 0, \dots, M$$

e le condizioni di interpolazione nei nodi $t_j = jh$, con $h = \frac{2\pi}{m+1}, j = 0, \dots, m$,

$$y_j = \sum_{k=-M}^{M} c_k e^{\imath k j h}, \quad j = 0, \dots, m$$

Interpolazione trigonometrica, continua

Il sistema lineare

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ e^{-\imath Mh} & \dots & e^{\imath(M-1)h} & e^{\imath Mh} \\ \vdots & \vdots & \vdots & \vdots \\ e^{-\imath M(m-1)h} & \dots & e^{\imath(M-1)(m-1)h} & e^{\imath M(m-1)h} \\ e^{-\imath Mmh} & \dots & e^{\imath(M-1)mh} & e^{\imath Mmh} \end{pmatrix} \begin{pmatrix} c_{-M} \\ c_{-M+1} \\ \vdots \\ c_{M-1} \\ c_M \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{m-1} \\ y_m \end{pmatrix}$$

ha soluzione unica.

FFT

La soluzione può essere ottenuta **direttamente**

$$\sum_{j=0}^m y_j e^{-\imath \ell j h} = \sum_{j=0}^m e^{-\imath \ell j h} \sum_{k=-M}^M c_k e^{\imath k j h}, \quad \ell = -M, \dots, 0, \dots, +M.$$

Poichè risulta

$$\sum_{j=0}^m e^{-\imath (k-\ell) j h} = \begin{cases} 0 & \text{se } k \neq \ell \\ (m+1) & \text{se } k = \ell \end{cases}$$

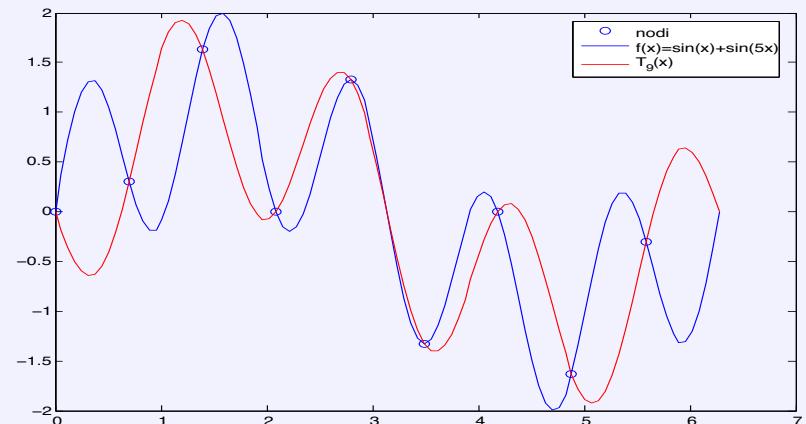
si ottiene (**trasformata discreta di Fourier**)

$$c_\ell = \frac{1}{m+1} \sum_{j=0}^m y_j e^{-\imath \ell j h}, \quad \ell = -M, \dots, 0, \dots, +M.$$

Il calcolo della soluzione richiede un prodotto matrice-vettore con un costo dell'ordine di $(m+1)^2$ flops. Tale costo può essere significativamente ridotto a $(m+1)\log_2(m+1)$ flops se si utilizza la **trasformata rapida di Fourier** o **FFT**. Anche $f = Tc$ (**trasformata discreta di Fourier inversa**) può essere calcolata in maniera efficiente con la versione rapida.

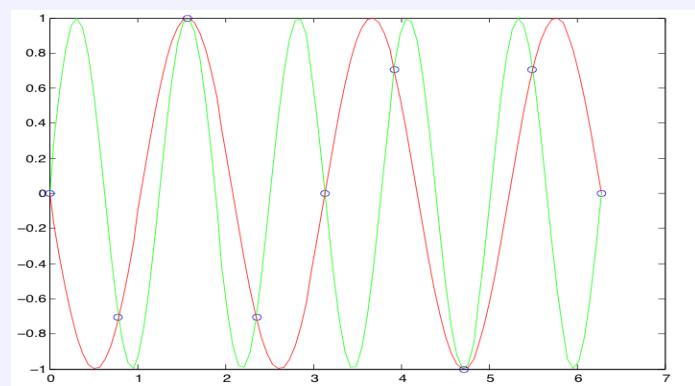
Esempio di "aliasing"

L'interpolazione trigonometrica può essere poco accurata in particolari situazioni. Consideriamo la funzione $f(t) = \sin(t) + \sin(5t)$ e costruiamo il polinomio trigonometrico interpolante con $m = 8$.



Il fenomeno di "aliasing"

I risultati ottenuti trovano spiegazione nel fatto che $\sin(5t)$ e $-\sin(3t)$ assumono gli stessi valori nei punti del campionamento $t_j = jh, j = 0, \dots, 8, h = \frac{2\pi}{9}$ e quindi il polinomio trigonometrico ottenuto approssima la funzione $\sin(t) - \sin(3t)$.



Solo aumentando il numero di nodi sarà possibile approssimare correttamente la componente $\sin(5t)$ di frequenza più alta della funzione, altrimenti indistinguibile da quella di frequenza più bassa $-\sin(3t)$.

Questo fenomeno prende il nome di "aliasing": finchè la discretizzazione non è sufficientemente fine da distinguere le frequenze più elevate dalle più basse l'approssimazione sarà poco accurata.

Fenomeno di "aliasing", continua

Dati i nodi di interpolazione $t_j = j \frac{2\pi}{m+1}$, $j = 0, 1, \dots, m$, indichiamo con $f_c = \frac{m+1}{2\pi}$ la **frequenza di campionamento**. Le frequenze delle funzioni $\cos(kt)$ e $\sin(kt)$ risultano essere $f_k = \frac{k}{2\pi}$. Sia $m = 2M$.

Se $k \leq M$ abbiamo che $f_k < \frac{f_c}{2}$ e, per l'unicità del polinomio trigonometrico, tutte le componenti del polinomio trigonometrico $\cos(kt)$ e $\sin(kt)$ sono interpolanti esattamente.

Se $k > M$ le funzioni $\cos(kt)$ e $\sin(kt)$ sono invece ricostruite dal polinomio interpolante con componenti di frequenza inferiore.

Per ricostruire esattamente le funzioni $\cos(kt)$ e $\sin(kt)$ con un polinomio trigonometrico T_{2M+1} la frequenza del campionamento f_c deve essere più grande del doppio della massima frequenza delle sue componenti.

Matlab e FFT

Dato in vettore f la sua trasformata discreta di Fourier viene realizzata in Matlab con l'istruzione

```
>> c=fft(f)/(m+1)
```

mentre la trasformata discreta di Fourier inversa viene richiamata dall'istruzione

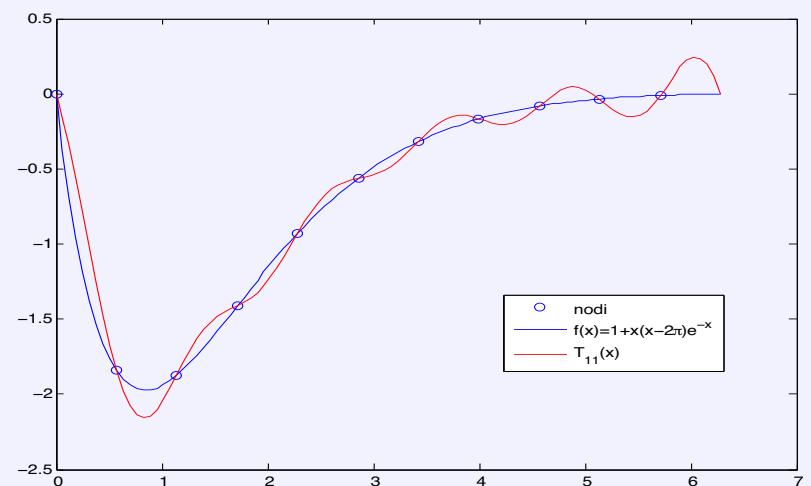
```
>> f=ifft(c)
```

Esempio Matlab di interpolazione trigonometrica

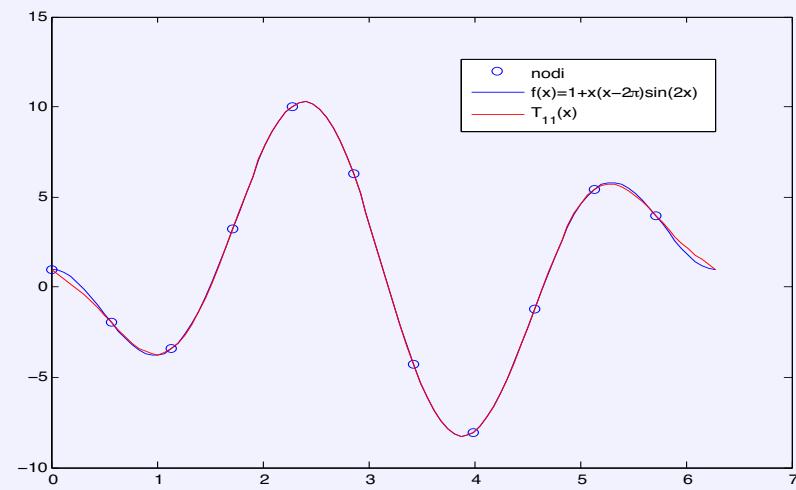
```
>>xx=linspace(0,2*pi,100);
>>yy=xx.*(xx-2*pi).*exp(-xx);
>>n=10; m=n/2;           % caso pari m=n/2;
>>xi=2*pi/(n+1)*[0:n];  % nodi di interpolazione
>>yi=xi.*(xi-2*pi).*exp(-xi);
>>c=fft(yi)/(n+1)
c =
    Columns 1 through 3
    -0.6575           -0.0524 + 0.4205i    0.1206 + 0.1632i
    Columns 4 through 6
    0.1026 + 0.0621i  0.0834 + 0.0250i    0.0746 + 0.0069i
    Columns 7 through 9
    0.0746 - 0.0069i  0.0834 - 0.0250i    0.1026 - 0.0621i
    Columns 10 through 11
    0.1206 - 0.1632i -0.0524 - 0.4205i
```

Esempio, continua

```
>>a=c(2:m+1)+c((n+1):-1:m+2)  
  
a =  
-0.1047 0.2412 0.2051 0.1668 0.1491  
>>b=(c(2:m+1)-c((n+1):-1:m+2))*i;  
  
b =  
-0.8410 -0.3264 -0.1241 -0.0500 -0.0138  
  
>>k=length(a);  
>>cc=cos(xx'*[1:k]);  
>>ss=sin(xx'*[1:k]);  
>>txx=c(1)+cc*a'+ss*b';  
>>plot(xi,yi,'o',xx,yy,'b',xx,txx,'r')  
>> legend('nodi','f(x)=1+x(x-2\pi)e^{-x}', 'T_{11}(x)')
```

Interpolazione trigonometrica per $f(x) = 1 + x(x - 2\pi)e^{-x}$ 

Interpolazione trigonometrica per

$$f(x) = 1 + x(x - 2\pi)\sin(2x)$$


Miglior approssimazione nel senso dei minimi quadrati

Quando i dati $(t_i, y_i), i = 0, \dots, m$ sono affetti da errore di misurazione o incertezza, l'approccio più opportuno consiste nel cercare la funzione

$$\varphi(t, x) = \sum_{j=0}^n x_j \phi_j(t),$$

dove $x = (x_0, \dots, x_n)^T$ e $\phi_i(t), i = 0, \dots, n$, con $\mathbf{n} \ll \mathbf{m}$ sono le funzioni base facilmente calcolabili, che risulta la migliore approssimazione nel senso dei **minimi quadrati**, i.e.

$$\min_{x \in \mathcal{R}^{n+1}} \sum_{i=0}^m (\varphi(t_i, x) - y_i)^2.$$

Tale problema è noto come *data-fitting*. Poiché la funzione $\phi(t_i, x)$ dipende linearmente da $x_i, i = 0, \dots, n$, si parla di data fitting **lineare**.

Miglior approssimazione, continua

Introducendo la matrice

$$A = \begin{pmatrix} \phi_0(t_0) & \phi_1(t_0) & \cdots & \phi_n(t_0) \\ \phi_0(t_1) & \phi_1(t_1) & \cdots & \phi_n(t_1) \\ \vdots & \ddots & \ddots & \vdots \\ \phi_0(t_m) & \phi_1(t_m) & \cdots & \phi_n(t_m) \end{pmatrix}$$

ed il vettore

$$y = (y_0, \dots, y_n)^T$$

il problema si riduce alla risoluzione del sistema lineare sovra-determinato

$$Ax \approx y$$

in modo da minimizzare la norma euclidea del residuo $r(x) = y - Ax$, i.e.

$$\min_{x \in \mathbb{R}^{n+1}} \|Ax - y\|_2 = \min_{x \in \mathbb{R}^{n+1}} \|r(x)\|_2$$

Le proprietà e la struttura della matrice A dipendono sia dalla scelta delle funzioni base $\phi_j, j = 0, \dots, n$ che dai punti $t_i, i = 0, \dots, m$. La risoluzione di tale problema viene affrontata con le strategie illustrate per la risoluzione di sistemi sovra-determinati

Esempio

Siano assegnati i dati $(t_i, y_i), i = 0, \dots, 4$, il data-fitting con un polinomio di grado $n = 2$ porta alla definizione del seguente problema sovra-determinato

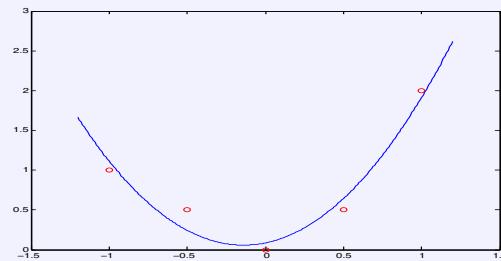
$$Ax = \begin{pmatrix} 1 & t_0 & t_0^2 \\ 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ 1 & t_3 & t_3^2 \\ 1 & t_4 & t_4^2 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} \approx \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}$$

La matrice A è una matrice di Vandermonde.

Esempio Matlab

Assegnati $(-1, 1), (-0.5, 0.5), (0.0), (0.5, 0.5), (1, 2)$, il programma Matlab che calcola la parabola che approssima i dati nel senso dei minimi quadrati è

```
t=-1:0.5:1; y=[1 0.5 0 0.5 2];  
p2=polyfit(t,y,2)  
tt=-1.2:0.01:1.2; pp2=polyval(p2,tt);  
plot(tt,pp2,'b',t,y,'ro')
```



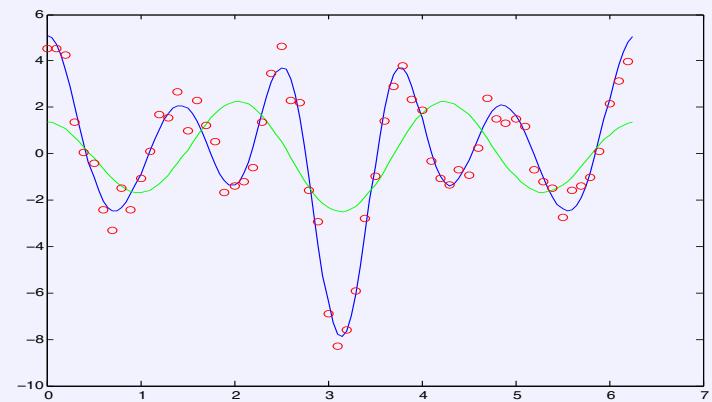
Esempio Matlab

Dobbiamo ricostruire un segnale, descritta dalla funzione Matlab $y=segnale(t)$, sulla base di dati perturbati in maniera casuale. Scegliamo come base di rappresentazione le funzioni $\cos(kt)$, $k = 0, \dots, n$.

```
function x=fitsegnale(n)
t=[0:0.1:pi*2]; y=segnale(t); %campionamento segnale
s=size(t); if s(1)==1 y=y'; t=t'; s(1)=s(2); end %vettore colonne
errore=rand(size(y)); ytilde=y+(2*errore-1); %perturbazione dati
%ricostruzione del segnale
A=ones(s(1),n);
for i=2:n A(:,i)=cos(i*t); end
x=A \ ytilde;
tt=[0:0.05:pi*2]; yy=tt*0;
for i=1:n
    if abs(x(i))>=0.05 yy=yy+x(i)*cos(i*tt); end
end
plot(tt,segnale(tt), 'b', tt,yy, 'g', t,ytilde, 'ro')
```

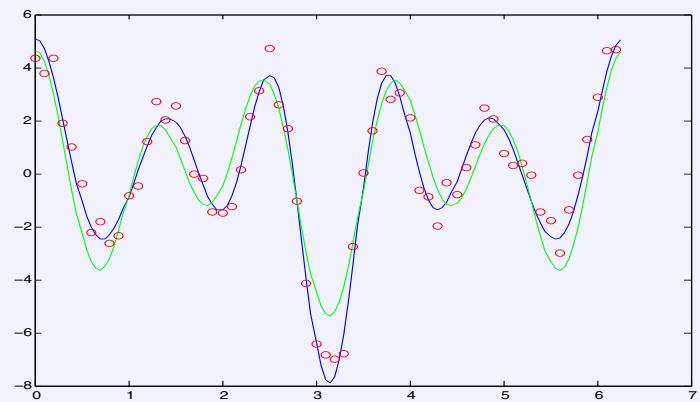
Risultati n=4

$$x = (0.0277, -0.5731, 1.9286, 0.0224)$$



Risultati n=5

$$x = (0.1666, -0.3595, 1.9096, -0.0390, 2.9294)$$



$$f(t) = \frac{1}{(t+1)^2} \quad P_0 = (0, f(0)) = (0, 1)$$

$P_1 = (1, f(1)) = (1, \frac{1}{4})$ e scrivi il polinomio che vi passa $\Rightarrow 3$ punti = PARABOLA

$$P_2 = (2, f(2)) = (2, \frac{1}{9})$$

con tecnica NEWTON: $p(t) = x_0 + x_1(t-t_0)^1 + x_2(t-t_0)^2(t-t_1)^1$
 $= x_0 + x_1 t + x_2 t(t-1)$

$$\begin{array}{c|cc|c} t_i & y_i & x_0 \\ \hline 0 & 1 & 1 \\ 1 & \frac{1}{4} & \frac{1-1}{1-0} = \frac{0}{1} \\ 2 & \frac{1}{9} & \frac{1-\frac{1}{4}}{2-1} = \frac{\frac{3}{4}}{1} \end{array} \quad \begin{array}{c} x_1 \\ \frac{3}{4} \\ x_2 \\ \frac{11}{36} \end{array}$$

$$\Rightarrow p(t) = 1 - \frac{3}{4}t + \frac{11}{36}t(t-1)$$

con MATRICE ASSOCIATA:

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 2 & 2 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{1}{4} \\ \frac{1}{9} \end{pmatrix} \quad \dots$$

Dato $P_3 = (3, f(3)) = (3, \frac{1}{16})$ determina il nuovo polinomio $\tilde{p}(t)$ t.c. $\tilde{p}(t) = p(t) + x_3 t(t-1)(t-2)$

- modo 1: $\tilde{p}(3) = \frac{1}{16} \Rightarrow \frac{1}{16} = p(3) + x_3 \cdot \frac{6}{2 \cdot 1} \Rightarrow x_3 = \frac{1}{16} - \frac{p(3)}{6}$ dove $p(3) = 1 - \frac{3}{4} \cdot 3 + \frac{11}{36} \cdot 3 \cdot 2 = \frac{23}{8}$ sostituisco nella formula iniziale

e poi trova x_3

- modo 2: aggiungo 1 riga nel sist. lineare

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 2 & 2 & 0 \\ 1 & 3 & 6 & 6 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{1}{4} \\ \frac{1}{9} \\ \frac{1}{16} \end{pmatrix} \quad \dots$$

\downarrow
 $t(t-1)(t-2)$

- modo 3: (tabella diff. divise)

$$\begin{array}{c|cc|c} t_i & y_i & x_0 \\ \hline 0 & 1 & 1 \\ 1 & \frac{1}{4} & \frac{1-1}{1-0} = \frac{0}{1} \\ 2 & \frac{1}{9} & \frac{1-\frac{1}{4}}{2-1} = \frac{\frac{3}{4}}{1} \\ 3 & \frac{1}{16} & \frac{1-\frac{1}{9}}{3-2} = \frac{\frac{8}{9}}{1} \end{array} \quad \dots \quad x_3$$

Determina q_1 di grado 1 che approssima al meglio $P_0 P_1 P_2$

$$q_1 = a + bt$$

$$\underbrace{\begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix}}_A \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{1}{4} \\ \frac{1}{9} \end{pmatrix}$$

$$A^T A = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \\ 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} 3 & 3 \\ 3 & 5 \end{pmatrix}$$

$$A^T y = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} 1 \\ \frac{1}{4} \\ \frac{1}{9} \end{pmatrix} = \begin{pmatrix} 1 + \frac{1}{4} + \frac{1}{9} = \frac{36+9+4}{36} = \frac{49}{36} \\ \frac{1}{4} + \frac{2}{9} = \frac{13}{36} \end{pmatrix}$$

$$\begin{cases} 3a + 3b = \frac{49}{36} \\ 3a + 5b = \frac{13}{36} \end{cases}$$

$$2b = \frac{17}{36} - \frac{49}{36} = -\frac{32}{36} \Rightarrow b = -\frac{32}{36} \cdot \frac{1}{2}$$

$$a = \dots$$

⋮

errore interpolaz $\rightarrow |f(t) - p(t)| = \frac{|(t)(t-1)(t-2)|}{3!} \cdot f^{(3)}(\xi) \leq \frac{1}{6} \cdot 2^3 \max_{t \in [q_1]} (f^{(3)}(t)) \leq \frac{8}{6} \cdot 24 = 32$

derivata 3°

grado