ANALISI NUMERICA
CALCOLO NUMERICO
(A.A. 2012-2013)

Prof. F. Pitolli

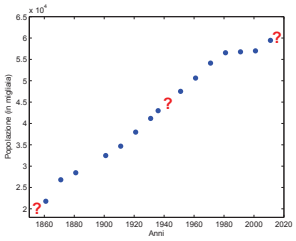
Appunti delle lezioni sull'approssimazione

Esempio 1

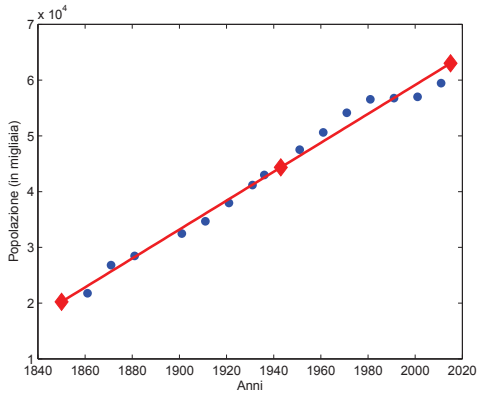
Nella **tavola** seguente è riportata la **popolazione** (in migliaia) dell'Italia censita ogni **10 anni** tra il **1861** e il **2011** (dati ISTAT).

Anno	1861	1871	1881	1901	1911	1921	1931	1936
Popolazione (in migliaia)	21'777	26'801	28'460	32'475	34'671	37'974	41'177	42'994

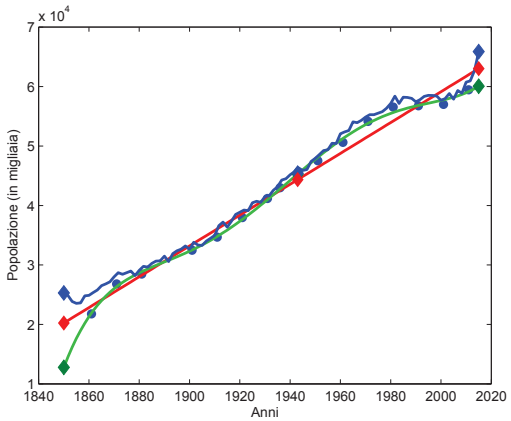
Anno	1951	1961	1971	1981	1991	2001	2011
Popolazione (in migliaia)	47'516	50'624	54'137	56'557	56'778	56'996	59'464



Determinare il **tasso di crescita** e approssimare la popolazione negli anni 1850, 1943, 2015. Come si può stimare l'**accuratezza** dei valori ottenuti?

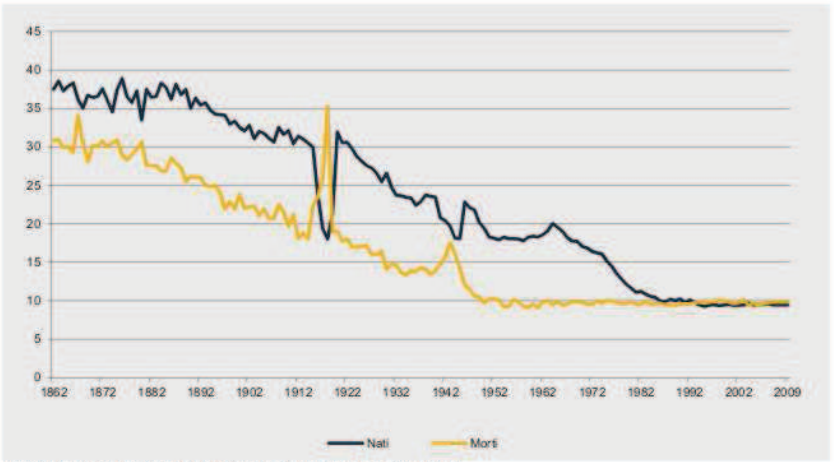



```
>> X = [1861:10:1881 1901:10:1931 1936 1951:10:2011];
>> Y=[21777 26801 28460 32475 34671 37974 41177 42994 47516 ...
      50624 54137 56557 56778 56996 59464];
>> XX = linspace(1850,2015);
>> p = polyfit(X,Y,1)
    1.0e+005 *
              0.00259298408877 -4.59466448675812
>> YY = polyval(p,XX)
```


```
>> X = [1861:10:1881 1901:10:1931 1936 1951:10:2011];
>> Y=[21777 26801 28460 32475 34671 37974 41177 42994 47516 ...
50624 54137 56557 56778 56996 59464];
>> XX = linspace(1850,2015);
>> p = polyfit(X,Y,5);YY = polyval(p,XX)
Warning:  Polynomial is badly conditioned.
>> p = polyfit(X,Y,12);YY = polyval(p,XX)
Warning:  Polynomial is badly conditioned.
```

Figura 2.1 - Nati e morti - Anni 1862-2009 ai confini attuali (per 1.000 abitanti)



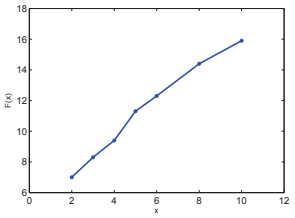
Fonte: Istat, Ricostruzione della popolazione residente e del bilancio demografico

Esempio 2

Nella tabella sono riportate le **misure sperimentali** relative alla **forza** $F(x)$ necessaria per allungare una **molla** fino alla **lunghezza** x.

x	2	3	4	5	6	8	10
$F(x)$	7.0	8.3	9.4	11.3	12.3	14.4	15.9

Determinare la **costante elastica** della molla.

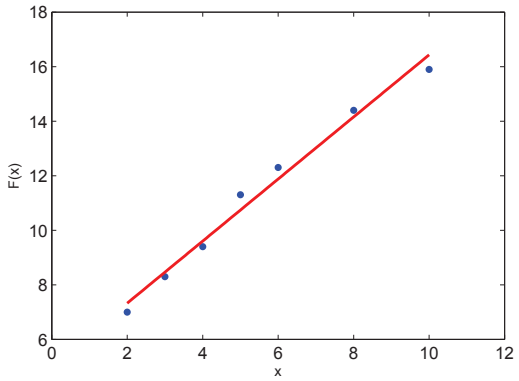


Soluzione

La forza $F(x)$ necessaria per allungare una molla fino alla lunghezza x è data da $\boxed{F(x) = k(x - l)}$ (**Legge di Hooke**) dove k è la **costante elastica** e l è la **lunghezza a riposo** della molla.

Le misure date possono essere approssimate con la **retta di regressione** $r(x) = a_1\, x + a_0$.

La **costante elastica** della molla è data dal **coefficiente angolare** della retta di regressione $k = a_1$.




```
>> X=[2 , 3 , 4 , 5 , 6 , 8 , 10]
>> F=[7.0 , 8.3 , 9.4 , 11.3 , 12.3 , 14.4 , 15.9]
>> X2 = linspace(X(1),X(7))
>> p = polyfit(X,F,1);y2 = polyval(p,X2)
```

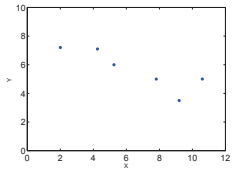
Esempio 3

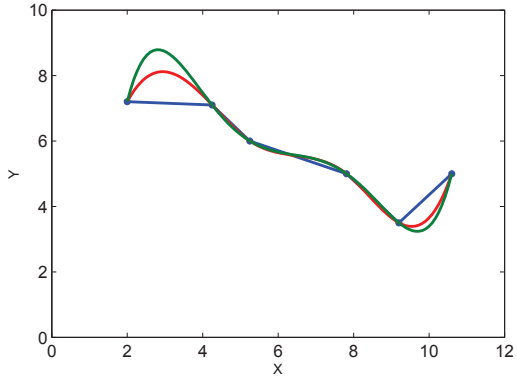
Per **misurare il diametro** di alcuni fori di **coordinate**

x pollici	2.00	4.25	5.25	7.81	9.20	10.60
y pollici	7.2	7.1	6.0	5.0	3.5	5.0

praticati su una lastra di metallo di dimensione $15'' \times 10''$ viene utilizzato un laser collocato all'estremità di un **braccio meccanico** di un robot.

Determinare il **cammino ottimale** che il braccio deve compiere per collegare i fori. Il cammino deve essere sufficientemente **regolare**, in modo da impedire variazioni di direzione troppo brusche, e **breve**.



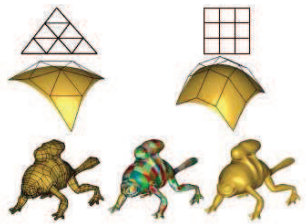



```
>> X=[2.00 4.25 5.25 7.81 9.20 10.60]
>> Y=[7.2 7.1 6.0 5.0 3.5 5.0]
>> x2 = linspace(X(1),X(6))
>> yy = spline(X,Y,x2)
>> p = polyfit(X,Y,5);y2 = polyval(p,x2)
```


Esempio 4

Date le coordinate di alcuni punti nello spazio, disegnare una superficie che riproduca l'andamento dei punti.

$\longrightarrow$ Computer graphics, disegno di fonts ...



1998 Academy Award
Best Animated Short Film

Approssimazione di dati e funzioni

Problema

Data la **tabella** $\{x_i, y_i\}$, $i = 0, \ldots, n$, si vuole trovare una **funzione analitica** φ_M che **approssimi** i dati.

La **tabella** $\{x_i, y_i\}$ può essere il risultato di **misure sperimentali** oppure può rappresentare i valori di una funzione la cui **espressione analitica** è nota ma **complicata** da calcolare direttamente.

Per poter costruire una **funzione approssimante** bisogna stabilire

- in quale **classe** di funzioni si vuole operare
- il **metodo di approssimazione**

Classi di funzioni approssimanti

- **Polinomi algebrici** di grado M a coefficienti reali: $M+1$ **parametri**

$$P_M(x) = a_0 + a_1 x + \cdots + a_{M-1}x^{M-1} + a_M x^M \qquad a_k \in \mathbf{R} \ \ \forall k$$

- **Polinomi trigonometrici** di ordine M a coefficienti reali: $2M+1$ **parametri**

$$T_M(x) = \sum_{k=0}^{M} \Big(a_k \cos(k\,x) + b_k \sin(k\,x)\Big) \qquad a_k, b_k \in \mathbf{R} \ \ \forall k$$

- **Funzioni razionali**: $M+N+2$ **parametri**

$$R_{M,N}(x) = \frac{p_M(x)}{p_N(x)} \qquad p_M, p_N \text{ polinomi}$$

- **Funzioni esponenziali**: $2M$ **parametri**

$$G_M(x) = \sum_{k=1}^{M} a_k \exp(b_k\,x) \qquad a_k, b_k \in \mathbf{R} \ \ \forall k$$

- **Funzioni splines**: polinomi a tratti di grado M e regolarità C^{M-1}

Metodi di approssimazione

- **Interpolazione**: si sceglie la funzione approssimante φ_M in modo che

$$\boxed{\varphi_M(x_i) = y_i \quad i = 0, 1, \ldots, n \qquad \textbf{Condizioni di interpolazione}}$$

Si usa quando i dati y_i sono **accurati**.

- **Approssimazione ai minimi quadrati** di dati discreti: si sceglie la funzione approssimante φ_M in modo che **minimizzi** la quantità

$$\boxed{\sum_{i=0}^{n} \Big[\varphi_M(x_i) - y_i\Big]^2 \qquad \textbf{Scarto quadratico}}$$

oppure, introducendo i **pesi** w_i,

$$\boxed{\sum_{i=0}^{n} w_i \, \Big[\varphi_M(x_i) - y_i\Big]^2 \qquad \textbf{Scarto quadratico pesato}}$$

Si usa quando i dati y_i sono poco **accurati** e in **numero elevato**.

Esempio: retta di regressione

Approssimazione ai minimi quadrati

Problema.
Data la **tabella** $\{x_i, y_i\}$, $i = 0, 1, \ldots, n$, si vuole trovare una **funzione analitica** φ_M che **approssimi** i dati.

In questo caso la **tabella** è il risultato di **misure sperimentali** ciascuna delle quali è affetta da un **errore di misura** ε_i.

Metodo di approssimazione: si sceglie la funzione approssimante φ_M in modo da **minimizzare**

$$\sum_{i=0}^{n} \big[\varphi_M(x_i) - y_i\big]^2 \qquad \textbf{Scarto quadratico}$$

oppure, introducendo i **pesi** $w_i > 0$, $\forall i$,

$$\sum_{i=0}^{n} w_i \, \big[\varphi_M(x_i) - y_i\big]^2 \qquad \textbf{Scarto quadratico pesato}$$

Caso lineare

Funzione approssimante:
$\varphi_M(x)$ dipende **linearmente** da M **parametri**:

$$\gamma_0, \gamma_1, \cdots, \gamma_M \qquad \boxed{M << n}$$

$$\Rightarrow \varphi_M(x) = \gamma_0\psi_0(x) + \gamma_1\psi_1(x) + \cdots + \gamma_M\psi_M(x)$$

dove $\{\psi_k(x)\}_{k=0,\ldots,M}$ è una base per lo spazio di approssimazione

Metodo di approssimazione: si minimizza lo **scarto quadratico**

$$\sigma^2(\gamma_0, \gamma_1, \ldots, \gamma_M) = \sum_{i=0}^{n} \Big[\underbrace{\gamma_0\psi_0(x_i) + \gamma_1\psi_1(x_i) + \cdots + \gamma_M\psi_M(x_i)}_{\varphi_M(x_i)} - y_i\Big]^2$$

Risolvere il problema dell'**approssimazione ai minimi quadrati** vuol dire individuare i **coefficienti reali** γ_k che rendono **minimo** $\sigma^2(\gamma_0, \cdots, \gamma_M)$.

Nota. L'approssimante ai minimi quadrati in generale **non passa** per i valori $\{x_i, y_i\}$ ma "*vicino*" ad essi.

Minimizzazione di σ

- Per minimizzare σ bisogna annullare il **gradiente**

$$\Rightarrow \frac{\partial \sigma^2}{\partial \gamma_k} = \frac{\partial}{\partial \gamma_k} \sum_{i=0}^{n} [\varphi_M(x_i) - y_i]^2 = 0 \qquad k = 0, 1, ..., M$$

$$\Rightarrow 2 \sum_{i=0}^{n} \left(\varphi_M(x_i) - y_i\right) \frac{\partial \varphi_M(x_i)}{\partial \gamma_k} = 0 \qquad k = 0, 1, ..., M$$

$$\Rightarrow 2 \sum_{i=0}^{n} \left[\gamma_0 \psi_0(x_i) + \gamma_1 \psi_1(x_i) + \cdots + \gamma_M \psi_M(x_i) - y_i\right] \psi_k(x_i) = 0 \qquad k = 0, 1, ..., M$$

$$\Rightarrow \boxed{\gamma_0 \sum_{i=0}^{n} \psi_0(x_i)\psi_k(x_i) + \gamma_1 \sum_{i=0}^{n} \psi_1(x_i)\psi_k(x_i) + \cdots + \cdots + \gamma_M \sum_{i=0}^{n} \psi_M(x_i)\psi_k(x_i) = \sum_{i=0}^{n} y_i \psi_k(x_i) \quad k = 0, \ldots, M}$$

Sistema delle equazioni normali

- Per trovare i coefficienti incogniti a_k bisogna risolvere il **sistema delle equazioni normali**:

$$\begin{cases} h_{0,0}\,\gamma_0 + h_{0,1}\,\gamma_1 + ... + h_{0,M}\,\gamma_M = v_0 \\ h_{0,1}\,\gamma_0 + h_{1,1}\,\gamma_1 + ... + h_{1,M}\,\gamma_M = v_1 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ h_{0,M}\,\gamma_0 + h_{1,M}\,\gamma_1 + ... + h_{M,M}\,\gamma_M = v_M \end{cases} \quad \text{dove} \quad h_{j,k} := \sum_{i=0}^{n} \psi_j(x_i)\psi_k(x_i), \qquad v_k := \sum_{i=0}^{n} y_i \psi_k(x_i)$$

$$\Updownarrow$$

$$H\,\Gamma = B$$

$$\Gamma = [\gamma_0, \gamma_1, ..., \gamma_M]^T \qquad H = \begin{bmatrix} h_{0,0} & h_{0,1} & \cdots & h_{0,M} \\ h_{0,1} & h_{1,1} & \cdots & h_{1,M} \\ \cdots & \cdots & \cdots & \cdots \\ h_{0,M} & h_{1,M} & \cdots & h_{M,M} \end{bmatrix} \in \mathbf{R}^{(M+1)\times(M+1)}$$

$$B = [v_0, v_1, ..., v_M]^T$$

Unicità della soluzione

Definiamo il vettore $Y = [y_0, y_1, ..., y_n]^T \Rightarrow \boxed{B = V^T Y \qquad H = V^T V}$

dove $V = \begin{bmatrix} \psi_0(x_0) & \psi_1(x_0) & \psi_2(x_0) & \cdots & \psi_M(x_0) \\ \psi_0(x_1) & \psi_1(x_1) & \psi_2(x_1) & \cdots & \psi_M(x_1) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \psi_0(x_n) & \psi_1(x_n) & \psi_2(x_n) & \cdots & \psi_M(x_n) \end{bmatrix} \in \mathbf{R}^{(n+1)\times(M+1)}$

$\Rightarrow$ Se le funzioni $\{\psi_k(x)\}$ sono **linearmente indipendenti**, la matrice V è **regolare**

$\Rightarrow$ Per ogni $X \in \mathbf{R}^N$ si ha $X^T H X = (VX)^T(VX) = ||VX||_2^2 \geq 0$. Inoltre, per la regolarità di V, l'uguaglianza vale se e solo se $X = 0$

$\Rightarrow$ H è **definita positiva** e quindi **regolare**

$\Rightarrow$ Il sistema delle **equazioni normali** ammette un' **unica soluzione** Γ

Calcolo dell'hessiano

- Per verificare che la soluzione del sistema sia un **minimo** bisogna studiare l'**hessiano**

$$\left[\frac{\partial^2 \sigma^2}{\partial \gamma_j \partial \gamma_k}\right] = 2 \left[\sum_{i=0}^{n} \psi_j(x_i)\psi_k(x_i)\right]_{j,k=0,...,M} = 2H$$

$\Rightarrow$ La matrice hessiana $2H$ è **definita positiva**

$\Rightarrow$ la **soluzione** del sistema delle equazioni normali corrisponde a un **minimo**.

Polinomio algebrico ai minimi quadrati

Tabella: $\{x_i, y_i\} \qquad i = 0, 1, ..., n$

Funzione approssimante:

$$P_M(x) = a_0 + a_1 x + \cdots + a_{M-1} x^{M-1} + a_M x^M \qquad \boxed{M << n}$$

Metodo di approssimazione: si minimizza lo **scarto quadratico**

$$\sigma^2(a_0, a_1, ..., a_M) = \sum_{i=0}^{n} [\underbrace{a_0 + a_1 x_i + \cdots + a_{M-1} x_i^{M-1} + a_M x_i^M}_{P_M(x_i)} - y_i]^2$$

Funzioni di base:

$$\psi_0(x) = 1, \ \psi_1(x) = x, \ \ldots, \ \psi_k(x) = x^k, \ \cdots, \ \psi_M(x) = x^M$$

Sistema delle equazioni normali

Definizioni: $s_k := \sum_{i=0}^{n} x_i^k \qquad v_k := \sum_{i=0}^{n} y_i x_i^k \qquad k = 0, 1, ..., M$

Il **sistema delle equazioni normali** diventa

$$\begin{cases} s_0 a_0 + s_1 a_1 + ... + s_M a_M = v_0 \\ s_1 a_0 + s_2 a_1 + ... + s_{M+1} a_M = v_1 \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ s_M a_0 + s_{M+1} a_1 + ... + s_{2M} a_M = v_M \end{cases} \iff HA = B$$

$$A = [a_0, a_1, ..., a_M]^T \qquad B = [v_0, v_1, ..., v_M]^T \qquad H = \begin{bmatrix} s_0 & s_1 & \cdots & s_M \\ s_1 & s_2 & \cdots & s_{M+1} \\ \cdots & \cdots & \cdots & \cdots \\ s_M & s_{M+1} & \cdots & s_{2M} \end{bmatrix} \in \mathbf{R}^{(M+1)\times(M+1)}$$

Condizionamento del sistema delle equazioni normali

$$H = [h_{jk}]_{j,k=0,M} \qquad h_{jk} = \underbrace{\frac{1}{n+1} \sum_{i=0}^{n} x_i^{j+k}}_{\text{somma integrale}} \approx \int_{x_0}^{x_n} x^{j+k}\, dx = \left. \frac{x^{j+k+1}}{j+k+1} \right|_{x_0}^{x_n}$$

Esempio: $[x_0, x_n] = [0, 1]$

$$H \propto \left[\tfrac{1}{j+k+1}\right]_{j,k=0,M} = \begin{bmatrix} 1 & 1/2 & 1/3 & \cdots & 1/M \\ 1/2 & 1/3 & 1/4 & \cdots & 1/(M+1) \\ 1/3 & 1/4 & 1/5 & \cdots & 1/(M+2) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1/M & 1/(M+1) & 1/(M+2) & \cdots & 1/2M \end{bmatrix}$$

H è proporzionale alla **matrice di Hilbert** $\Rightarrow$ H è **malcondizionata**

Retta di regressione

La **retta di regressione** è il **polinomio di grado 1**

$$P_1(x) = a_0 + a_1 x$$

che approssima i dati $\{x_i, y_i\}$, $i = 0, 1, \ldots, n$, $(n >> 1)$ nel senso dei **minimi quadrati**.

Equazioni normali $\Rightarrow$ **Soluzione**

$$\begin{cases} a_0 s_0 + a_1 s_1 = v_0 \\ a_0 s_1 + a_1 s_2 = v_1 \end{cases} \Rightarrow \quad a_0 = \frac{v_0 s_2 - v_1 s_1}{s_0 s_2 - s_1^2} \qquad a_1 = \frac{s_0 v_1 - s_1 v_0}{s_0 s_2 - s_1^2}$$

dove

$$s_0 = n + 1 \qquad s_1 = \sum_{i=0}^{n} x_i \qquad s_2 = \sum_{i=0}^{n} x_i^2$$

$$v_0 = \sum_{i=0}^{n} y_i \qquad v_1 = \sum_{i=0}^{n} y_i x_i$$

Esempio 2

La forza $F(x)$ necessaria per allungare una molla fino alla lunghezza x è data da $\boxed{F(x) = k(x - l)}$ (**Legge di Hooke**) dove k è la **costante elastica** e l è la **lunghezza a riposo** della molla.

Nella tabella sono riportate le **misure sperimentali** relative alla **forza** $F(x)$ necessaria per allungare una **molla** fino alla **lunghezza** x.

i	0	1	2	3	4	5	6
x_i	2	3	4	5	6	8	10
$F(x_i)$	7.0	8.3	9.4	11.3	12.3	14.4	15.9

Determinare la **costante elastica** della molla.

Esempio 2: soluzione

Per trovare la costante elastica della molla si devono approssimare i dati in tabella con il polinomio ai minimi quadrati di primo grado (**retta di regressione**): $\boxed{P_1(x) = a_0 + a_1\, x}$

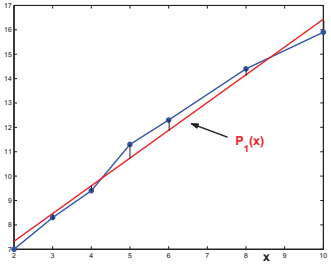
Il coefficiente a_1 fornisce l'approssimazione della **costante elastica**. Risolvendo il sistema delle equazioni normali si ottiene

$$a_0 = 5.049 \qquad a_1 = 1.1383.$$

Per questi valori dei coefficienti si ha

$$\sigma^2(a_0, a_1) = \sum_{i=0}^{6} [a_0 + a_1 x_i - y_i]^2 =$$

$$= 1.0071$$



Retta di regressione: programma Fortran

```
      program rettaregr

      implicit none
      real s0, s1, s2, v0, v1, a0, a1
      real x(125), y(125), den
      integer n, i
      character filename*15
*
* Lettura dei dati di input
*
      write (*,*) 'Inserisci nome del file'
      read (*,*) filename
      open(1,file=filename,status='old')
      read (1,*) n
      if (n .gt. 125) stop 'n troppo grande'
      write(*,*) 'n=', n
      read (1,*) (x(i),y(i),i=1,n)
      close(1)
```



```
* Calcolo di s0, s1, s2, v0, v1
*
      s0 = float(n)
      s1 = 0.
      s2 = 0.
      v0 = 0.
      v1 = 0.
      do i = 1, n
         s1 = s1 + x(i)
         s2 = s2 + x(i)*x(i)
         v0 = v0 + y(i)
         v1 = v1 + x(i)*y(i)
      enddo
*
* Calcolo coefficienti della retta di regressione
*
      den = s0*s2-s1*s1
      a0 = (v0*s2-v1*s1) / den
      a1 = (s0*v1-s1*v0) / den
*
* Output
*
      write (*,*) 'a0=', a0, 'a1 e'' ', a1
*
* Fine del programma
      end
```

Interpretazione probabilistica - 1

Siano x una **variabile deterministica** e $y = a_0 + a_1 x$ la **variabile dipendente**, legata a x da una **relazione lineare**. Per ogni coppia $\{x_i, y_i\}$ valgono le relazioni $y_i = a_0 + a_1 x_i$, $i = 0, \ldots, n$.
Supporremo che i **dati** $\{x_i, y_i\}$ siano affetti da **rumore** con **errore statistico** ε_i.

Definizione. A partire dai dati $\{x_i, y_i\}$ si definiscono la **varianza** e la **covarianza** rispettivamente come

$$var(x) = \frac{1}{n+1}\sum_i (x_i - \bar{x})^2 \qquad cov(x,y) = \frac{1}{n+1}\sum_i (x_i - \bar{x})(y_i - \bar{y})$$

dove

$$\bar{x} = \frac{1}{n+1}\sum_i x_i \qquad \bar{y} = \frac{1}{n+1}\sum_i y_i$$

sono le **medie osservate**.

Interpretazione probabilistica - 2

Ipotesi:

1) x è una **variabile deterministica**
2) $E(\varepsilon_i) = 0$ (**valore atteso**)
3) $var(\varepsilon_i)$ **costante** per ogni i
4) $cov(\varepsilon_i, \epsilon_j) = 0$ per ogni $i \neq j$

Nel metodo dei **minimi quadrati** si minimizza la quantità

$$\sigma^2(a_0, a_1) = \sum_i (a_0 + a_1 x_i - y_i)^2$$

Se valgono **Hp. 1-4** i coefficienti a_0 e a_1, soluzione del problema di minimo, possono essere scritti come

$$a_1 = \frac{(n+1)(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{(n+1)(\sum_i x_i^2) - (\sum_i x_i)^2} = \frac{cov(x,y)}{var(x)}$$

$$a_0 = \frac{(\sum_i y_i)(\sum_i x_i^2) - (\sum_i x_i)(\sum_i x_i y_i)}{(n+1)(\sum_i x_i^2) - (\sum_i x_i)^2} = \bar{y} - a_1\bar{x} = \bar{y} - \frac{cov(x,y)}{var(x)}\bar{x}$$

Interpolazione

Tabella: $\{x_i, y_i\}$ $\quad i = 0, 1, \ldots, n$

Funzione approssimante:
$\varphi_M(x)$ dipende **linearmente** da $M + 1 = n + 1$ **parametri**:

$$\gamma_0, \gamma_1, \cdots, \gamma_n$$
$$\Downarrow$$

$$V = \begin{bmatrix} \psi_0(x_0) & \psi_1(x_0) & \psi_2(x_0) & \cdots & \psi_n(x_0) \\ \psi_0(x_1) & \psi_1(x_1) & \psi_2(x_1) & \cdots & \psi_n(x_1) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \psi_0(x_n) & \psi_1(x_n) & \psi_2(x_n) & \cdots & \psi_n(x_n) \end{bmatrix} \in \mathbf{R}^{(n+1)\times(n+1)}$$

$$\underbrace{V^T V}_{H}\,\Gamma = \underbrace{V^T Y}_{B} \Rightarrow \boxed{V\,\Gamma = Y}$$

Risolvere il **problema dell'interpolazione** equivale a risolvere il sistema lineare $\quad V\,\Gamma = Y$

Condizioni di interpolazione

$$V\Gamma = Y$$
$$\Updownarrow$$

$$\begin{cases} \underbrace{\gamma_0\,\psi_0(x_0) + \gamma_1\,\psi_1(x_0) + \gamma_2\,\psi_2(x_0) + \cdots + \gamma_n\,\psi_n(x_0)}_{\varphi_n(x_0)} = y_0 \\ \underbrace{\gamma_0\,\psi_0(x_1) + \gamma_1\,\psi_1(x_1) + \gamma_2\,\psi_2(x_1) + \cdots + \gamma_n\,\psi_n(x_1)}_{\varphi_n(x_1)} = y_1 \\ \vdots \qquad\qquad \vdots \\ \underbrace{\gamma_0\,\psi_0(x_n) + \gamma_1\,\psi_1(x_n) + \gamma_2\,\psi_2(x_n) + \cdots + \gamma_n\,\psi_n(x_n)}_{\varphi_n(x_n)} = y_n \end{cases}$$

$$\Updownarrow$$

$$\varphi_n(x_0) = y_0, \quad \varphi_n(x_1) = y_1, \quad \cdots \quad \varphi_n(x_n) = y_n$$

$\Rightarrow$ La funzione interpolante **passa** per i valori $\{x_i, y_i\}$.

Interpolazione polinomiale

Tabella: $\{x_i, y_i\} \qquad i = 0, \ldots, n$

Intervallo di interpolazione: $[a, b] = [x_0, x_n]$

Funzione approssimante: $p_n(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} + a_n x^n$

Metodo di approssimazione: $p_n(x_i) = y_i \quad (i = 0, 1, \ldots, n)$ $\rightarrow$ **Interpolazione**

Risolvere il **problema dell'interpolazione** vuol dire individuare il polinomio p_n, cioè i **coefficienti reali** a_k, che soddisfano le **condizioni di interpolazione**. Questo equivale a risolvere il **sistema lineare**

$$\begin{array}{l} p(x_0) = \\ p(x_1) = \\ \\ p(x_n) = \end{array} \left\{ \begin{array}{ccccccccccc} a_0 & + & a_1 x_0 & + & a_2 x_0^2 & + & \cdots & + & a_n x_0^n & = & y_0 \\ a_0 & + & a_1 x_1 & + & a_2 x_1^2 & + & \cdots & + & a_n x_1^n & = & y_1 \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ \vdots & & \vdots & & \vdots & & \vdots & & \vdots & & \vdots \\ a_0 & + & a_1 x_n & + & a_2 x_n^2 & + & \cdots & + & a_n x_n^n & = & y_n \end{array} \right. \rightarrow \quad VA = Y$$

Unicità del polinomio interpolatore

$\boxed{VA = Y}$ con $V = \underbrace{\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix}}_{\text{Matrice di Vandermonde}} \quad A = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{pmatrix} \quad Y = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{pmatrix}$

La **matrice di Vandermonde** di $n+1$ nodi **distinti** $\{x_i\}$, $i = 0, \ldots, n+1$, è **regolare** poiché

$$\det V = \prod_{j>i} (x_i - x_j) \neq 0$$

$\Rightarrow$ esiste un'**unica** soluzione A del sistema.

$\Downarrow$

Esiste **uno e uno solo** polinomio p_n di grado n che verifica le **condizioni di interpolazione**

$$p_n(x_i) = y_i \qquad i = 0, \ldots, n$$

Condizionamento della matrice di Vandermonde

La **matrice di Vandermonde** può essere **malcondizionata**.

Esempio 1.: $n+1$ nodi **equispaziati** in $[0, 1]$

$n+1$	1	2	3	4	5	6
$K_1(V)$	4	24	216	1.7067e+003	1.2500e+004	9.8784e+004

$n+1$	7	8	9	10
$K_1(V)$	8.1271e+005	6.2915e+006	4.8184e+007	4.0042e+008

Esempio 2.: $n+1$ nodi di **Chebyshev** in $[0, 1]$

$n+1$	1	2	3	4	5	6
$K_1(V)$	2.4142	6.9761	18.637	46.951	1.3157e+002	3.7648e+002

$n+1$	7	8	9	10
$K_1(V)$	1.0200e+003	2.6550e+003	6.7024e+003	1.6514e+004

Nota. Poiché la matrice di Vandermonde è **malcondizionata** per n elevato e, inoltre, il **costo computazionale** per la soluzione del sistema lineare $VA = Y$ può essere **elevato**, si preferisce ricorrere ad **altre strategie** per costruire l' **unico polinomio interpolatore** p_n.

Polinomi di base di Lagrange

Base dei monomi: $\{1, x, x^2, \ldots, x^{n-1}, x^n\}$

- x^k è un monomio di grado k

$\Rightarrow p_n(x) = a_0 + a_1 x + \cdots + a_{n-1} x^{n-1} + a_n x^n$

Polinomi di base di Lagrange: $\{\ell_0(x), \ell_1(x), \ldots, \ell_{n-1}(x), \ell_n(x)\}$

$$\ell_k(x) := \prod_{\substack{j=0 \\ j \neq k}}^{n} \frac{x - x_j}{x_k - x_j} = \frac{x - x_0}{x_k - x_0} \cdot \frac{x - x_1}{x_k - x_1} \cdots \frac{x - x_{k-1}}{x_k - x_{k-1}} \cdot \frac{x - x_{k+1}}{x_k - x_{k+1}} \cdots \frac{x - x_n}{x_k - x_n}$$

- $\ell_k(x)$, $k = 0, \ldots, n$, è un **polinomio di grado** n
- $\ell_k(x_i) = \begin{cases} 1 & \text{se } i = k \\ 0 & \text{se } i \neq k \end{cases}$
- l'espressione di $\ell_k(x)$ dipende solo dai **nodi** $\{x_i\}$

Espressione di Lagrange del polinomio interpolatore

$$p_n(x) = \gamma_0 \ell_0(x) + \gamma_1 \ell_1(x) + \cdots + \gamma_{n-1}\ell_{n-1}(x) + \gamma_n \ell_n(x)$$

$$\boxed{V\,\Gamma = Y}$$

$$V = \begin{bmatrix} \ell_0(x_0) & \ell_1(x_0) & \ell_2(x_0) & \cdots & \ell_n(x_0) \\ \ell_0(x_1) & \ell_1(x_1) & \ell_2(x_1) & \cdots & \ell_n(x_1) \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ \ell_0(x_n) & \ell_1(x_n) & \ell_2(x_n) & \cdots & \ell_n(x_n) \end{bmatrix} = I \Rightarrow \boxed{\Gamma = Y}$$

$$\Rightarrow \qquad p_n(x) = \sum_{i=0}^{n} y_i \; \ell_i(x) \equiv L_n(x) \in \mathbb{P}_n$$

compaiono esplicitamente i **valori** nei nodi

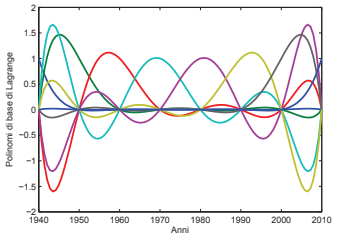
$\ell_i(x)$ dipende solo dai **nodi**

Esempio 1

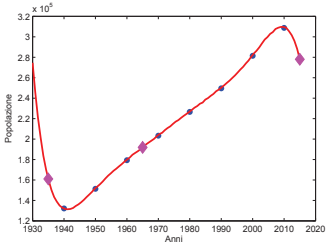
Data la tabella

Anno	1940	1950	1960	1970	1980	1990	2000	2010
Popolazione (in migliaia)	132 165	151 326	179 323	203 302	226 542	249 633	281 421	308 745

approssimare la popolazione negli anni 1930, 1965, 2015. Quanto sono **accurati** i valori ottenuti?



Polinomi di base di Lagrange



Polinomio interpolatore (il simbolo $\diamond$ indica i punti di interpolazione)

Interpolazione di polinomi

L'interpolazione è **esatta** per i polinomi q_m di grado $m \le n$.

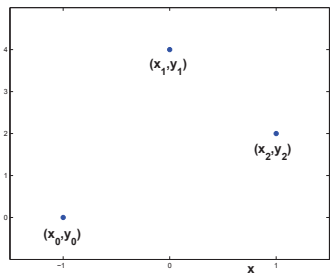
$$\{x_k, q_m(x_k)\} \quad k = 0, \ldots, n \Rightarrow L_n(x) = \sum_{k=0}^{n} q_m(x_k)\; \ell_k(x) \equiv q_m(x)$$

Esempio.

Data la tavola

i	0	1	2
x_i	-1	0	1
y_i	0	4	2

costruire il relativo **polinomio interpolatore**.



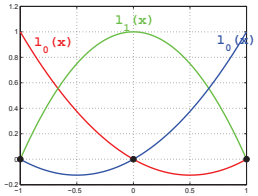
Soluzione

I dati sono $n+1 = 3$ quindi il polinomio interpolatore è un polinomio di **grado 2**.

$$\ell_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{x(x-1)}{(-1)(-1-1)} = \frac{1}{2}x(x-1)$$

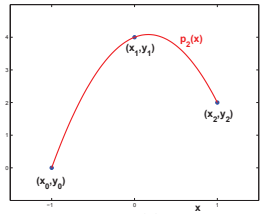
$$\ell_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x+1)(x-1)}{(1)(-1)} = -(x+1)(x-1)$$

$$\ell_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x+1)x}{(1+1)(1)} = \frac{1}{2}(x+1)x$$



$$\Rightarrow \quad p_2(x) = \sum_{k=0}^{2} y_k\; \ell_k(x) = 0 \cdot \ell_0(x) + 4 \cdot \ell_1(x) + 2 \cdot \ell_2(x)$$

$$= -4(x+1)(x-1) + (x+1)x = -3x^2 + x + 4$$



Aggiungiamo alla tabella la coppia di valori $(x_3, y_3) = (2, -6)$ e calcoliamo il nuovo polinomio interpolatore.

i	0	1	2	3
x_i	-1	0	1	2
y_i	0	4	2	-6

$\Rightarrow$ il **polinomio interpolatore** è di **grado 3**

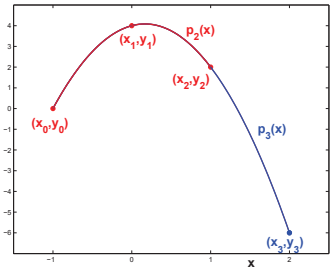
$$\ell_0(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} = -\frac{1}{3}x(x-1)(x-2)$$

$$\ell_1(x) = \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} = \frac{1}{2}(x+1)(x-1)(x-2)$$

$$\ell_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} = -\frac{1}{2}(x+1)x(x-2)$$

$$\ell_3(x) = \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} = \frac{1}{3}(x+1)x(x-1)$$

$$\Rightarrow p_3(x) = \sum_{k=0}^{3} y_k\ \ell_k(x) = $$
$$= -3x^2 + x + 4 = p_2(x)!!!$$



Interpolazione di funzioni costanti

L'interpolazione è **esatta** per i polinomi q_m di grado $m \le n$.

Se $m = 0 \Rightarrow L_n(x) = \sum_{k=0}^{n} q_0(x_k)\ \ell_k(x) \equiv q_0(x) = costante$

In particolare, se $q_0(x) \equiv 1$, allora $q_0(x_k) = 1$ per $k = 0, 1, ..., n$

$$\Rightarrow L_n(x) = \sum_{k=0}^{n} \underbrace{q_0(x_k)}_{1}\ \ell_k(x) = q_0(x) = 1$$

$$\boxed{\sum_{k=0}^{n} \ell_k(x) = 1}$$

Errore nell'interpolazione polinomiale

Tabella: $\{x_i, y_i\} \quad i = 0, \ldots, n$

Errore di troncamento: causato dall'aver **approssimato** la funzione con il polinomio interpolatore

Errore di propagazione: dovuto agli **errori sui dati** di input (errori di misura o di arrotondamento)

Errore totale

polinomio "ideale"

$$E_t(x) = f(x) - p_n(x) = \underbrace{f(x) - p_n^*(x)}_{\text{errore di troncamento}} + \underbrace{p_n^*(x) - p_n(x)}_{\text{errore di propagazione}} =$$

$$= E_n(x) + E_n^*(x)$$

Espressione dell'errore di troncamento

Polinomio "ideale": $p_n^*(x) = \sum_{k=0}^{n} f(x_k)\ \ell_k(x)$

Errore di troncamento: $E_n(x) = f(x) - p_n^*(x)$

- L'**errore di troncamento** deve essere **nullo nei nodi** poiché il polinomio interpolatore soddisfa le condizioni di interpolazione:
  $p_n^*(x_i) = f(x_i) \Rightarrow E_n(x_i) = f(x_i) - p_n^*(x_i) = 0 \qquad i = 0, \ldots, n$
- L'**errore di troncamento** deve essere **nullo** se $f(x)$ è un **polinomio di grado** $\le n$

Polinomio nodale: $\pi_n(x) := \prod_{i=0}^{n} (x - x_i) = (x - x_0)(x - x_1) \cdots (x - x_n)$

Errore di troncamento: $\boxed{E_n(x) = \pi_n(x) \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \qquad \xi(x) \in [x_0, x_n]}$

Limitazioni dell'errore di troncamento - 1

Errore di troncamento: $\boxed{E_n(x) = \pi_n(x)\frac{f^{(n+1)}(\xi(x))}{(n+1)!} \qquad \xi(x) \in [a,b]}$

Il punto $\xi(x)$ in cui calcolare la derivata **non è noto** $\rightarrow$ la formula dell'errore di troncamento ci permette di ricavare delle limitazioni

Limitazioni per eccesso e per difetto

$$m \le f^{(n+1)}(x) \le M \qquad x \in [a,b]$$

$$\Downarrow$$

$$\frac{m\,\pi_n(x)}{(n+1)!} \le E_n(x) \le \frac{M\,\pi_n(x)}{(n+1)!} \qquad \text{se } \pi_n(x) > 0$$

$$\frac{m\,\pi_n(x)}{(n+1)!} \ge E_n(x) \ge \frac{M\,\pi_n(x)}{(n+1)!} \qquad \text{se } \pi_n(x) < 0$$

Limitazioni dell'errore di troncamento - 2

Limitazione sul modulo

$$|f^{(n+1)}(x)| \le \mu \qquad x \in [a,b]$$

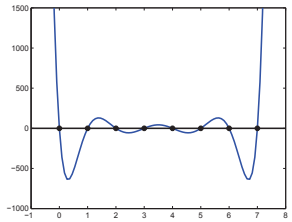
$$\Downarrow$$

$$|E_n(x)| \le \frac{\mu}{(n+1)!}|\pi_n(x)|$$

Esempio: nodi equispaziati $x_i = x_0 + ih \quad (i = 0, \ldots, n)$

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

Grafico di $\pi_7(x)$



Errore di propagazione

Errore sui dati: $\begin{cases} \epsilon_i = f(x_i) - y_i & \text{errore di arrotondamento} \\ \epsilon_i = \delta_i & \text{errore di misura} \end{cases}$

$$\downarrow$$

$$E_n^*(x) = p_n^*(x) - p_n(x) = \sum_{i=0}^{n} f(x_i)\ \ell_i(x) - \sum_{i=0}^{n} (f(x_i) + \epsilon_i)\ \ell_i(x)$$

$$\Rightarrow \boxed{E_n^*(x) = -\sum_{i=0}^{n} \epsilon_i\ \ell_i(x)}$$

Se $|\epsilon_i| \le \epsilon$ si ottiene la limitazione $\boxed{|E_n^*(x)| \le \epsilon \sum_{i=0}^{n} |\ell_i(x)| = \epsilon\,\Lambda(x)}$

Funzione di Lebesgue: $\boxed{\Lambda(x) = \sum_{i=0}^{n} |\ell_i(x)|}$ rappresenta il **coefficiente di amplificazione** degli errori sui dati

Costante di Lebesgue: $\boxed{\Lambda_n = \max_{a \le x \le b} \Lambda(x)}$

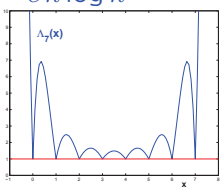
Proprietà della funzione di Lebesgue

- $\Lambda(x)$ dipende solo dai polinomi fondamentali di Lagrange e, quindi, dalla **distribuzione dei nodi**
- Poiché $\sum_{i=0}^{n} \ell_i(x) = 1 \;\Rightarrow\; \Lambda(x) \ge 1$

Nodi equispaziati in $[a,b]$

$x_i = a + ih \quad h = \frac{b-a}{n}, \quad i = 0, \ldots, n$

$\Lambda_n \rightarrow \frac{2^{n+1}}{e\,n\log n}$ per $n \rightarrow \infty$

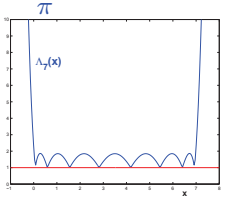


$n = 7$ Intervallo: $[0,7]$

Nodi di Chebyshev in $[a,b]$

$x_i = \frac{b-a}{2}\cos\frac{2i+1}{n+1}\frac{\pi}{2} + \frac{b+a}{2}$, $i = 0, \ldots, n$

$\Lambda_n \rightarrow \frac{2}{\pi}\log n$ per $n \rightarrow \infty$



$n = 7$ Intervallo: $[0,7]$

Esercizio

Data la tabella

i	0	1	2	3
x_i	0.0	0.4	0.8	1.2
y_i	0.00000	0.42839	0.74210	0.91031

a) scrivere l'espressione del **polinomio interpolatore di Lagrange**;

b) sapendo che $0 < f^{(4)}(x) \leq 5.657$, dare una limitazione per eccesso e una per difetto dell'**errore di troncamento** nei punti $t_1 = 0.2$, $t_2 = 0.6$, $t_3 = 1.0$;

c) valutare la **costante di Lebesgue** e dare una stima dell'errore di propagazione.

Traccia della soluzione

a) La tabella contiene **4 nodi**, quindi il **polinomio interpolatore** è di **grado 3**.

$$\ell_0(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} = -\frac{1}{0.384}(x-0.4)(x-0.8)(x-1.2)$$

$$\ell_1(x) = \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} = \frac{1}{0.128}x(x-0.8)(x-1.2)$$

$$\ell_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} = -\frac{1}{0.128}x(x-0.4)(x-1.2)$$

$$\ell_3(x) = \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} = \frac{1}{0.384}x(x-0.4)(x-0.8)$$

$$\Rightarrow \quad p_3(x) = \sum_{k=0}^{3} y_k\,\ell_k(x) \quad \Rightarrow \quad \begin{cases} p_3(t_0) = p_3(0.2) = \sum_{k=0}^{3} y_k\,\ell_k(0.2) = 0.226606 \\ p_3(t_1) = p_3(0.6) = \sum_{k=0}^{3} y_k\,\ell_k(0.6) = 0.601508 \\ p_3(t_2) = p_3(1.0) = \sum_{k=0}^{3} y_k\,\ell_k(1.0) = 0.846320 \end{cases}$$

b) Errore di troncamento: $E_3(x) = \dfrac{\pi_3(x)}{4!} f^{(4)}(\tau) \qquad \tau \in (0.0, 1.2)$

$$\pi_3(x) = (x-x_0)(x-x_1)(x-x_2)(x-x_3) = x(x-0.4)(x-0.8)(x-1.2)$$

$$\begin{cases} \pi_3(t_0) = \pi_3(0.2) = -0.024 \\ \pi_3(t_1) = \pi_3(0.6) = 0.0144 \\ \pi_3(t_2) = \pi_3(1.0) = -0.024 \end{cases} \qquad m = 0 < f^{(4)}(x) \leq M = 5.657$$

Stima di $E_3(t_0)$ $\rightarrow$ $-0.00566 \simeq \dfrac{\pi_3(t_0)}{4!}M \leq E_3(t_0) \leq \dfrac{\pi_3(t_0)}{4!}m < 0$

Stima di $E_3(t_1)$ $\rightarrow$ $0 < \dfrac{\pi_3(t_1)}{4!}m \leq E_3(t_1) \leq \dfrac{\pi_3(t_1)}{4!}M \simeq 0.00339$

Stima di $E_3(t_2)$ $\rightarrow$ $-0.00566 \simeq \dfrac{\pi_3(t_2)}{4!}M \leq E_3(t_2) \leq \dfrac{\pi_3(t_2)}{4!}m < 0$

Nota. Le approssimazioni di t_0 e t_2 sono per **eccesso**, mentre l'approssimazione di t_1 è per **difetto**.
Inoltre $|E_3(t_0)| \leq 0.05$, $|E_3(t_2)| \leq 0.05$ $\rightarrow$ **un decimale esatto**, mentre $|E_3(t_1)| \leq 0.005$ $\rightarrow$ **due decimali esatti**.

c) Errore di propagazione: $|E_3^*(t-0)| \leq \varepsilon \sum_{i=0}^{3} |\ell_i(x)| = \varepsilon\,\Lambda(x)$

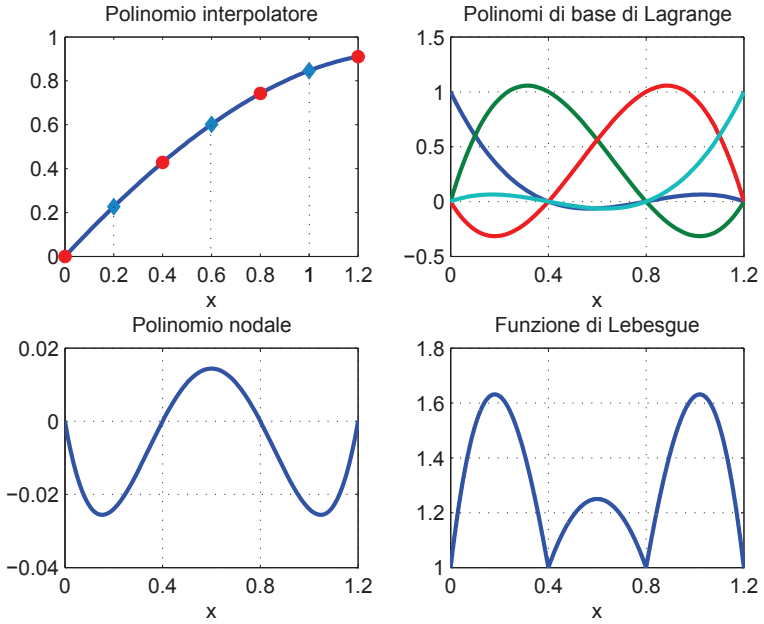
Poiché i dati di input hanno **5 decimali** l'**errore di arrotondamento** sui dati è : $|\varepsilon_i| \leq 0.5 \cdot 10^{-5} = \varepsilon$.

$$\Lambda(t_0) = \sum_{i=0}^{3} |\ell_i(t_0)| = 1.625 \rightarrow |E_3^*(t_1)| \leq 0.8 \cdot 10^{-5}$$

$$\Lambda(t_1) = \sum_{i=0}^{3} |\ell_i(t_1)| = 1.25 \rightarrow |E_3^*(t_2)| \leq 0.6 \cdot 10^{-5}$$

$$\Lambda(t_2) = \sum_{i=0}^{3} |\ell_i(t_2)| = 1.625 \rightarrow |E_3^*(x)| \leq 0.8 \cdot 10^{-5}$$

Nota. L'errore di propagazione è **trascurabile** rispetto all'errore di troncamento.



Svantaggi dell'espressione di Lagrange del polinomio interpolatore

- L'espressione di **ciascun polinomio di base di Lagrange** $l_k(x)$ dipende da **tutti** i nodi x_i.

Esempio: $\{x_0, x_1, x_2\}$

$$l_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} \qquad l_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} \qquad l_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$$

- Se si **aggiunge un nodo** bisogna **ricalcolare** tutti i polinomi l_k e, quindi, p_n.

Esempio: $\{x_0, x_1, x_2, x_3\}$

$$l_0(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} \qquad l_1(x) = \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)}$$

$$l_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} \qquad l_3(x) = \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}$$

Un'**espressione** del polinomio interpolatore p_n più efficiente è data dalla **formula di Newton alle differenze divise**.

Differenze divise

Tabella: $\{x_i, f_i\}$ $\quad i = 0, 1, \ldots, n$ (**Nodi distinti**)

Differenze divise di **ordine zero**: $\boxed{f[x_i] := f_i \qquad i = 0, 1, \ldots, n}$

Differenze divise prime: $\boxed{f[x_i, x_j] = \dfrac{f[x_i] - f[x_j]}{x_i - x_j} \quad i, j = 0, 1, \ldots, n \quad i \neq j}$

Differenze divise seconde:

$$\boxed{f[x_i, x_j, x_k] = \frac{f[x_i, x_j] - f[x_j, x_k]}{x_i - x_k} \quad i, j, k = 0, 1, \ldots, n \qquad i \neq j \neq k \neq i}$$

La **differenze divisa** k-esima relativa a $k+1$ **nodi distinti** $x_0, x_1, \ldots, x_k$ è definita da

$$\boxed{f[x_0, x_1, \ldots, x_k] = \frac{f[x_0, x_1, \ldots, x_{k-1}] - f[x_1, x_2, \ldots, x_k]}{x_0 - x_k}}$$

Tavola delle differenze divise

- Per calcolare la **differenza divisa** $k-esima$ servono $k+1$ valori della funzione
- Con $n+1$ **nodi** si possono costruire $n-k+1$ **differenze divise** $k-esime$, e pertanto **una sola** differenza divisa $n-esima$

$$\begin{array}{llllll}
x_0 & f_0 & & & & \\
 & & f[x_0, x_1] & & & \\
x_1 & f_1 & & f[x_0, x_1, x_2] & & \\
 & & f[x_1, x_2] & & f[x_0, x_1, x_2, x_3] & \\
x_2 & f_2 & & f[x_1, x_2, x_3] & & f[x_0, x_1, x_2, x_3, x_4] \\
 & & f[x_2, x_3] & & f[x_1, x_2, x_3, x_4] & \\
x_3 & f_3 & & f[x_2, x_3, x_4] & & \\
 & & f[x_3, x_4] & & & \\
x_4 & f_4 & & & &
\end{array}$$

Tavola delle differenze divise

- Se si **aggiunge** un nodo la **tavola** alle differenze divise viene modificata solo per l'**aggiunta** di una nuova riga

$$
\begin{array}{llllll}
x_0 & f_0 & & & & \\
 & & f[x_0,x_1] & & & \\
x_1 & f_1 & & f[x_0,x_1,x_2] & & \\
 & & f[x_1,x_2] & & f[x_0,x_1,x_2,x_3] & \\
x_2 & f_2 & & f[x_1,x_2,x_3] & & f[x_0,x_1,x_2,x_3,x_4] \\
 & & f[x_2,x_3] & & f[x_1,x_2,x_3,x_4] & & f[x_0,x_1,x_2,x_3,x_4,x_5] \\
x_3 & f_3 & & f[x_2,x_3,x_4] & & f[x_1,x_2,x_3,x_4,x_5] \\
 & & f[x_3,x_4] & & f[x_2,x_3,x_4,x_5] & \\
x_4 & f_4 & & f[x_3,x_4,x_5] & & \\
 & & f[x_4,x_5] & & & \\
x_5 & f_5 & & & &
\end{array}
$$

Forma di Newton del polinomio interpolatore

L'espressione del **polinomio interpolatore** nella base dei **polinomi nodali**

$$\{1, (x-x_0), (x-x_0)(x-x_1), \ldots, (x-x_0)(x-x_1)\cdots(x-x_{n-1})\}$$

è data dal **polinomio di Newton alle differenze divise**:

$$
\begin{aligned}
P_n(f;x) &= f[x_0] + (x-x_0)f[x_0,x_1] + (x-x_0)(x-x_1)f[x_0,x_1,x_2] + \\
&\quad + \cdots + (x-x_0)(x-x_1)\cdots(x-x_{n-1})f[x_0,x_1,\ldots,x_n] \equiv p_n(x)
\end{aligned}
$$

Errore di troncamento

$$E_n(x) = \pi_n(x)\, f[x, x_0, x_1, \ldots, x_n, x_{n+1}]$$

Espressione dell'errore di troncamento

$$E_n(x) = \pi_n(x)\frac{f^{(n+1)}(\xi(x))}{(n+1)!} = \pi_n(x)\, f[x, x_0, x_1, \ldots, x_n]$$

$\Rightarrow$ Se $f \in C^k[x_0, x_k]$ $\Rightarrow$ $\boxed{f[x_0, x_1, \ldots, x_k] = \dfrac{f^{(k)}(t_k)}{k!} \qquad t_k \in [x_0, x_k]}$

Stima dell'errore di troncamento

Se $f^{(n+1)}(x)$ esiste $\Rightarrow$ $f[x_0, x_1, \ldots, x_{n+1}] = \dfrac{f^{(n+1)}(t_{n+1})}{(n+1)!}$ $t_{n+1} \in [x_0, x_{n+1}]$

Se $f^{(n+1)}(x)$ **varia poco** in $[x_0, x_n]$ $\Rightarrow$ $f^{(n+1)}(\xi) \simeq f^{(n+1)}(t_{n+1})$

$$
\left.
\begin{aligned}
E_n(x) &= \pi_n f[x, x_0, x_1, \ldots, x_n] = \\
&= \pi_n(x)\frac{f^{(n+1)}(\xi)}{(n+1)!} \simeq \pi_n(x)\frac{f^{(n+1)}(t_{n+1})}{(n+1)!} \\
&= \pi_n(x)\, f[x_0, x_1, \ldots, x_n, x_{n+1}]
\end{aligned}
\right\} \Rightarrow \boxed{E_n(x) \simeq \pi_n(x)\, f[x_0, x_1, \ldots, x_n, x_{n+1}]}
$$

Propagazione degli errori nella tavola alle differenze

Il calcolo delle differenze divise è **molto sensibile** agli **errori sui dati**

Esempio:

$$
\begin{array}{llllll}
x_0 & f_0 & & & & \\
 & & f[x_0,x_1] & & & \\
x_1 & f_1 & & f[x_0,x_1,x_2] + \frac{\epsilon}{2h^2} & & \\
 & & f[x_2,x_1] + \frac{\epsilon}{h} & & f[x_0,x_1,x_2,x_3] - \frac{\epsilon}{2h^3} & \\
x_2 & f_2 + \epsilon & & f[x_1,x_2,x_3] - \frac{\epsilon}{h^2} & & f[x_0,x_1,x_2,x_3,x_4] + \frac{\epsilon}{4h^4} \\
 & & f[x_3,x_2] - \frac{\epsilon}{h} & & f[x_1,x_2,x_3,x_4] + \frac{\epsilon}{2h^3} & \\
x_3 & f_3 & & f[x_2,x_3,x_4] + \frac{\epsilon}{2h^2} & & \\
 & & f[x_4,x_3] & & & \\
x_4 & f_4 & & & &
\end{array}
$$

- L'**amplificazione** della perturbazione nel calcolo delle differenze divise è tanto **maggiore** quanto maggiore è l'**ordine** della differenza divisa che si vuole calcolare
- Poiché le differenze divise al crescere dell'ordine hanno **valori** via via più **piccoli**, si possono verificare fenomeni di **cancellazione numerica**

Differenze divise: function Matlab

```
function[out1,out2] = poldifdiv(xnodi,fnodi,xeval)
% Polinomio interpolatore alle differenze divise:
%             [pn,tavdif] = poldifdiv(xnodi,fnodi,xeval)
%
% Input:
% xnodi(1:nnodi): coordinate dei nodi
% fnodi(1:nnodi): valori nei nodi della funzione da interpolare
% xeval(i:m): coordinate deipunti in cui calcolare il polinomio interpolatore
%
% Output:
% pn(1:m): valore del polinomio interpolatore calcolato in xeval
% tavdif(1:nnodi,1:nnodi): tavola delle differenza divise
%
% Usa la function difdiv
%

% Calcolo della tavola alle differenze divise
     m_difdiv = difdiv(xnodi, fnodi);
% Calcolo del polinomio interpolatore
     nnodi = length(fnodi);
     pn = m_difdiv(1,1);
     pnod = 1;      %polinomi nodali
     for i=1:nnodi-1
       pnod = pnod.*(xeval-xnodi(i));
       pn = pn + pnod*m_difdiv(1,i+1);
     end
% Output
     out1 = pn;
     out2 = m_difdiv;
```



```
function[out1] = difdiv(xnodi, f)
%
% m_difdiv = DIFDIV(xnodi, fnodi)
%
% Calcolo della tavola alle differenze divise a partire
% dai valori dei nodi e dalla funzione nei nodi (memorizzati nella
% prima colonna della matrice m_difdiv).
%
% Input:
% - xnodi(1:nnodi+1): coordinate dei nodi
% - fnodi(1:nnodi+1): valori della funzione nei nodi
% Output:
% - difdiv(1:nnodi+1,1:kord+1): tavola delle differenze divise (per colonne)
%                               nella prima colonna ci sono i valori fnodi
%
%
      nnodi=length(xnodi);
      kord = nnodi-1;
      m_difdiv = zeros(nnodi,kord+1);
      m_difdiv(:,1) = f';
      for j = 1:kord,
         for i = 0:nnodi-1-j,
            m_difdiv(i+1,j+1) = (m_difdiv(i+1,j)-m_difdiv(i+2,j))/(xnodi(i+1)-xnodi(i+j+1));
         end
      end
%
      out1 = m_difdiv;
```


Differenze divise: programma Fortran

```
      program poldifdi
*
* Calcolo del polinomio interpolatore alle differenze divise.
*
* Input:
* - nnodi: numero di nodi (<20)
* - xnodi(0:nnodi): ascissa dei nodi
* - fnodi(0:nnodi): ordinata dei nodi
* - x: punto in cui si vuole calcolare il polinomio interpolatore
* Output:
* - difdiv(0:nnodi,0:nnodi): tavola delle differenze divise
*       (memorizzate per colonne)
* - polintx: valore del polinomio interpolatore nel punto x
*
      parameter (nmax=20)
      double precision xnodi(0:nmax), fnodi(0:nmax)
      double precision difdiv(0:nmax,0:nmax)
      double precision x, polintx
      integer nnodi, kord, i, j
      logical distinti, puntidis
```



```
* Lettura dei dati di input
*
      open (10, file='valnodi.dat')
         read (10,*) nnodi
         if (nnodi .gt. nmax) stop 'numero di nodi troppo grande'
         read (10,*) (xnodi(i),fnodi(i),i=0,nnodi)
      close (10)
*
* Verifica se i nodi sono distinti
*
      distinti = puntidis( nnodi+1, xnodi)
      if ( .not. distinti ) stop 'i nodi non sono distinti'
*
* Costruzione della tavola alle differenze divise (fino all'ordine nnodi)
*
      open (10, file='diffdiv.dat')
      do i = 0, nnodi
         difdiv(i,0) = fnodi(i)
      enddo
      kord = nnodi
      call diffdiv( nmax, nnodi, xnodi, difdiv, kord)
      write (10,*) 'Tavola alle differenze divise'
      do i = 0, nnodi
         write (10,100) xnodi(i), (difdiv(i,j),j=0,nnodi-i)
      enddo
100   format (2x,f6.2,5(2x,e12.6))
```

```
*
* Calcolo del polinomio interpolatore
*
*      polintx = polhorn( nnodi, xnodi, difdiv, x)
*      write (10,110) x, polintx
110    format (2x,'Il polonomio interpolatore in x=',f8.5,
     &         '  vale:',e12.6)
*
* Fine
*
      close(10)
      stop
      end
```



```
      logical function puntidis(n,xn)
*
* Verifica se gli n punti xn sono distinti
*
* Input:
* - xn(n): coordinate dei punti
* Output:
* - puntidis: e' .false. se i punti non sono distinti
*
      double precision xn(n)
*
      puntidis = .true.
      do i = 1, n
         do j = i+1, n
            if (xn(i).eq.xn(j)) then
               puntidis = .false.
              return
            endif
         enddo
      enddo
*
      end
```



```
      subroutine diffdiv( nmax, nnodi, xnodi, difdiv, kord)
* Calcolo della tavola alle differenze divise a partire
* dai valori dei nodi e dalla funzione nei nodi (memorizzati nella
* prima colonna della matrice difdiv.
*
* Input:
* - nnodi: numero di nodi
* - xnodi(0:nnodi): coordinate dei nodi
* - difdiv(0:nnodi,j=0): valori della funzione nei nodi
* - kord: ordine delle differenze a cui arrivare (se kord>nnodi si pone kord=nnodi)
* Output:
* - difdiv(0:nnodi,0:kord): tavola delle differenze divise (per colonne)
*
      double precision xnodi(0:nnodi), difdiv(0:nmax,0:nmax)
      integer nnodi, kord, i, j
*
      kord = min(kord,nnodi)
      do j = 1, kord
         do i = 0, nnodi-j
          difdiv(i,j) = (difdiv(i,j-1)-difdiv(i+1,j-1))
     &                  /(xnodi(i)-xnodi(i+j))
         enddo
      enddo
*
      return
      end
```


Convergenza dei polinomi interpolatori

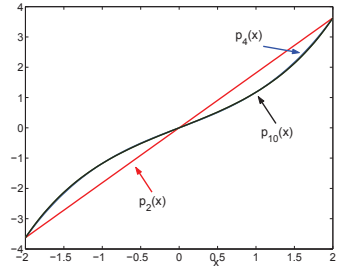
Si ha **convergenza** se, in **assenza** di **errori di arrotondamento**,

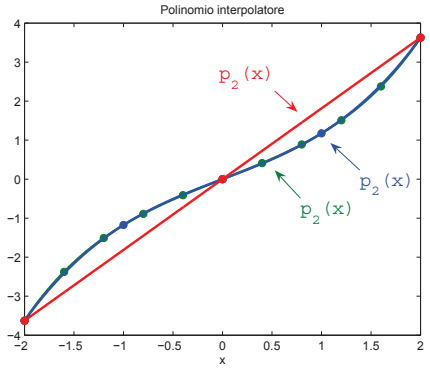
$$\lim_{n\to\infty} p_n(x) = f(x)$$

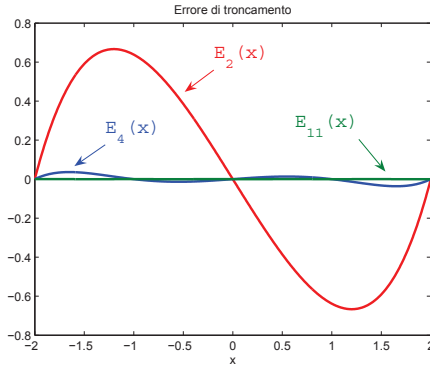
cioè se all'**aumentare del numero dei nodi** il polinomio interpolatore approssima sempre **meglio** la funzione.

Esempio 1: $f(x) = \sinh(x) \quad x \in [-2, 2]$

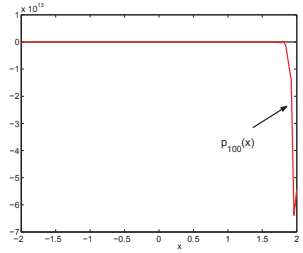
All'aumentare di n il polinomio interpolatore approssima sempre meglio la funzione. Per $n = 10$ il grafico di $p_{10}(x)$ coincide con il grafico di $f(x)$.





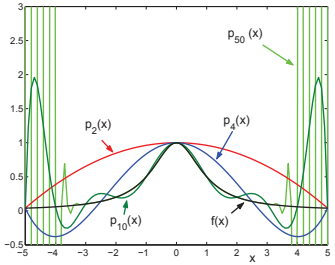


Nota. All'aumentare di n aumentano gli **errori di arrotondamento** che possono distruggere il risultato.



Esempio 2: $f(x) = \dfrac{1}{1+x^2}$ $x \in [-5,5]$ (Funzione di Runge)

All'aumentare di n il polinomio interpolatore **oscilla** sempre di più in prossimità dei bordi (non dipende dagli errori di arrotondamento)



Teoremi di convergenza - 1

Teorema 1. Se $f \in C^\infty[a,b]$ e $\lim\limits_{k\to\infty} \dfrac{(b-a)^k}{k!} \max\limits_{a\le x\le b} |f^{(k)}(x)| = 0$, allora $\lim\limits_{n\to\infty} p_n(x) = f(x)$ uniformemente in $[a,b]$.

Nota. Le funzioni con derivata equilimitata in $[a,b]$ soddisfano le ipotesi del teorema. Esempi: $\sin x$, $\cos x$, e^x.

Teorema 2. Sia $F(z)$ l'estensione di $f(x)$ nel piano complesso. Se $F(z)$ è **olomorfa** in una regione A del piano complesso contenente $[a,b]$ e, detta d la **distanza** di ∂A da $[a,b]$,

$$d > b - a$$

allora $\lim\limits_{n\to\infty} p_n(x) = f(x)$ uniformemente in $[a,b]$.

Nota. Per l'esempio di Runge, la funzione $F(z) = \dfrac{1}{1+z^2}$ ha due **singolarità** in $\pm i$. Per l'intervallo $[-5,5]$ si ha $d = 1 < b - a = 10$, quindi **non** si ha convergenza. Se si considera invece l'intervallo $[2,3]$ si ha convergenza poiché in questo caso $d = \sqrt{5} > b - a = 1$.

Polinomi di Chebyshev

Si possono dare delle condizioni di convergenza con ipotesi meno restrittive su $f(x)$ se si scelgono nodi di interpolazione x_i **particolari**.

Ponendo $x = \cos\theta$ si definisce

$$T_n(x) := \cos n\theta = \cos n(\arccos x) \qquad n = 0, 1, \ldots$$

Ricordando che $2\cos\theta\cos n\theta = \cos(n+1)\theta + \cos(n-1)\theta$

$$\Rightarrow \begin{cases} T_0(x) = 1 \qquad T_1(x) = x \\ T_{n+1}(x) = 2x\,T_n(x) - T_{n-1}(x) \\ \text{(\textbf{Relazione di ricorrenza})} \end{cases} \Rightarrow \text{per ogni } n,\ T_n(x) \text{ è un } \textbf{polinomio} \text{ di grado } n$$

I primi polinomi del sistema $\{T_n(x)\}$ sono

$$T_0(x) = 1 \quad T_1(x) = x \quad T_2(x) = 2x^2 - 1 \quad T_3(x) = 4x^3 - 3x$$

Nodi di Chebyshev

Per ogni n i **nodi di Chebyshev** sono gli $n+1$ **zeri** del polinomio di Chebyshev di grado $n+1$.

Dalla definizione di T_n si ottiene immediatamente

$$T_{n+1}(x) = 0 \iff (n+1)\theta = (2\,i+1)\frac{\pi}{2} \qquad i = 0, 1, ..., n$$

- **Nodi di Chebyshev** dell'intervallo $[-1, 1]$

$$x_i^C = \cos\left(\frac{2i+1}{n+1} \cdot \frac{\pi}{2}\right) \qquad i = 0, 1, \ldots, n$$

- **Nodi di Chebyshev** dell'intervallo $[a, b]$

Cambiamento di coordinate: $x = \dfrac{b-a}{2}t + \dfrac{a+b}{2}$

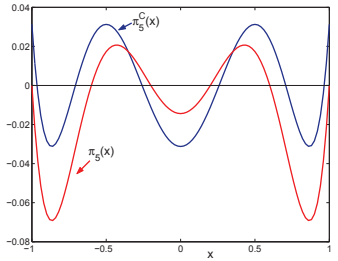
$$x_i^C = \frac{b-a}{2}\cos\left(\frac{2i+1}{n+1} \cdot \frac{\pi}{2}\right) + \frac{b+a}{2} \qquad i = 0, 1, \ldots, n$$

Teoremi di convergenza - 2

Teorema 3. Se f è **lipschitziana** in $[a, b]$, la successione dei polinomi interpolatori sui nodi di Chebyschev **converge** a f uniformemente in $[a, b]$.

Il **polinomio nodale** $\pi_n^C(x)$ costruito sui nodi di Chebyschev ha le seguenti proprietà :

- $\max\limits_{x\in[a,b]} |\pi_n^C(x)| = \dfrac{(b-a)^{n+1}}{2^{2n+1}}$
- $\max\limits_{x\in[a,b]} |\pi_n(x)| > \max\limits_{x\in[a,b]} |\pi_n^C(x)|$



Nota. I **nodi di Chebyshev** sono tutti **interni** all'intervallo $[a, b]$

$$a < x_i^C < b \qquad i = 0, 1, \ldots, n$$

Funzioni splines

Definizione. Si definisce **partizione** Δ di un intervallo $[a, b]$ un insieme di punti x_i, $i = 0, 1, \ldots, n$ tali che

$$\Delta : a = x_0 < x_1 < x_2 < \ldots < x_{n-1} < x_n = b$$

Definizione. Si definisce **funzione spline** di grado m associata alla **partizione** Δ di $[a, b]$ una funzione $S_m(x)$ tale che

- $S_m(x)$ è un **polinomio** di grado m in ciascun **sottointervallo** $T_i = [x_{i-1}, x_i]$, $i = 1, 2, \ldots, n$
- $S_m(x) \in C^{m-1}[a, b]$

Nota. $S_m(x)$ è un **polinomio** per $x \in (x_{i-1}, x_i) \Longrightarrow S_m(x) \in C^\infty(x_{i-1}, x_i)$

$S_m^{(k)}(x_i^-) = S_m^{(k)}(x_i^+)$ per $k = 0, 1, \ldots, m-1 \Longrightarrow$ la spline è '*liscia*'

Splines lineari interpolanti

- Una **spline lineare** $S_1(x)$ è un **polinomio lineare a tratti** nell'intervallo $[a, b]$ che nei nodi x_i deve soddisfare le condizioni di

Continuità : $S_1(x_i^+) = S_1(x_i^-) \qquad i = 0, 1, \ldots, n$

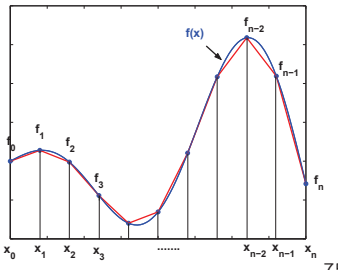
- Una **spline lineare interpolante** nei nodi x_i deve soddisfare anche le

Condizioni di interpolazione: $S_1(x_i) = y_i \qquad i = 0, 1, \ldots, n$

Per $x \in T_i = [x_{i-1}, x_i]$ $S_1(x) \in \mathbb{P}_1$. Dalle condizioni di interpolazione

$\Longrightarrow$ $S_1(x) = \dfrac{1}{h_i}\left[(x_i - x)y_{i-1} + (x - x_{i-1})y_i\right]$ $\qquad h_i = x_i - x_{i-1} \quad x \in x_i$

$\Longrightarrow$ La spline lineare interpolante **esiste** ed è **unica**



Una base per lo spazio delle splines lineari

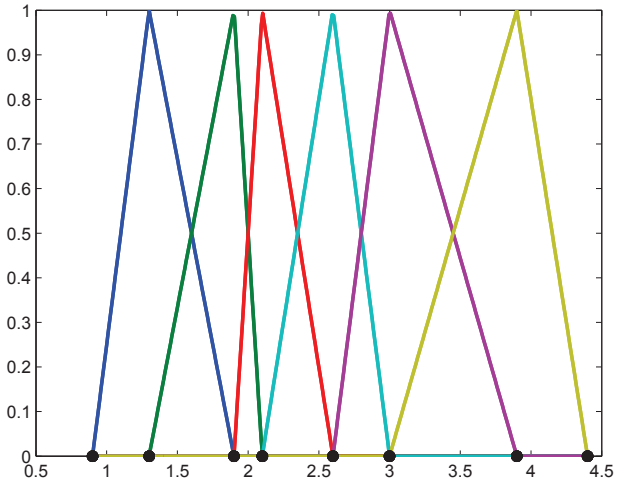
Se introduciamo una **base** (**B-splines**) $B_i(x)$, $i = 0, \ldots, n$, per lo spazio delle spline lineari tale che

$$B_i(x_j) = \begin{cases} 1 & \text{se } i = j \\ 0 & \text{se } i \neq j \end{cases} \qquad \Rightarrow S_1(x) = \sum_{i=0}^{n} y_i B_i(x) \qquad x \in [a, b]$$

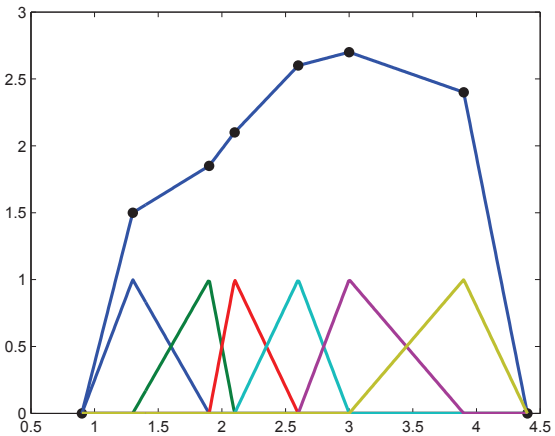
La funzione $B_i(x)$ è un polinomio lineare in T_i che soddisfa il **problema di interpolazione**: $B_i(x_j) = e_j$, $j = 0, \ldots, n$, dove $e_i = (0, 0, \ldots, \underbrace{1}_{i}, \ldots, 0)$

$$B_i(x) = \begin{cases} \dfrac{x - x_{i-1}}{h_i} & \text{se} \quad x \in [x_{i-1}, x_i] \\ \dfrac{x_{i+1} - x}{h_{i+1}} & \text{se} \quad x \in [x_i, x_{i+1}] \qquad i = 1, \ldots, n-1 \\ 0 & \text{altrove} \end{cases}$$

$$B_0(x) = \begin{cases} \dfrac{x_1 - x}{h_1} & \text{se} \quad x \in [x_0, x_1] \\ 0 & \text{altrove} \end{cases} \qquad B_n(x) = \begin{cases} \dfrac{x - x_{n-1}}{h_n} & \text{se} \quad x \in [x_{n-1}, x_n] \\ 0 & \text{altrove} \end{cases}$$



Partizione: $a = x_0 = 0.9, x_1 = 1.3, x_3 = 1.9, x_4 = 2.1,$
$x_5 = 2.6, x_6 = 3.0, x_7 = 3.9, b = x_8 = 4.4$



$y_i = 0, 1.5, 1.85, 2.1, 2.6, 2.7, 2.4, 0$

Spline lineare interpolante: $S_1(x) = \sum_{i=0}^{7} y_i B_i(x)$

Splines cubiche interpolanti

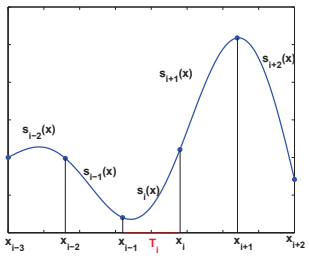
Splines cubiche ($m = 3$): $S_3^{(k)}(x_i^-) = S_3^{(k)}(x_i^+) \quad k = 0, 1, 2$

Condizioni di interpolazione: $S_3(x_i) = y_i \qquad i = 0, 1, \ldots, n$

Costruzione della spline cubica interpolante

$S_3(x)$ è un **polinomio** di grado 3 in ogni **sottointervallo** $T_i = [x_{i-1}, x_i]$, $i = 1, 2, \ldots, n$:

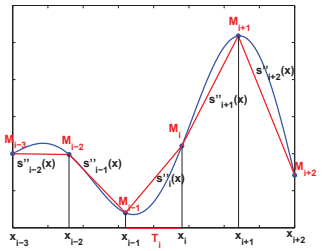
$$S_3(x)|_{x \in T_i} = s_i(x) \in \mathbb{P}_3$$



La **derivata seconda** di $S_3(x)$ è una funzione **lineare a tratti**:

$$S_3''(x)|_{x\in T_i} = s_i''(x) = \frac{1}{h_i}\Big[(x_i - x)M_{i-1} + (x - x_{i-1})M_i\Big] \qquad h_i = x_i - x_{i-1}$$

$$\Rightarrow \quad s_i''(x_{i-1}) = M_{i-1},\ s_i''(x_i) = M_i \ \Rightarrow\ S_3''(x_i) = M_i \qquad i = 0, 1, ..., n$$



Nota. Le costanti M_i sono **incognite** da determinare e **individuano** la spline.

Poichè $S_3''(x)$ è **continua** si può **integrare** due volte per ottenere $S_3(x)$:

$$S_3'(x)|_{x\in T_i} = s_i'(x) = \int_{T_i} s_i''(x)dx =$$

$$= \frac{1}{2h_i}\Big[-(x_i - x)^2 M_{i-1} + (x - x_{i-1})^2 M_i\Big] + C_i$$

$$S_3(x)|_{x\in T_i} = s_i(x) = \int_{T_i} s_i'(x)dx =$$

$$= \frac{1}{6h_i}\Big[(x_i - x)^3 M_{i-1} + (x - x_{i-1})^3 M_i\Big] + C_i(x - x_{i-1}) + D_i$$

Le **costanti di integrazione** C_i e D_i e le **incognite** M_i **individuano** la spline cubica interpolante e si **determinano** imponendo che $S_3(x)$ soddisfi:

i) le **condizioni di interpolazione**: $S_3(x_i) = y_i \qquad i = 0, 1, \ldots, n$

ii) la **continuità della derivata prima** nei nodi interni:

$$S_3'(x_i^-) = S_3'(x_i^+) \ \Rightarrow\ s_{i+1}'(x_i) = s_i'(x_i) \qquad i = 1, \ldots, n-1$$

Le **condizioni di interpolazione** danno

$$\begin{cases} y_i = s_i(x_i) = \dfrac{1}{6h_i}\underbrace{(x_i - x_{i-1})^3}_{h_i^3} M_i + \underbrace{(x_i - x_{i-1})}_{h_i} C_i + D_i = \dfrac{1}{6}h_i^2 M_i + h_i C_i + D_i \\ y_{i-1} = s_i(x_{i-1}) = \dfrac{1}{6h_i}\underbrace{(x_i - x_{i-1})^3}_{h_i^3} M_{i-1} + D_i = \dfrac{1}{6}h_i^2 M_{i-1} + D_i \end{cases}$$

$$\Downarrow$$

$$\begin{cases} C_i = \dfrac{1}{h_i}(y_i - y_{i-1}) - \dfrac{h_i}{6}(M_i - M_{i-1}) \\ D_i = y_{i-1} - \dfrac{h_i^2}{6}M_{i-1} \end{cases} \qquad i = 1, 2, \ldots, n$$

$$\boxed{\begin{aligned} S_3(x)|_{x\in T_i} &= s_i(x) = \frac{1}{6h_i}\Big[(x_i - x)^3 M_{i-1} + (x - x_{i-1})^3 M_i\Big] + \\ &+ \left[\frac{1}{h_i}(y_i - y_{i-1}) - \frac{h_i}{6}(M_i - M_{i-1})\right](x - x_{i-1}) + y_{i-1} - \frac{h_i^2}{6}M_{i-1} \end{aligned}}$$

La **continuità della derivata prima** dà

$$s_{i+1}'(x_i) = s_i'(x_i) \qquad i = 1, \ldots, n-1$$

$$\Downarrow$$

$$-\frac{1}{2h_{i+1}}\underbrace{(x_{i+1} - x_i)^2}_{h_{i+1}^2} M_i + C_{i+1} = \frac{1}{2h_i}\underbrace{(x_i - x_{i-1})^2}_{h_i^2} M_i + C_i$$

$$\Downarrow$$

$$-\frac{1}{2}h_{i+1}M_i + \frac{1}{h_{i+1}}(y_{i+1} - y_i) - \frac{h_{i+1}}{6}(M_{i+1} - M_i) = \frac{1}{2}h_i M_i + \frac{1}{h_i}(y_i - y_{i-1}) - \frac{h_i}{6}(M_i - M_{i-1})$$

$$\Downarrow$$

$$\frac{h_i}{6}M_{i-1} + \frac{h_i + h_{i+1}}{3}M_i + \frac{h_{i+1}}{6}M_{i+1} = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}$$

Per $i = 1, 2, \ldots, n-1$ le relazioni

$$\frac{h_i}{6}M_{i-1} + \frac{h_i + h_{i+1}}{3}M_i + \frac{h_{i+1}}{6}M_{i+1} = \frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}$$

formano un **sistema lineare** di $n-1$ **equazioni** nelle $n+1$ **incognite** $M_0, M_1, \ldots, M_n$ $\Rightarrow$ restano **due parametri** liberi.

Spline cubiche naturali

Si pone $M_0 = M_n = 0$, il che equivale a richiedere che la spline sia **lineare** al di fuori dell'intervallo $[a, b]$.

Spline cubiche naturali

$$M_0 = M_n = 0$$

$$\underbrace{\begin{pmatrix} \frac{h_1+h_2}{3} & \frac{h_2}{6} & 0 & 0 & \cdots & 0 \\ \frac{h_2}{6} & \frac{h_2+h_3}{3} & \frac{h_3}{6} & 0 & \cdots & 0 \\ 0 & \frac{h_3}{6} & \frac{h_3+h_4}{3} & \frac{h_4}{6} & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 0 & \frac{h_{n-1}}{6} & \frac{h_{n-1}+h_n}{3} \end{pmatrix}}_{C} \underbrace{\begin{pmatrix} M_1 \\ M_2 \\ \vdots \\ M_{n-2} \\ M_{n-1} \end{pmatrix}}_{M} = \underbrace{\begin{pmatrix} \frac{y_2-y_1}{h_2} - \frac{y_1-y_0}{h_1} \\ \frac{y_3-y_2}{h_3} - \frac{y_2-y_1}{h_2} \\ \vdots \\ \frac{y_{n-1}-y_{n-2}}{h_{n-1}} - \frac{y_{n-2}-y_{n-3}}{h_{n-2}} \\ \frac{y_n-y_{n-1}}{h_n} - \frac{y_{n-1}-y_{n-2}}{h_{n-1}} \end{pmatrix}}_{F}$$

La **matrice dei coefficienti** C è **simmetrica**, **tridiagonale**, **diagonalmente dominante** e ha elementi diagonali **positivi**.

$\Rightarrow$ La matrice C è **definita positiva** e, quindi, **regolare**.

$\Rightarrow$ La **spline cubica naturale interpolante** è **unica**.

Teoremi di convergenza

Teorema 1. Per ogni funzione $v \in C^2[a, b]$, passante per i punti (x_i, y_i), si ha

$$\int_a^b |S_3''(x)|^2 dx \leq \int_a^b |v''(x)|^2 dx$$

Nota. Poichè la **curvatura** di una funzione è legata alla sua **derivata seconda**, la **spline cubica naturale** è quella che ha **minima curvatura globale** tra tutte le funzioni interpolanti $v \in C^2[a, b]$.

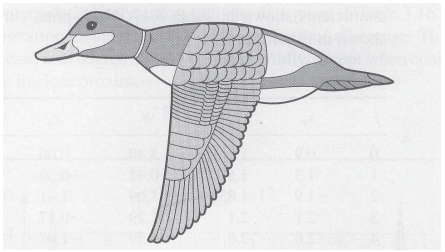
Teorema 2. Sia $f \in C^4[a, b]$, posto $h = \max_{1 \leq i \leq n} h_i$, risulta

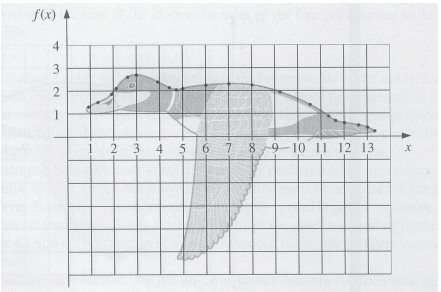
$$\max_{x \in [a,b]} |f^{(k)}(x) - S_3^{(k)}| = O(h^{4-k}), \qquad k = 0, 1, 2, 3$$

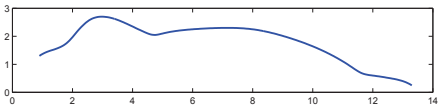
Nota. La spline cubica naturale interpolante converge alla **funzione** f e alle sue **derivate** sotto ipotesi **meno restrittive** rispetto all'interpolazione polinomiale.

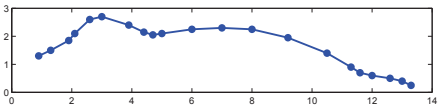
Esempio

La figura qui sotto mostra il profilo di un'anatra in volo.

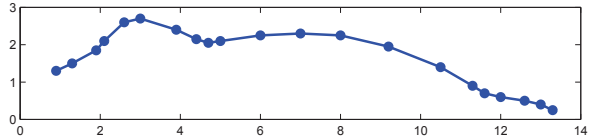








Per approssimare il profilo superiore scegliamo alcuni punti lungo la curva in modo che siano più fitti dove la curva varia più rapidamente.

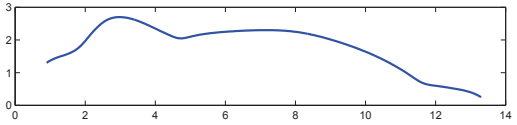


I punti scelti corrispondono alla tabella seguente.

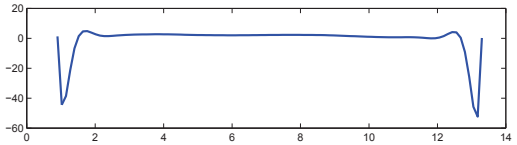
x	0.9	1.3	1.9	2.1	2.6	3.0	3.9	4.4	4.7	5.0	6.0
$f(x)$	1.3	1.5	1.85	2.1	2.6	2.7	2.4	2.15	2.05	2.1	2.25

x	7.0	8.0	9.2	10.5	11.3	11.6	12.0	12.6	13.0	13.3
$f(x)$	2.3	2.25	1.95	1.4	0.9	0.7	0.6	0.5	0.4	0.25

La **spline cubica naturale** che interpola i dati in tabella è mostrata nella figura: la spline è praticamente indistinguibile dal profilo dell'anatra.



Nella figura qui sotto è mostrato il **polinomio interpolatore** che approssima gli stessi dati: si tratta di un polinomio di grado 20 che presenta delle forti oscillazioni soprattutto in prossimità dei bordi dell'intervallo di integrazione



Approssimazione trigonometrica

Se la funzione o i dati che si vogliono approssimare hanno un andamento **periodico** si sceglie come classe di **funzioni approssimanti** l'insieme $\mathcal{T}_N$ dei **polinomi trigonometrici**.

$$\mathcal{T}_N := \left\{ T_N(x) = \sum_{k=0}^{N} \Big(a_k \cos(k\,x) + b_k \sin(k\,x) \Big), a_k, b_k \in \mathbf{R} \;\; \forall k, x \in [0, 2\pi) \right\}$$

- **Interpolazione**: si usa quando si vogliono interpolare pochi **dati periodici** non affetti da errori
- **Approssimazione discreta ai minimi quadrati**: si usa quando si vogliono interpolare molti **dati periodici** affetti da **errori**

Approssimazione ai minimi quadrati di dati discreti periodici

Tabella: $\{x_i, y_i\}$ con nodi $x_i = i\,\dfrac{2\pi}{n+1}$, $i = 0, 1, ..., n$, **equispaziati**

Funzione approssimante:

$$T_M(x) = \frac{a_0}{2} + \sum_{k=1}^{M} \Big(a_k \cos(k\,x) + b_k \sin(k\,x) \Big) \qquad \boxed{n+1 >> 2M+1}$$

Funzioni di base:

$\psi_0(x) = \frac{1}{2},\;\; \psi_1(x) = \cos(x),\;\; \psi_2(x) = \sin(x),\;\; \ldots\;\; \psi_{2M}(x) = \cos(Mx),\;\; \psi_{2M+1}(x) = \sin(Mx)$

Metodo di approssimazione: si **minimizza** lo **scarto quadratico**

$$\sigma^2(a_0, ..., a_M, b_1, ..., b_M) = \sum_{i=0}^{n} \left[\underbrace{\frac{a_0}{2} + \sum_{k=1}^{M} \Big(a_k \cos(k\,x_i) + b_k \sin(k\,x_i) \Big)}_{T_M(x_i)} - y_i \right]^2$$

Risolvere il problema dell'**approssimazione trigonometrica ai minimi quadrati** equivale a trovare i valori dei $2M+1$ coefficienti a_k, b_k che rendono minimo σ

Sistema delle equazioni normali

$\boxed{H_t\, A = B}$ dove $A = [a_0, a_1, b_1, ..., a_M, b_M]^T$

$$B = \left[\tfrac{1}{2}\sum_{i=0}^{n} y_i, \sum_{i=0}^{n} y_i \cos(x_i), \sum_{i=0}^{n} y_i \sin(x_i), ..., \sum_{i=0}^{n} y_i \cos(Mx_i), \sum_{i=0}^{n} y_i \sin(Mx_i)\right]^T$$

$$H_t = \begin{bmatrix}
\frac{1}{4}(n+1) & \frac{1}{2}\sum_{i=0}^{n}\cos(x_i) & \frac{1}{2}\sum_{i=0}^{n}\sin(x_i) & \cdots & \frac{1}{2}\sum_{i=0}^{n}\cos(Mx_i) & \frac{1}{2}\sum_{i=0}^{n}\sin(Mx_i) \\
\frac{1}{2}\sum_{i=0}^{n}\cos(x_i) & \sum_{i=0}^{n}\cos^2(x_i) & \sum_{i=0}^{n}\cos(x_i)\sin(x_i) & \cdots & \sum_{i=0}^{n}\cos(x_i)\cos(Mx_i) & \sum_{i=0}^{n}\cos(x_i)\sin(Mx_i) \\
\frac{1}{2}\sum_{i=0}^{n}\sin(x_i) & \sum_{i=0}^{n}\sin(x_i)\cos(x_i) & \sum_{i=0}^{n}\sin^2(x_i) & \cdots & \sum_{i=0}^{n}\sin(x_i)\cos(Mx_i) & \sum_{i=0}^{n}\sin(x_i)\sin(Mx_i) \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\frac{1}{2}\sum_{i=0}^{n}\cos(Mx_i) & \sum_{i=0}^{n}\cos(Mx_i)\cos(x_i) & \sum_{i=0}^{n}\cos(Mx_i)\sin(x_i) & \cdots & \sum_{i=0}^{n}\cos^2(Mx_i) & \sum_{i=0}^{n}\cos(Mx_i)\sin(Mx_i) \\
\frac{1}{2}\sum_{i=0}^{n}\sin(Mx_i) & \sum_{i=0}^{n}\sin(Mx_i)\cos(x_i) & \sum_{i=0}^{n}\sin(Mx_i)\sin(x_i) & \cdots & \sum_{i=0}^{n}\sin(Mx_i)\cos(Mx_i) & \sum_{i=0}^{n}\sin^2(Mx_i)
\end{bmatrix}$$

Condizioni di ortogonalità

Per **nodi equispaziati** in $[0, 2\pi)$ valgono le condizioni ($r \geq 0$, $s \geq 0$)

$$\sum_{i=0}^{n} \cos(rx_i)\cos(sx_i) = \begin{cases} 0 & \text{se } r \neq s \\ (n+1)/2 & \text{se } r = s \neq 0 \text{ oppure} \\ & \quad r = s \neq (n+1)/2 \text{ con } (n+1) \text{ pari} \\ n+1 & \text{se } r = s = 0 \text{ oppure} \\ & \quad r = s = (n+1)/2 \text{ con } (n+1) \text{ pari} \end{cases}$$

$$\sum_{i=0}^{n} \sin(rx_i)\sin(sx_i) = \begin{cases} 0 & \text{se } r \neq s \text{ oppure } r = s = 0 \text{ oppure} \\ & \quad r = s = (n+1)/2 \text{ con } (n+1) \text{ pari} \\ (n+1)/2 & \text{se } r = s \neq 0 \text{ oppure} \\ & \quad r = s \neq (n+1)/2 \text{ con } (n+1) \text{ pari} \end{cases}$$

$$\sum_{i=0}^{n} \sin(rx_i)\cos(sx_i) = 0 \qquad \forall\ r, s$$

Soluzione del sistema delle equazioni normali

Dalle **condizioni di ortogonalità** segue che la matrice H_t è una matrice **diagonale**.

$$H_t = diag\left(\frac{n+1}{2}, \frac{n+1}{2}, ..., \frac{n+1}{2}\right)$$

$\Rightarrow$ H_t **regolare** e **facilmente invertibile**

$\Rightarrow$
$$\begin{cases} a_k = \dfrac{2}{n+1}\displaystyle\sum_{i=0}^{n} y_i \cos(k\, x_i) & k = 0, 1, 2, ..., M \\ b_k = \dfrac{2}{n+1}\displaystyle\sum_{i=0}^{n} y_i \sin(k\, x_i) & k = 1, 2, ..., M \end{cases}$$

$\Rightarrow$
$$T_M(x) = \frac{a_0}{2} + \sum_{k=1}^{M}\left(a_k \cos(k\, x) + b_k \sin(k\, x)\right)$$

Interpolazione trigonometrica (dati dispari)

Se il numero dei dati è **uguale** al numero dei coefficienti, cioè se

$$\boxed{n + 1 = 2M + 1}$$

le formule precedenti danno l'espressione del **polinomio trigonometrico interpolante** i dati $\{x_i, y_i\}$, $i = 0, ..., 2M$,

$$T_M(x) = \frac{a_0}{2} + \sum_{k=1}^{M}\left(a_k \cos(k\, x) + b_k \sin(k\, x)\right)$$

con
$$\begin{cases} a_k = \dfrac{2}{2M+1}\displaystyle\sum_{i=0}^{2M} y_i \cos(k\, x_i) & k = 0, 1, 2, ..., M \\ b_k = \dfrac{2}{2M+1}\displaystyle\sum_{i=0}^{2M} y_i \sin(k\, x_i) & k = 1, 2, ..., M \end{cases}$$

$\Rightarrow T_M(x_i) = y_i \qquad i = 0, 1, \ldots, 2M$

Interpolazione trigonometrica (dati pari)

Se il numero dei dati è **pari**, cioè se

$$\boxed{n+1=2M}$$

l'espressione del **polinomio trigonometrico interpolante** i dati $\{x_i, y_i\}$, $i = 0, ..., 2M-1$, diventa

$$T_M(x) = \frac{a_0}{2} + \sum_{k=1}^{M-1} \left(a_k \cos(k\,x) + b_k \sin(k\,x) \right) + \frac{a_M}{2} \cos(M\,k)$$

con

$$\begin{cases} a_k = \dfrac{2}{2M} \displaystyle\sum_{i=0}^{2M-1} y_i \cos(k\,x_i) & k = 0, 1, 2, ..., M \\ \\ b_k = \dfrac{2}{2M} \displaystyle\sum_{i=0}^{2M-1} y_i \sin(k\,x_i) & k = 1, 2, ..., M-1 \end{cases}$$

$\Rightarrow T_M(x_i) = y_i \qquad i = 0, 1, \ldots, 2M-1$

Costo computazionale

I coefficienti a_k, b_k nell'approssimazione ai **minimi quadrati (discreta)** o nell'**interpolazione** costituiscono l'**Analisi di Fourier Discreta** di una funzione (segnale).

Per calcolare **ciascun coefficiente** sono necessarie $n-1$ **operazioni** (moltiplicazione+addizione).

$\Rightarrow$ Per calcolare t_N sono necessarie

$(2M+1)(n-1)$ **operazioni** (**minimi quadrati**)
$(n-1)^2$ **operazioni** (**interpolazione**)

Nelle applicazioni all'**analisi di segnali** i dati sono **molte centinaia**

$\Rightarrow$ il **costo computazionale** $C_c = \mathcal{O}(n^2)$ è dell'ordine di qualche **milione** di operazioni

$\Rightarrow$ l'accumularsi degli **errori di arrotondamento** distrugge il risultato numerico e, inoltre, il **tempo di calcolo** può essere molto elevato

Queste difficoltà sono state superate dall'algoritmo della **Fast Fourier Transform** il cui costo computazionale è $C_c = \mathcal{O}(n \log_2 n)$.

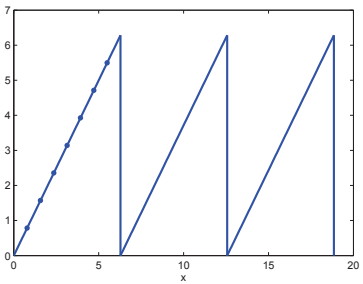
Esercizio

Costruire il polinomio interpolatore della **funzione periodica**

$$f(x) = x \qquad x \in [0, 2\pi)$$

relativo ai **nodi**

$$x_i = i\,\frac{\pi}{4} \qquad i = 0, 1, \ldots, 7$$



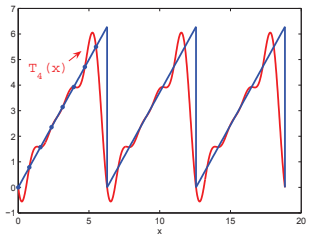
Traccia della soluzione

L'espressione del polinomio interpolatore è

$$T_4(x) = \frac{a_0}{2} + \sum_{k=1}^{3} \left(a_k \cos(k\,x) + b_k \sin(k\,x) \right) + \frac{a_4}{2} \cos(4\,x)$$

con

$$\begin{cases} a_k = \dfrac{1}{4} \displaystyle\sum_{i=0}^{7} y_i \cos(k\,x_i) & k = 0, ..., 4 \\ \\ b_k = \dfrac{1}{4} \displaystyle\sum_{i=0}^{7} y_i \sin(k\,x_i) & k = 1, ..., 3 \end{cases}$$



Polinomio trigonometrico interpolante

Polinomio trigonometrico ai minimi quadrati: function Matlab

```
function[out1,out2,out3] = polminquadtrig(xnodi,fnodi,ndeg,xeval)

% Polinomio trigonometrico ai minimi quadrati:
%             [pn,ak,bk] = polminquadtrig(xnodi,fnodi,ndeg,xeval)
%
% Input:
% xnodi(1:mnodi): coordinate dei nodi
% fnodi(1:mnodi): valori nei nodi della funzione da interpolare
% ndeg: grado del polinomio ai minimi quadrati
% xeval(i:nx): coordinate deipunti in cui calcolare il polinomio interpolatore
%
% Output:
% pn(1:nx): valore del polinomio calcolato in xeval
% ak(1:ndeg),bk(1:ndeg): tavola delle differenza divise
%
%

% Calcolo dei coefficienti
mnodi = length(xnodi);
ak0 = 2/mnodi*sum(fnodi);
for i=1:ndeg
    ak(i) = 2/mnodi*sum(fnodi.*cos(i*xnodi));
    bk(i) = 2/mnodi*sum(fnodi.*sin(i*xnodi));
end
```



```
% Calcolo del polinomio approssimatore
pn = ak0/2;
for i=1:ndeg
    pn = pn + ak(i)*cos(i*xeval)+bk(i)*sin(i*xeval);
end

% Output
out1 = pn;
out2 = [ak0 ak];
out3 = bk;
```


Riferimenti bibliografici

L. Gori, *Calcolo Numerico*:

Cap. 6 §§ 6.1-6.5, 6.10-6.13

L. Gori, M.L. Lo Cascio, F. Pitolli, *Esercizi di Calcolo Numerico*:

Cap. 3: Es. 3.1, 3.3-3.6, 3.8-3.10, 3.12-3.15, 3.18, 3.20-3.21

Esercizi d'esame: Es. 7.2, 7.26, 7.37, 7.45, 7.50, 7.70, 7.77, 7.79, 7.81, 7.83