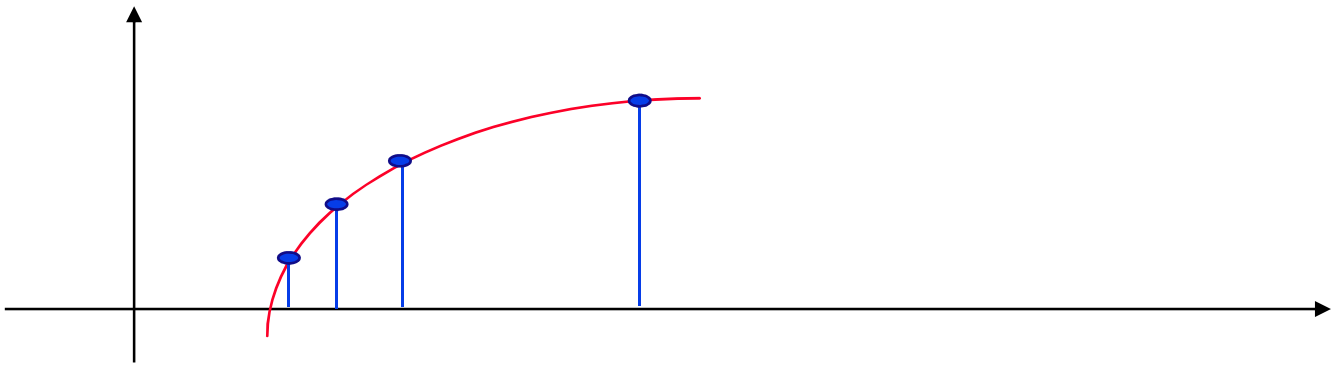
INTERPOLAZIONE

- Il problema ***dell'approssimazione di una funzione*** è di importanza fondamentale in diverse discipline dell'ingegneria

- Consiste nella ***sostituzione di una funzione nota per punti*** (*o troppo complicata*) con ***una più semplice*** (normalmente polinomi algebrici o trigonometrici)

- Esistono differenti tecniche



INTERPOLAZIONE

- Si supponga di avere la seguente tabella per alcuni valori di una funzione f(X)

X_1	X_2	X_3	...	X_N
$f(X_1)$	$f(X_2)$	$f(X_3)$	...	$f(X_N)$

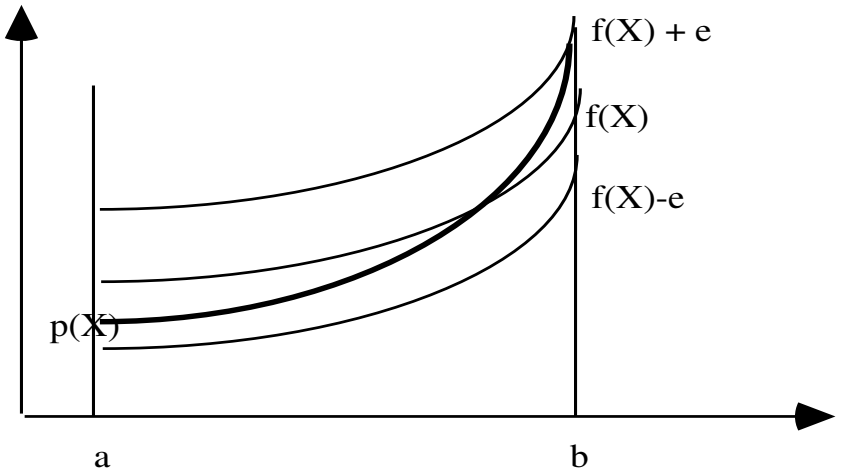
- I punti $X_1,...,X_N$ sono supposti distinti

- Si ha che per ogni *n>0* esiste ed è unico il ***polinomio $p_{n-1}(X)$ detto polinomio di interpolazione***, tale che:
  - grado di $p_{n-1}(X)$ è al più **n-1**
  - $p_{n-1}(X) = f(X_i)$ per i=1,2,...,n

- più alto il numero di punti conosciuti, più alto il grado del polinomio, migliore l'approssimazione

INTERPOLAZIONE

- Teorema di Weierstrass:



Ogni funzione f(X) continua in un intervallo (a,b) può essere rappresentata in quell'intervallo con la precisione richiesta mediante un polinomio p(X) tale che:

$$|f(X) - p(X)| <= \varepsilon$$

con ε prefissato e piccolo a piacere per ogni X

INTERPOLAZIONE CON POLINOMI

$$p_n(X) = a_0 + a_1X + a_2X^2 + \ldots + a_nX^n$$

è caratterizzato da **n+1** coefficienti (a_0, a_1, ..., a_n)

- Imponendo al polinomio di passare per gli n+1 punti dati, si ottiene il sistema:

$$f(X_0) = p_n(X_0) = a_0 + a_1X_0 + a_2X_0^2 + \ldots + a_nX_0^n$$
$$f(X_1) = p_n(X_1) = a_0 + a_1X_1 + a_2X_1^2 + \ldots + a_nX_1^n$$
$$\ldots$$
$$f(X_n) = p_n(X_n) = a_0 + a_1X_n + a_2X_n^2 + \ldots + a_nX_n^n$$

Si può dimostrare che questo sistema ha sempre ***determinante diverso da zero*** => l'unica soluzione fornisce coefficienti polinomio

POLINOMIO INTERPOLAZIONE: LAGRANGE

- Polinomio ha la forma:

$p_n(X)=L_0(X)f(X_0)+L_1(X)f(X_1)+..+L_n(X)f(X_n) \Rightarrow p_n(X)=\sum_{j=0,n} L_j(X)f(X_j)$

dove ***L_j sono polinomi di grado <=n***

- Affinché il polinomio rappresenti la curva passante per X_0, X_1,.. X_n è necessario che:

per ogni i=0,..n **$p_n(X_i)=f(X_i)$** e quindi

$L_i(X_i)=1$
$L_i(X_j)=0$ per ogni j≠i

Determinazione dei termini L_i

$L_i(X)$ deve essere un polinomio di grado minore o uguale a n che si annulla per $X = X_0, X_1, X_2,..., X_{i-1}, X_{i+1},...,X_N$

POLINOMIO INTERPOLAZIONE: LAGRANGE

- Assumiamo $L_i(X)=C_i(X-X_0)(X-X_1)\ldots(X-X_{i-1})(X-X_{i+1})\ldots(X-X_N)$

dove C_i è una costante determinata in base alla condizione $L_i(X_i) = 1$

$$C_i = \frac{1}{(X_i - X_0)(X_i - X_1)\cdots(X_i - X_{i-1})(X_i - X_{i+1})\cdots(X_i - X_n)}$$

$$\Rightarrow C_i = \frac{1}{\prod_{j=0, i\neq j}^{n} (X_i - X_j)}$$

- Quindi:

$$L_i(X) = \prod_{\substack{j=0 \\ j\neq i}}^{n} \frac{(X - X_j)}{(X_i - X_j)}$$

INTERPOLAZIONE LAGRANGE: ALGORITMO

Il calcolo di un polinomio nella forma di Lagrange in corrispondenza di un valore fissato **x** può essere effettuato come segue:

```
float lagrange (float *X, float*Y,int dim, float x) {
/*X e Y vettori coordinate dei punti noti, dim numero punti
   noti,  x valore in cui calcolare p(x) */
   float ris, L;
   int i,j;
   ris=0;
   for (i=0; i<dim; i++) {
      L=1;
      for (j=0; j<dim; j++) /* costruz. Li(x)*/
            if (j!=i)
                  L*=(x-X[j])/(X[i]-X[j]);
      ris+=L*Y[i]; /*sommo termine i-simo*/
   }
   return ris;
}
```

INTERPOLAZIONE LAGRANGE: UTILIZZO

```
#include <stdio.h>
#include <stdlib.h>
void leggi_v(float *V, int dim);
float lagrange(float *X, float*Y,int dim, float x);

main() {
   float X[N], Y[N], x, R; int d;
   printf("Quanti punti si conoscono?"); scanf("%d", &d);
   printf("Ascisse dei punti noti:  \n"); leggi_v(X,  d);
   printf("Ordinate dei punti noti:  \n"); leggi_v(Y,  d);
   printf("\nAscissa del punto da calcolare?");
   scanf("%f", &x);
   R=lagrange(X, Y, d, x);
   printf("Val. di f(%f)=%f\n", x, R);
}
void leggi_v(float *V, int dim) {
   int i;
   for (i=0; i<dim; i++) {
      printf("elemento %d? ", i); scanf("%f", &V[i]);
   }
}
```

INTERPOLAZIONE CON POLINOMI

- ***Metodo di Lagrange***
  - Metodo computazionalmente costoso: per ogni particolare valore di X abbiamo circa $\mathbf{n^2}$ operazioni
  - conveniente solo se si deve calcolare per pochi valori di X
- Esistono metodi ***meno costosi***
  - Metodo di ***Newton o delle differenze divise*** (si possono aggiungere facilmente ulteriori valori allo schema senza invalidare le computazioni precedenti)

INTERPOLAZIONE POLINOMIO: NEWTON

Caratteristiche:

- Possibile effettuare raffinamento dell'approssimazione solo aggiungendo un'ulteriore osservazione X_k, senza alterare la computazione precedente

Idea di Base:

- Si vuole approssimare f con una polinomiale $p_k(X)$ di grado ***al più k*** che passa nei punti $(X_0, f(X_0)),(X_1,f(X_1)),..$ $(X_k, f(X_k))$

❑ Polinomio può essere messo nella forma di Newton:

$$p_n(X) = a_0 + a_1(X-X_0) + a_2(X-X_0)(X-X_1) + ...$$
$$+ a_n(X-X_0)...(X-X_{n-1})$$

POLINOMIO INTERPOLAZIONE: NEWTON

Definizione:

- ***Differenza divisa*** *di ordine 1*

$$f[X_0, X_1] = \frac{f(X_1) - f(X_0)}{X_1 - X_0}$$

- Differenza divisa di ordine 2

$$f[X_0, X_1, X_2] = \frac{f[X_1, X_2] - f[X_0, X_1]}{X_2 - X_0}$$

- Procedendo per induzione: ***differenza divisa di ordine n***

$$f[X_0, X_1, \cdots, X_n] = \frac{f[X_1, X_2, \cdots, X_n] - f[X_0, X_1, \cdots, X_{n-1}]}{X_n - X_0}$$

$$= \sum_{k=0}^{n} \frac{f(X_k)}{\prod_{j=0, j\neq k}^{n} (X_k - X_j)}$$

Nota: differenza divisa è invariante rispetto alle permutazioni degli argomenti. Ad es:

$$f[X_0, X_1, X_2] = f[X_1, X_0, X_2] = f[X_2, X_1, X_0]$$

POLINOMIO INTERPOLAZIONE: NEWTON

Ogni coefficiente a_i si può ricavare sostituendo i valori X_i (i=0,...,n-1) nell'equazione:

$$p_n(X)=a_0+a_1(X-X_0)+a_2(X-X_0)(X-X_1)+\ldots$$
$$+a_n(X-X_0)\ldots(X-X_{n-1})$$

e imponendo $p(X_i) = f(X_i)$

Si trova che:

$a_0 = f(X_0)$

$a_1 = (f(X_1) - f(X_0))/(X_1-X_0) = f[X_0,X_1]$

$a_2 = (f[X_1,X_2] - f[X_0,X_1])/(X_2-X_0) = f[X_0,X_1,X_2]$

...

an = f[X0,....,Xn]

POLINOMIO INTERPOLAZIONE: NEWTON

Calcolo dell'errore

- Consideriamo di nuovo la definizione di differenza divisa:

$$f[X_0, X_1] = \frac{f(X_1) - f(X_0)}{X_1 - X_0}$$

$$f(X) = f(X_0) + (X-X_0)f[X,X_0]$$

- Quindi il ***termine $P_0(X) = f(X_0)$ rappresenta il polinomio di grado 0*** interpolante la funzione f(X) per $X = X_0$
  - $(X-X_0)f[X,X_0]$ denota l'**errore** $f(X) - p_0(X)$

- Consideriamo la definizione di differenza divisa di ordine 2:

$$f[X, X_0, X_1] = \frac{f[X_1, X_0] - f[X_0, X]}{X_1 - X}$$

  - sostituendola in $f(X) = f(X_0) + (X-X_0)\, f[X,X0]$

POLINOMIO INTERPOLAZIONE: NEWTON

Si ottiene:

$f(X)=f(X_0)+(X-X_0)\, f[X_0,X_1]+(X-X_0)(X-X_1)\, f[X,X_0,X_1]$

- Quindi il termine $p_1(X)=f(X_0)+(X-X_0)\, f[X_0,X_1]$ rappresenta il polinomio di grado 1 interpolante la funzione f(X) per $X = X_0$ e $X=X_1$
- $(X-X_0)(X-X_1)\, f[X,X_0,X_1]$ denota l'errore $f(X)-p_1(X)$

- Iterando il procedimento, fino ad includere tutti i nodi $X_0,X_1,\ldots,X_n$ otteniamo **$f(X) = p_n(X) + E_n(X)$**

dove

$p_n(X) = f(X_0) + f[X_0,X_1](X-X_0) +\ldots+ f[X_0,X_1,\ldots X_n](X-X_0)\ldots(X-X_{n-1})$
$E_n(X)=(X-X_0)(X-X_1)\ldots(X-X_n)f[X,X_0,X_1,\ldots,X_n]$

POLINOMIO INTERPOLAZIONE NEWTON

Esempio:

Costruire il polinomio di grado n=4 passante per i punti:
(-4, 1245), (-1, 33), (0,5), (2,9), (5, 1335)

X_i	$f(X_i)$	$f[X_i,X_{i+1}]$	$f[X_i,X_{i+1},X_{i+2}]$.....		
-4	1245				
-1	33	-404			
0	5	-28	94		
2	9	2	10	-14	
5	1335	442	88	13	3

$$p_4(X) = 1245-404(X+4)+94(X+4)(X+1)$$
$$-14(X+4)(X+1)X+3(X+4)(X+1)X(X-2)=$$
$$= 3X^4 -5X^3 +6X^2-14X+5$$

POLINOMIO INTERPOLAZIONE NEWTON

Vantaggi del metodo:

- Costruzione polinomio richiede ***n^2 sottrazioni e $n^2/2$ divisioni***

- Polinomio $p_{n+1}(X)$ costruito sui nodi $X_0, X_1, \ldots X_n$ e su un nuovo punto Xn+1 si può ricavare da $p_n(X)$ con la seguente relazione:

$$p_{n+1}(X) = p_n(X) + (X-X0)(X-X1)\ldots(X-Xn)f[X0,X1,\ldots,Xn+1]$$

CALCOLO DELLE DIFFERENZE DIVISE

```
float diff_div (float *X, float *Y, int max, int min) {
/*versione ricorsiva */
   float k;
   if ((max-min)==0) return Y[0];        /*ordine 0 */
   else
      if ((max-min)==1) /*ordine 1*/
      return ((Y[max]-Y[min])/(X[max]-X[min]));
      else {
            k=((diff_div(X,Y,max, min+1)-
                  diff_div(X,Y,max-1,min))/(X[max]-X[min]));
            return k;
       }
 }
```

$$f[X_0, X_1, \cdots, X_n] = \frac{f[X_1, X_2, \cdots, X_n] - f[X_0, X_1, \cdots, X_{n-1}]}{X_n - X_0}$$

$$= \sum_{k=0}^{n} \frac{f(X_k)}{\prod_{j=0,\, j\neq k}^{n} (X_k - X_j)}$$

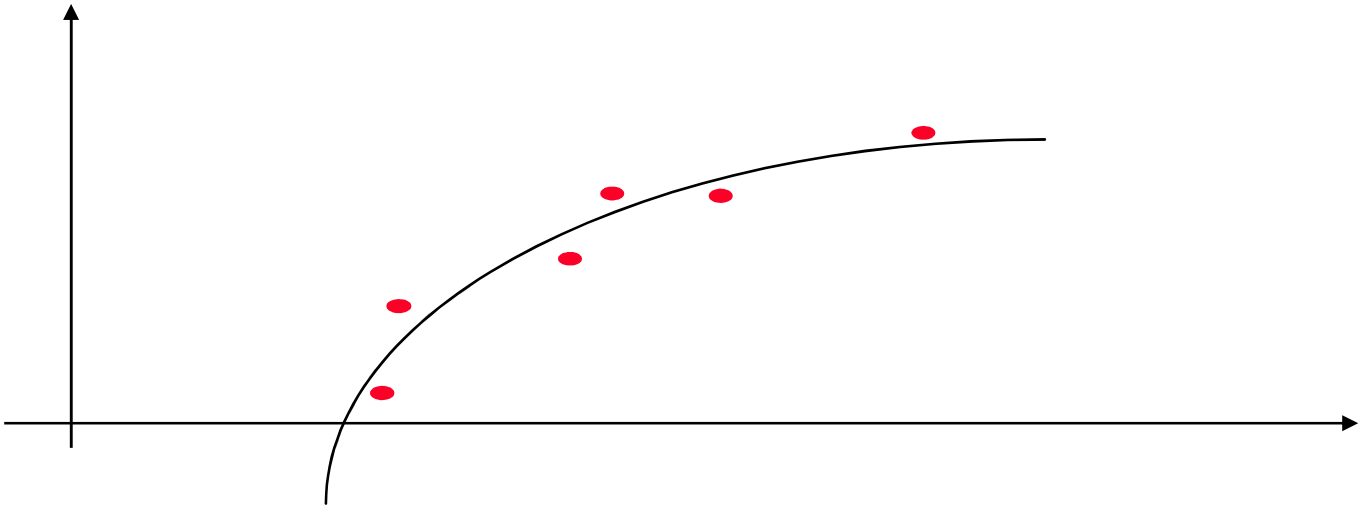
Polinomio Interpolazione Newton: ALGORITMO

```
float diff_div(float *X, float *Y, int max, int min);

float newton (float *X, float*Y, int dim, float x) {
/* X e Y vettori coordinate punti noti, dim numero punti
   noti,  x valore in cui calcolare p(x) */

   float ris=0, T; int i,k;
   for (i=0; i<dim; i++) {
/* calcolo coeff. i-simo polinomio (differenza divisa) */
      T=diff_div(X,Y,i,0);
      for(k=0; k<i; k++)
            T*=(x-X[k]);
      ris+=T;
   }
   return ris;
 }
```

METODO DEI MINIMI QUADRATI

- Metodi di Lagrange e Newton impongono che curva approssimante passi esattamente per punti dati

- Il metodo dei ***minimi quadrati*** trova ***curva che minimizza la "distanza" media dei punti dati da essa***. In generale, quindi, i punti dati non giacciono sulla curva approssimante



METODO DEI MINIMI QUADRATI

- Usato per:
  - Risolvere ***sistemi lineari rettangolari*** (numero di incognite inferiore al numero delle equazioni)
  - Determinare ***espressione analitica di curva*** passante per un determinato numero di punti
- Conviene se il numero n di punti è molto grande (polinomio approssimante di grado m<<n) o se il valore dei punti è approssimato
- Possibilità di attribuire diversa importanza a differenti osservazioni:
  - peso w_i di una misura sperimentale (X_i, Y_i)

METODO DEI MINIMI QUADRATI

- Si consideri come funzione approssimante Y = f(X) un polinomio di grado m<<n:

$$y=f(X) = a_0 + a_1X + a_2X^2 +.... + a_mX^m \quad (*)$$

- Si tratta di ***determinare i coefficienti*** a_j in modo che il ***polinomio dia una buona approssimazione*** della curva passante per gli n punti

- Sostituendo gli n punti in (*) si otterrebbe un sistema lineare di n equazioni lineari in m+1 incognite (coefficienti a_j):

$$\mathbf{y_i = f(X_i) = a_0 + a_1X_i + a_2X_i^2 +....+a_mX_i^m}$$

$$\mathbf{i=(1,...,n)}$$

METODO DEI MINIMI QUADRATI

- Sistema trovato ***non ammette soluzioni*** avendo un numero di equazioni > delle incognite (a meno che n-(m+1) righe siano combinazioni lineari delle altre m+1)

- Ciascuna equazione allora sarà soddisfatta a meno di ***residuo Ri***:

$\forall$ i=1,..n

$y_i = f(X_i) = a_0 + a_1X_i + a_2X_i^2 + ... + a_mX_i^m + R_i$

$R_i = y_i - (a_0 + a_1X_i + a_2X_i^2 + ... + a_mX_i^m)$

- Per determinare ***valore dei coefficienti*** a_j (j =0,...m) si applica il principio dei minimi quadrati:

"Il polinomio approssimante ottimo degli n punti dati è quello che rende minima la somma pesata dei quadrati dei residui"

METODO DEI MINIMI QUADRATI

- Si deve quindi trovare il minimo della funzione:

$$f(a_0,a_1,....,a_m)=w_1R_1^2+w_2R_2^2+...+w_nR_n^2$$

(w_1, w_2,..w_n sono i pesi dei vari campioni)

- Minimo si ha per quei valori di a_j che ***annullano m+1 derivate parziali*** di $f(a_0,a_1,....,a_m)$ fatte rispetto ai parametri a_j (j=0,1,...,m)

$$\frac{\partial f}{\partial a_0}=2\cdot(w_1R_1\frac{\partial R_1}{\partial a_0}+w_2R_2\frac{\partial R_2}{\partial a_0}+\cdots w_nR_n\frac{\partial R_n}{\partial a_0})=0$$

$$\frac{\partial f}{\partial a_1}=2\cdot(w_1R_1\frac{\partial R_1}{\partial a_1}+w_2R_2\frac{\partial R_2}{\partial a_1}+\cdots w_nR_n\frac{\partial R_n}{\partial a_1})=0$$

$$\cdots$$

$$\frac{\partial f}{\partial a_m}=2\cdot(w_1R_1\frac{\partial R_1}{\partial a_m}+w_2R_2\frac{\partial R_2}{\partial a_m}+\cdots w_nR_n\frac{\partial R_n}{\partial a_m})=0$$

- ***Sistema lineare di m+1 equazioni DIFFERENZIALI in m+1 incognite***

METODO DEI MINIMI QUADRATI

Ipotesi semplificative "brutali":

- Interpolante semplicissima retta f(x)=mx+q

Nel caso di interpolante f(x)=mx+q

Occorre minimizzare $\sum_{i=1}^{n}(y_i - f(x_i))^2$ ⇒ risultato:

$$m = \frac{\sum x_i \sum y_i - N\sum y_i x_i}{\left(\sum x_i\right)^2 - N\sum x_i^2}$$

$$q = \frac{\sum x_i \sum x_i y_i - \sum x_i^2 \sum y_i}{\left(\sum x_i\right)^2 - N\sum x_i^2}$$

METODO DEI MINIMI QUADRATI

- Interpolante come funzione lineare rispetto ai parametri
  $f(x) = p_1 f_1(x) + p_2 f_2(x) + \ldots + p_k f_k(x)$
  con k<<n (numero dei punti noti)

$$\Rightarrow \quad Ap \sim y \qquad \text{dove } A = \begin{bmatrix} f_1(x_1) & \ldots & \ldots & f_k(x_1) \\ \ldots & & & \ldots \\ \ldots & & & \ldots \\ f_1(x_n) & \ldots & \ldots & f_k(x_n) \end{bmatrix}$$

$$p = \begin{bmatrix} p_1 \\ \ldots \\ p_k \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ \ldots \\ y_n \end{bmatrix}$$

Minimizzare $$norma(Ap - y)^2 = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

METODO DEI MINIMI QUADRATI

Andando ad annullare le derivate:

$\Rightarrow (Ap - y)^T \; A = 0$

$\Rightarrow A^T Ap = A^T y \qquad \Rightarrow \qquad p = (A^T A)^{-1} \; A^T y$

$(A^T A)^{-1} \; A^T$ matrice pseudo-inversa

TECNICA BEST-FIT

- Mediante questa tecnica si può ***determinare grado m del polinomio approssimante*** n punti dati
- Si parte da una retta e si determinano i parametri coi minimi quadrati; successivamente una parabola, una cubica e via via polinomi di grado crescente fino a raggiungere l'approssimazione richiesta
- Per valutare grado approssimazione mediante determinato polinomio di grado m si considera ***scarto quadratico medio:***

$$\bar{S} = \sqrt{\frac{\sum_{i=1}^{n} w_i R_i^2}{n-(m+1)}}$$

dove n numero di punti dati;

(n-(m+1)) sono i gradi di libertà del problema

BEST FIT

- Essendo un metodo statistico si ***preferiscono valori bassi di m***

- In pratica, si prova iterativamente con m = 0,1,2,... e ci si arresta quando S aumenta oppure la diminuzione è minore di un valore fissato

- Nota: come funzione approssimante si può utilizzare ***qualsiasi funzione i cui parametri incogniti siano legati da una relazione lineare***