

CALCOLO NUMERICO – GIUNTA/GALLETTI

UNIVERSITÀ DEGLI STUDI DI NAPOLI
“PARTHENONE”



Appunti a cura di
FIORENTINO MICHELE

INTRODUZIONE

I seguenti appunti sono stati scritti durante la frequentazione del corso di CALCOLO NUMERICO tenuto dai professori Giulio Giunta e Ardelio Galletti nell' a.a. 2020/2021.

CALCOLO NUMERICO - INDICE

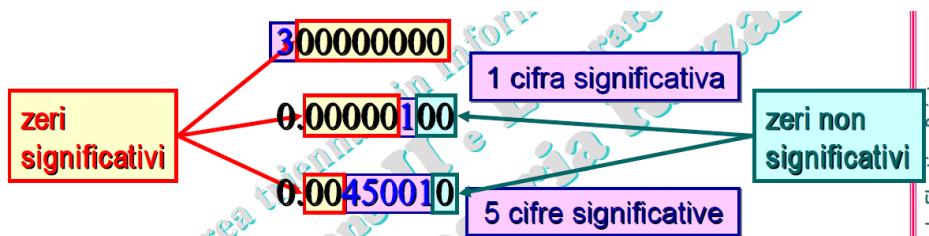
- | | | |
|----|--------------|--|
| 2 | - Lez 1: | Sistema aritmetico di un computer |
| 5 | - Lez 2: | Continuo sui richiami FP |
| 8 | - Lez 3: | Risoluzione di un'equazione, e algoritmo di bisezione |
| 14 | - Lez 4: | Algoritmo di Newton |
| 16 | - Lez 5: | Algoritmo delle secanti e algoritmi ibridi |
| 19 | - Lez 6: | Problema del punto fisso |
| 23 | - Lez 7: | Problema del minimo di una funzione |
| 29 | - Lez 8: | Calcolo del minimo di una funzione unimodale |
| 34 | - Lez 9: | Richiami di algebra lineare |
| 45 | - Lez 10: | Risoluzione di sistemi lineari |
| 49 | - Lez 11: | Risoluzione di sistemi lineari con Gauss |
| 56 | - Lez 12: | Fattorizzazione LU |
| 60 | - Lez 13: | Fitting di dati: interpolazione di Lagrange |
| 64 | - Lez 14: | Fitting di dati: interpolazione di Hermite e Qualità della ricostruzione |
| 69 | - Lez 15: | Interpolazione con polinomi a tratti |
| 72 | - Lez 16: | Continuo spline cubiche, e cubiche di Hermite |
| 78 | - Lez 17: | Fitting di dati: approssimazione (ricostruzione di trend) |
| 84 | - Lez 18: | Continuo approssimazioni, SL sovradeterminati e Quadratura |
| 91 | - Lez 19/20: | Formule composite e andamento dell'errore |
| 99 | - Lez 21: | Statistica descrittiva |

LEZIONE 1 – SISTEMA ARITMETICO DI UN COMPUTER

La notazione più compatta per rappresentare i numeri reali è quella scientifica in cui si rappresentano solo le cifre significative.

Sostanzialmente si può dire che gli zeri non significativi di un numero sono quelli per cui, anche se ignorati, il suo valore non cambia.

Altri zero invece sono significativi in quanto determinano l'ordine di grandezza del numero (centinaia, migliaia, ecc.).



notazione scientifica

$$300000000 = 3 \cdot 10^8 = 3e+8$$

$$0.00000100 = 1 \cdot 10^{-6} = 1e-6$$

$$0.00450010 = 4.5001 \cdot 10^{-3} = 4.5001e-3$$

La *normalizzazione della mantissa* viene utilizzata per individuare univocamente la notazione scientifica di un numero.

Tale mantissa m deve essere tale che $1 \leq m < \beta$.

Naturalmente i numeri normalizzati che richiedono la nostra attenzione sono quelli espressi nel sistema di numerazione binario.

Notar bene che i **numeri binari normalizzati hanno come prima cifra significativa sempre 1**.

Questa conclusione fornisce la possibilità di evitare di memorizzare esplicitamente tale cifra significativa dando origine alla cosiddetta “rappresentazione a bit隐式”.

L'esponente è memorizzato nella forma **esponente + bias**, in modo da memorizzare sempre un numero positivo.

Lo 0 è rappresentato come una sequenza di bit nulli.

In tutti i computer (e dispositivi digitali) i numeri sono rappresentati in modo standard e sono detti *numeri floating point* (fp). Il formato standard di un numero floating point è:

- **base 2** (solo due cifre: 0,1);
- **mantissa** (normalizzata, detta **frazione**) per base elevata a **esponente** (intero);
- **a precisione finita** (sia la mantissa che l'esponente hanno un numero prefissato di cifre)

Più precisamente, il **sistema aritmetico floating point** denota l'insieme dei numeri reali rappresentabili in un computer e le operazioni definite su di essi.

Esso è rappresentato come $F(\beta, t, E_{\min}, E_{\max})$, dove:

- β = base del sistema di numerazione;
- t = numero di cifre $_{\beta}$ per la mantissa;
- $E_{\min} < E_{\max}$ = limitazioni per il campo esponente. (Quindi, con E si intende il range dell'esponente).

IEEE Standard 754

Lo standard per la rappresentazione dei numeri in virgola mobile è l'IEEE 754, utilizzato ormai da tutti i computer moderni.

In particolar modo distinguiamo la rappresentazione fp a *singola precisione* e a *doppia precisione*.



Il **motivo** per cui si è deciso di rappresentare in questo modo i numeri fp è quello di simulare, con un errore piccolo (che dipende dalla precisione di t), un grande intervallo di numeri reali.

Da notar bene che per rappresentare esattamente qualsiasi numero reale, a meno che non si tratti di un intero o di un razionale, è necessario una mantissa a precisione infinita.

Proprio perché è possibile rappresentare solo una porzione dei numeri reali, parliamo di **intervallo rappresentabile**, sul quale dobbiamo fare attenzione a 3 cose:

- **overflow**: numeri troppo grandi, in valore assoluto;
- **underflow**: numeri troppo piccoli, in valore assoluto;
- **numeri rappresentabili esattamente**: numeri la cui mantissa ha un numero di cifre minore o uguale alla precisione del sistema.

Quando non è possibile rappresentare esattamente un determinato numero reale, utilizziamo la filosofia **round to nearest** (di solito si prende quello “pari” più vicino).

Ovviamente tale approssimazione da origine ad un errore, il quale è minore di 2^{-t+1} . In doppia precisione t=52, quindi l'errore sulla mantissa è $<2^{-53}$, che è circa 10^{-16} .

Ciò significa che se tale numero è espresso in base 10, allora la sedicesima cifra significativa (cioè l'ultima) del numero fp può essere sbagliata per meno di mezza unità.

Massima accuratezza

È possibile ottenere la *massima accuratezza* attraverso registri interni di almeno 80 bit, con mantissa di almeno 58 bit.

Aritmetica standard IEEE

Le varie componenti di un numero sono rappresentate come segue:

- il **segno** è rappresentato con un bit, che vale 0 se positivo o 1 se negativo.
- la **base** dell'esponente è 2
- l'**esponente** è 127+e in singola precisione, 1023+e in doppia precisione.
- il **primo bit della mantissa**, che è sempre 1, non viene memorizzato. Quindi il campo della mantissa contiene solo la *frazione*.

Da notar bene che il più piccolo numero rappresentabile è del tipo:

$$0 \ 00000001 \ 00000000000000000000000000000000 \rightarrow 1 \times 2^{-126}$$

da notare come l'esponente = 0 sia riservato per situazioni eccezionali.

LEZIONE 2 – CONTINUO SUI RICHIAMI FP

Operazioni aritmetiche con numeri fp

L'unità aritmetica di ogni computer è in grado di eseguire solo le quattro operazioni aritmetiche (+, -, *, /) su numeri fp. Il risultato dato da tale operazione sarà ancora un numero fp.

Si noti che in generale, ogni qualvolta si effettua un'operazione aritmetica in un computer, sul risultato grava un errore, cioè l'errore sulla mantissa $< 2^{-53} \approx 10^{-16}$ ed errore assoluto $< 10^{-16}$ per l'ordine di grandezza del risultato.

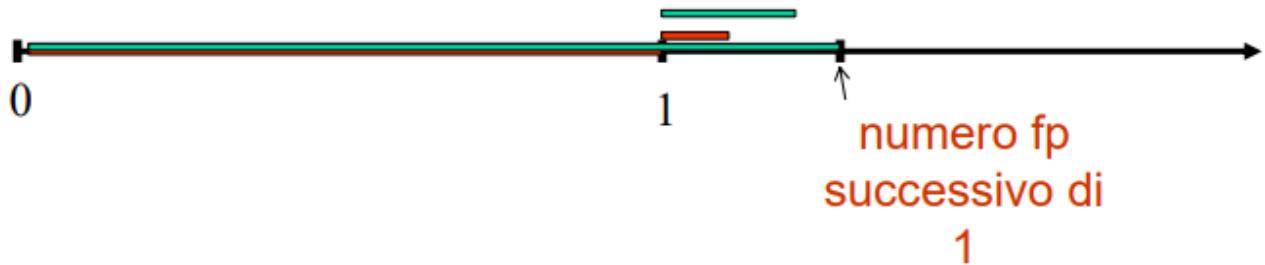
Epsilon macchina

L'*epsilon macchina* è il più piccolo numero fp, che sommato ad 1.0, mediante *addizione fp*, fornisce un risultato strettamente maggiore di 1.0.

Dalla definizione deduciamo che la somma fp fra 1.0 ed un qualunque numero fp **y**, compreso nell'intervallo $[0, \text{epsMacchina})$, darà sempre come risultato 1.0 .

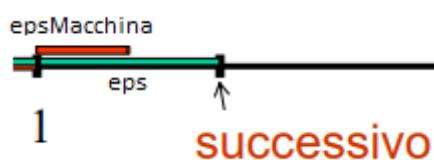
Ciò significa che l'operazione di addizione *fp*, a differenza della operazione di addizione usuale, **ammette elementi neutri diversi da zero**.

$$\forall y \in [0, \text{epsMacchina}) \Rightarrow 1 \oplus y = 1$$



NB: in Matlab è presente una variabile “*eps*” che tuttavia non rappresenta l'*epsMacchina* ma 2^{-52} , ovvero il valore dell'ultimo bit in doppia precisione.

Per ottenere *epsMacchina* dobbiamo fare *eps/2*.



Errore assoluto & Errore relativo

(Teoremi dell'errore) sia dato un numero x e una sua approssimazione a , diremo che:

errore assoluto di a

$$E_{ass} = |x - a|$$

se x e a hanno la stessa **parte intera** e le stesse prime m cifre della **parte frazionaria**, allora

$$E_{ass} = |x - a| < 10^{-m}$$

errore relativo di a

$$E_{rel} = \frac{|x - a|}{|x|}$$

se x e a hanno le stesse prime m **cifre significative**, allora

$$E_{rel} = \frac{|x - a|}{|x|} \approx 10^{-m}$$

ricordando che le cifre significative di un numero sono le cifre della sua mantissa esclusi gli eventuali “zeri in testa”.

Da notar bene come non sia strettamente vero il viceversa, ovvero non è detto che se $E_{ass} = \dots$, allora x e a hanno la stessa parte intera e le stesse prime m cifre della parte frazionaria, e non è detto che se $E_{rel} = \dots$, allora x e a hanno le stesse prime m cifre significative.

Da tali teoremi ci rendiamo conto che l'errore assoluto coinvolge tutta l'informazione sui due numeri, mentre l'errore relativo coinvolge solo le loro mantisse, cioè solo le cifre significative e non l'ordine di grandezza.

ESEMPI

$$x = 21.1281 \quad a = 21.1243$$

$$E_{ass} = 0.38 \cdot 10^{-2} < 10^{-2}$$

$$x = 1234.5649 \quad a = 1234.5611$$

$$E_{ass} = 0.38 \cdot 10^{-2} < 10^{-2}$$

$$x = 21.1281 \quad a = 21.1276$$

$$E_{ass} = 0.5 \cdot 10^{-3} < 10^{-2}$$

$$x = 1.9999 \quad a = 2.0001$$

$$E_{ass} = 0.2 \cdot 10^{-3} < 10^{-2}$$

$$x = 21.1281 \quad a = 21.1243$$

$$E_{rel} = 1.79 \cdot 10^{-4} \approx 10^{-4}$$

$$x = 1234.5649 \quad a = 1234.5611$$

$$E_{rel} = 3.07 \cdot 10^{-6} \approx 10^{-6}$$

$$x = 21.1281 \quad a = 21.1276$$

$$E_{rel} = 2.36 \cdot 10^{-5} \approx 10^{-5}$$

$$x = 1.9999 \quad a = 2.0001$$

$$E_{rel} \approx 10^{-4}$$

C'è da fare un importante **considerazione** sull'errore assoluto e su quello relativo:

- **errore assoluto**: è una quantità *dimensionale*.

Infatti essendo una semplice sottrazione, se **x** ed **a** avessero un'unità di misura, tale unità di misura verrebbe preservata.

Es. $|1.81 \text{ kg} - 1.43 \text{ kg}| = 0.38 \text{ kg}$.

- **errore relativo**: è una quantità *adimensionale*.

In tal caso avviene una divisione fra l'errore assoluto e il valore assoluto di **x**.

Avendo entrambi la stessa unità di misura, ed essendo una divisione, questa viene semplificata e rimane un numero puro.

$$\text{Es. } \frac{(|1.81 - 1.43|) \text{ kg}}{|1.83| \text{ kg}} \rightarrow \frac{(|1.81 - 1.43|)}{|1.83|} = 0.207 .$$

NB:

Quando lavoriamo con un computer, l'*epsMacchina* consente di maggiorare l'errore relativo che si commette ogni volta che si esegue una addizione in un computer.

Se **x** e **y** sono numeri fp, allora:

$$\frac{|(x \oplus y) - (x + y)|}{|x + y|} \leq \text{epsMacchina}$$

In MATLAB scriveremo tale operazione come:

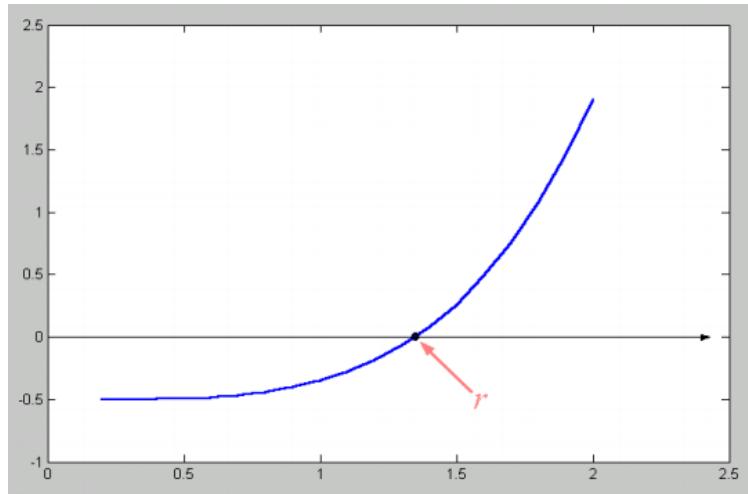
$$|(x \oplus y) - (x + y)| \leq 0.5 * \text{eps}(\text{abs}(x+y))$$

LEZIONE 3 – RISOLUZIONE DI UNA EQUAZIONE (e alg. di bisezione)

Con *risoluzione di un'equazione* si intende risolvere l'equazione $f(x) = 0$, ove f è una funzione di una variabile $f : \mathbb{R} \rightarrow \mathbb{R}$, cioè significa trovare un numero r tale che $r : f(r) = 0$.

r prende il nome di **zero** della funzione (ma è anche detto *soluzione* o *radice*).

Dal punto di vista grafico, r è l'ascissa del punto di intersezione del grafico della funzione $f(x)$ con l'asse delle x .



Da notare che $f(x)=0$ potrebbe anche apparire in forma diversa, ma è sempre facilmente riconducibile ad esso.

Es. $w(x) = v(x) \rightarrow w(x) - v(x) = 0$, ove $f(x) = w(x) - v(x)$

o anche. $f(x) = w(x)/v(x) - 1 \rightarrow f(x) = 0$

Un grafico $f(x)$ potrebbe anche **solo toccare** l'asse delle x , o ancora potrebbe avere **zeri multipli**. È un dettaglio importante in quanto la presenza di **zeri multipli** può avere degli effetti su determinati algoritmi.

Da notare che la funzione **f non è nota esattamente**. Infatti, utilizzando un computer, sia i valori di x che $f(x)$ sono noti al più con 16 cifre (in base 10), cioè con un errore relativo di circa 10^{-16} .

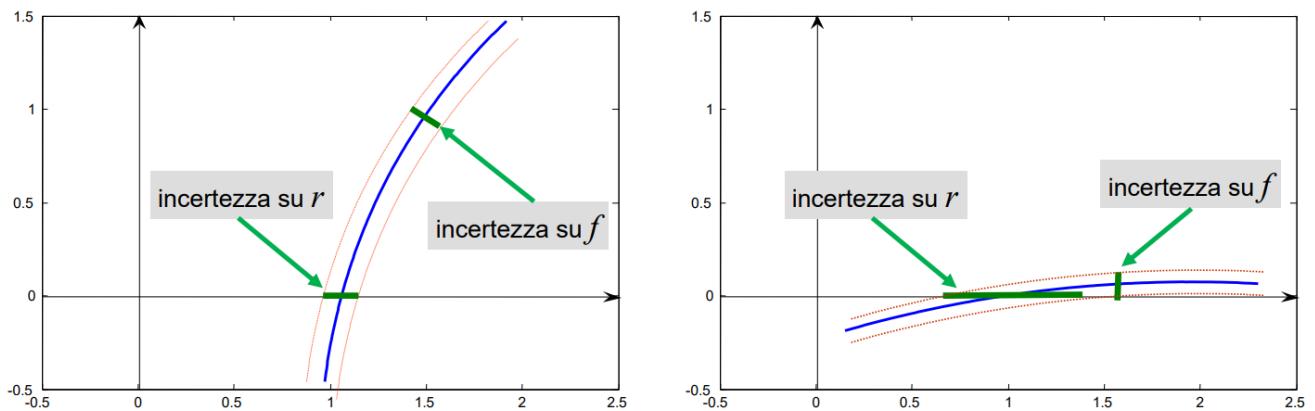
Tuttavia non è detto che se su f è presente una certa incertezza, allora anche la soluzione r dovrà avere la stessa incertezza. In particolar modo si dice che:

- un problema è **ben condizionato** se è poco sensibile ad errori sulla funzione, quindi anche con piccoli errori nella funzione non abbiamo soluzioni molto diverse.
- un problema è **mal condizionato** se è molto sensibile ad errori sulla funzione, quindi piccoli errori nella funzione possono dare soluzioni molto diverse.

Da notare che l'incertezza è rappresentato da tutta la regione nella quale è contenuta la funzione, la quale indica dove può trovarsi la funzione stessa. Da notare che l'intervallo di **incertezza su f** non sta solo su f ma anche su x (per questo considero l'intervallo in diagonale, e non nello specifico sulla verticale).

Dall'intervallo di incertezza su f dipende l'intervallo di **incertezza sulla soluzione (r)**.

Notare come nel primo caso sia f che r hanno la stessa incertezza (p. ben condizionato) mentre nel secondo r ha una incertezza molto più ampia (p. mal condizionato).



Da un punto di vista *quantitativo* è possibile stimare il condizionamento di un problema attraverso l'**indice di condizionamento**:

$$\kappa_f(r) = |f'(r)|^{-1}$$

(l'indice di condizionamento k , che dipende sia da r che da f , è dato dal valore assoluto dell'inverso della derivata prima di f in r).

TEOREMA DI ABEL

Non sempre è possibile esprimere la soluzione r di una equazione mediante una **formula chiusa**.

Infatti non esiste una formula, che contiene un numero finito di operazioni aritmetiche, per calcolare gli zeri di un qualunque polinomio di grado maggiore o uguale a 5.

(nb: con formula "chiusa" si intendono quelle formule come ad esempio $x_{1,2} = (-b \pm \sqrt{d})/2$, che esistono anche per polinomi di 3 o 4 grado, ma non per quelli dal 5 in poi).

Di conseguenza posso solo calcolare un'approssimazione di r , ricavabile attraverso una **formula ricorrente**, che a sua volta può essere calcolata attraverso un **algoritmo iterativo**, che sotto opportune ipotesi, converge alla soluzione del problema.

Un qualsiasi algoritmo iterativo che permetta di risolvere $f(x)=0$ ha bisogno di:

- un'**approssimazione iniziale** x_0 .
- un **criterio di arresto**, ovvero dopo quanti passi devo arrestare il processo iterativo. (rappresenta l'*accuratezza richiesta*);

Da qui ricaviamo che gli input di tale algoritmo saranno la funzione f , x_0 , e il criterio di arresto.

Mi fermerà quando l'errore (err_k) sarà minore rispetto all'accuratezza richiesta, ove l'errore dovrà essere calcolato ad ogni passo k .

Due algoritmi che risolvono entrambi il problema di $f(x)=0$ possono essere messi a confronti (ad un livello prestazionale) attraverso i parametri di **convergenza** e **rapidità di convergenza** (quanti passi impiega).

Un **qualsiasi algoritmo che risolve $f(x)=0$** effettua una serie di **passaggi**:

1. se ci sono più soluzioni, **individuare** la soluzione di interesse;
Questi algoritmi iterativi sono in grado di calcolare solo una soluzione alla volta, se fossi interessato a più soluzioni dovrei riapplicare l'algoritmo più volte.
2. fissare l'**accuratezza** richiesta per il calcolo dell'approssimazione della soluzione (ove tale approssimazione rappresenta una *maggiorazione* dell'errore assoluto o dell'errore relativo (ricorda i *teoremi dell'errore*, se scegliere uno o l'altro dipende da cosa vogliamo)).
3. **applicare** l'algoritmo per trovare una approssimazione di tale soluzione.
4. quantificare la **bontà** dell'approssimazione.

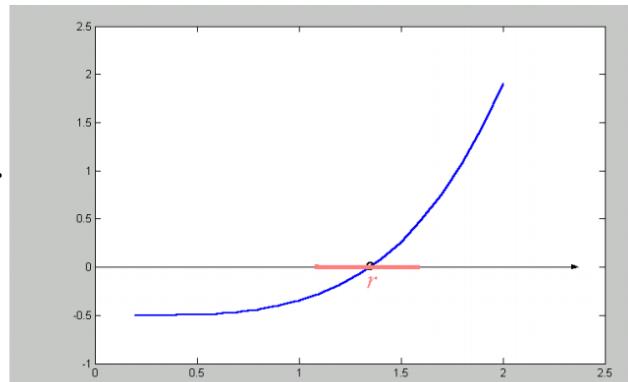
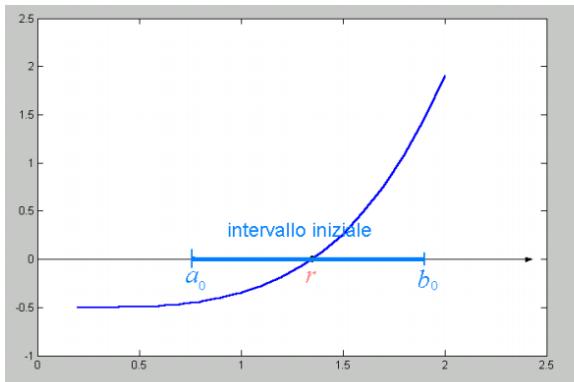
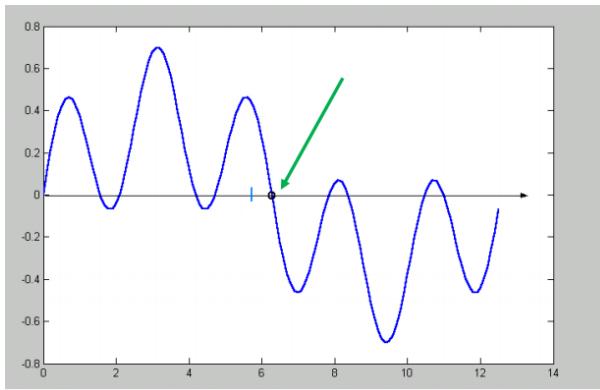
La **localizzazione** di una soluzione di interesse può avvenire in due modi:

- dando un valore che sia vicino a quello della soluzione esatta (si va ad occhio);
- fissando gli estremi di un intervallo che contiene la soluzione, andando ad ogni passo a diminuire l'ampiezza di tale intervallo selezionando la semi-ampiezza che contiene la soluzione esatta. Il *punto medio* dell'intervallo rappresenta l'approssimazione r .

Il secondo approccio ha un vantaggio particolare. Infatti l'ampiezza dell'intervallo è una *stima dell'incertezza*, e così facendo andiamo a diminuire tale incertezza ad ogni passo.

Il **residuo** è il valore della funzione nella posizione x_k , ovvero $f(x_k)$. Se il residuo è circa dello stesso ordine di grandezza dell'errore assoluto, il problema è *ben condizionato*.

nb: quando si risolve il problema $f(x)=0$, l'errore assoluto deve essere minore dell'accuratezza richiesta.



nb:

- prob. ben condizionato: residuo piccolo IMPLICA errore piccolo
- prob. mal condizionato: residuo piccolo NON IMPLICA errore piccolo

Algoritmo di bisezione

Il metodo di bisezione è un algoritmo iterativo **globale** e a **velocità di convergenza lineare** che risolve il problema $f(x) = 0$.

Con *globale* si intende che l'algoritmo è sempre convergente (quando è applicabile!).

Con *velocità di convergenza lineare* si intende che l'errore si riduce di un fattore costante a ogni passo di iterazione.

L'algoritmo prende in **input** la funzione f e due valori a, b che rappresentano gli estremi dell'intervallo. Per poter utilizzare tale algoritmo devono essere rispettate due **condizioni**:

- $f(x)$ deve essere una funzione continua in $[a,b]$;
- $f(x)$ cambia segno in $[a,b]$.

Inoltre, per essere **applicabile**:

- $f(x)$ deve essere nota (quindi conosciamo l'espressione $f(x)$, oppure abbiamo un programma che ci permetta di calcolare il valore di $f(x)$ per ogni x in $[a,b]$);
- $[a,b]$ deve essere noto.

L'algoritmo è basato sull'approccio detto divide et impera, che riduce progressivamente (dimezzandolo a ogni passo) l'intervallo che contiene la soluzione.

A ogni passo k , si ha un intervallo che contiene la soluzione: il **punto medio** dell'intervallo rappresenta l'**approssimazione** della soluzione mentre la **semi-ampiezza** dell'intervallo rappresenta la **stima dell'errore** tra tale approssimazione e la soluzione.

Il fatto che l'errore si dimezza a ogni passo di iterazione implica che se al passo k l'errore è 2^{-s} (cioè l'approssimazione ha s cifre della parte frazionaria corrette) allora al passo successivo ($k+1$) l'errore sarà 2^{-s-1} , ovvero la nuova approssimazione avrà un bit corretto in più rispetto all'approssimazione precedente.

Il ragionamento possiamo farlo anche in base 10, con la differenza che abbiamo bisogno di 4 bit (10 simboli), quindi la nuova approssimazione avrà una cifra corretta in più dopo quattro passi di iterazione.

esempio di esecuzione

ad ogni passo k calcoliamo il punto medio m , e verifichiamo se l'intervallo $f(a_k, m_k) \leq 0$. Se così fosse, $[a_k, m_k]$ diventerà al passo successivo il nuovo intervallo $[a_k, b_k]$, altrimenti lo diventerà l'intervallo $[m_k, b_k]$.

Al passo k , **[a_k, b_k]** conterrà la radice.

L'**errore** di approssimazione è dato da:

$$err_k = |r - m_k| \leq \frac{1}{2} |b_k - a_k| \equiv e_k$$

$$e_k = \left(\frac{1}{2}\right)^{k+1} (b - a)$$

È possibile determinare il **minimo** k tale che l'errore di approssimazione sia minore di un'**accuratezza prefissata** (Δ). Ergo sapendo che:

$$\begin{array}{c} k : e_k \approx |r - m_k| \leq \Delta \\ \downarrow \\ \text{risolvere la disequazione} \\ \text{rispetto a } k \\ k : \left(\frac{1}{2}\right)^{k+1} (b - a) \leq \Delta \end{array}$$

il **minimo** k intero che verifica l'uguaglianza sarà:

$$k = \left\lceil \log_2 \left(\frac{b - a}{\Delta} \right) - 1 \right\rceil$$

nb: notare come somigli all'algoritmo di ricerca binaria, dove al size dell'input corrisponde il rapporto tra l'*incertezza iniziale* e l'*accuratezza richiesta*.

La **dimensione computazionale del problema** è data dal rapporto tra l'incertezza iniziale e l'accuratezza richiesta: $\frac{b-a}{\Delta}$.

Da notare che in un algoritmo che risolve il problema per $f(x)=0$, nella **complessità di tempo**, l'operazione dominante è la **valutazione della funzione f**.

Dunque:

$T\left(\frac{b-a}{\Delta}\right)$ = numero di passi **per** numero di valutazioni di f a ogni passo

LEZIONE 4 – RISOLUZIONE DI UNA EQUAZIONE (con alg. di Newton)

Il metodo di Newton è un algoritmo iterativo **locale** e a **velocità di convergenza quadratica** che risolve il problema $f(x) = 0$.

Con locale si intende che l'algoritmo non è sempre convergente e che la convergenza dipende da come si sceglie la condizione iniziale. Con velocità di convergenza quadratica si intende che, a ogni passo di iterazione, l'errore relativo si riduce diventando il (proporzionale al) quadrato dell'errore precedente (si ricorda che gli errori sono del tipo 10^{-s}).

Tale algoritmo è utilizzabile se f è una funzione continua e derivabile (con derivata nota $f'(x)$) e se si conosce una buona approssimazione iniziale x_0 della soluzione.

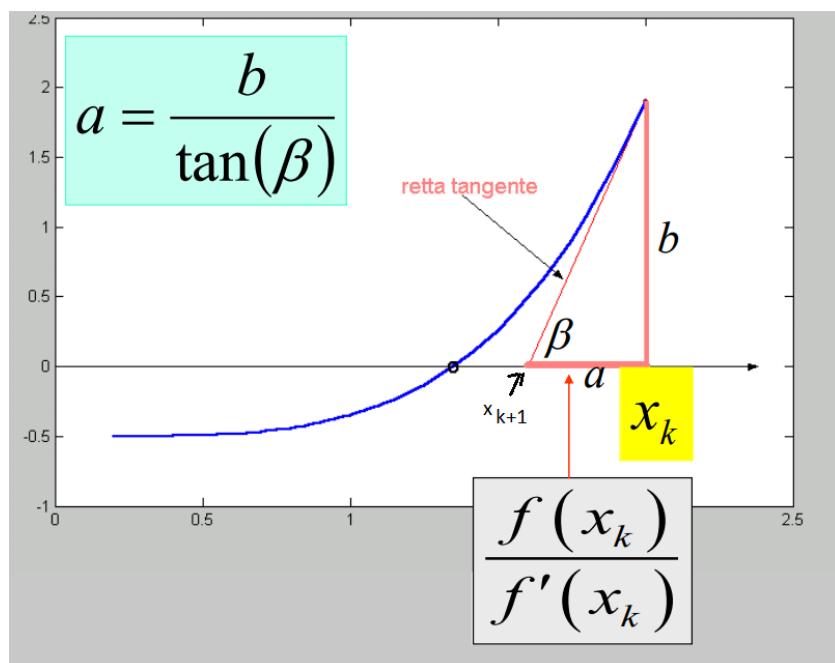
Da un **punto di vista geometrico** consiste nel tracciare, ad ogni passo, la retta tangente alla funzione $f(x)$ nel punto $(x_k, f(x_k))$.

Calcolare $f'(x)$ significa calcolare la *tangente* dell'angolo che si viene a creare dall'intersezione della retta tangente in quel punto con la funzione $f(x)$, ovvero significa calcolare la **pendenza della retta**.

La retta tangente intercercherà sicuramente l'asse delle ascisse da qualche parte. Tale punto di intersezione x_{k+1} diventerà la nuova approssimazione.

Per poterlo esprimere in **forma ricorrente** dobbiamo innanzitutto notare come fra la retta tangente, l'asse delle x ed $f(x_k)$ si formi un **triangolo rettangolo**.

Come sappiamo dalla trigonometria, in un *triangolo rettangolo con cateti a e b , la lunghezza del cateto a è uguale al rapporto tra la lunghezza del cateto b e la tangente dell'angolo opposto al cateto b (beta).*



Essendo $\beta = f'(x)$ e $b=f(x_k)$, per conoscere x_{k+1} dovrò togliere da x_k la lunghezza del cateto a .

$$\text{Ovvero: } X_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

$$\text{o anche più semplicemente: } X_{k+1} = x_k + c_k \quad , \text{ dove } c_k = -\frac{f(x_k)}{f'(x_k)}$$

(la nuova approssimazione è uguale alla vecchia approssimazione più una **correzione**).

Ci **fermeremo** quando $x_k + c_k$ non è diverso da x_k , che si verificherà quando c_k diventerà più piccolo dell'epsilon di x_k . Ciò comporta che $x_k + c_k$ sarà uguale a x_k , e quindi sarebbe inutile calcolarlo.

METODO DI NEWTON COME RISOLUZIONE DI UNA SUCCESSIONE DI PROBLEMI LINEARI (IMPORTANTE)

Il **metodo di newton** può essere utilizzato per risolvere una successione di problemi lineari (lo zero di una funzione lineare, ovvero lo zero di una retta).

Si tratta di un **paradigma** generale che viene utilizzato per risolvere problemi non lineari. La logica è che: invece di risolvere un problema non lineare, si risolve una *successione di problemi lineari* che approssimano il problema lineare di partenza.

Consideriamo il metodo di Newton da un **punto di vista analitico**.

Chiamando $t(x)$ la retta che passa per il punto $(x_k, f(x_k))$ e che ha pendenza $f'(x_k)$, la cui equazione è (per definizione):

$$t(x) = y = f(x_k) + f'(x_k)(x - x_k)$$

dove: $f(x_k)$ è il valore dell'ordinata, $f'(x_k)$ è la pendenza, la quale è moltiplicata per la differenza fra x ed il valore dell'ascissa del punto per cui deve passare la retta.

Il nostro obiettivo ora è quello di **risolvere** $t(x) = 0$.

Effettuando qualche operazione (divido tutto per $f'(x_k)$ e sposto x dall'altra parte) ottengo:

$$x_k - \frac{f(x_k)}{f'(x_k)} = x \quad \rightarrow \quad X_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

E quindi la **soluzione** al problema di $t(x)=0$ è x , che io prendo come successiva approssimazione del metodo iterativo, e che chiamerò quindi x_{k+1} .

LEZIONE 5 – RIS. EQUAZIONE (metodo delle secanti e ibridi)

Metodo delle secanti

Il metodo delle secanti è un algoritmo iterativo **locale** e a velocità di **convergenza superlineare** che risolve il problema $f(x) = 0$; è utilizzabile se f è una funzione continua e se si conosce una buona approssimazione iniziale della soluzione.

Con locale si intende che l'algoritmo non è sempre convergente e che la convergenza dipende da come si sceglie la condizione iniziale. Con velocità di convergenza superlineare si intende che, a ogni passo di iterazione, l'errore relativo si riduce diventando una (proporzionale alla) potenza dell'errore precedente.

Nella convergenza superlineare la potenza è compresa tra 1 e 2. In particolare, nel metodo delle secanti si ha che la potenza è il numero $1.618\dots$, noto come sezione aurea. Si ricorda che nel caso di convergenza quadratica la potenza è 2.

L'**errore** al passo $k+1$ sarà quindi minore o uguale ad una certa costante moltiplicata per l'errore al passo k elevato alla golden ratio.

$$|x_{k+1} - r| \leq C \cdot |x_k - r|^{1.618}$$

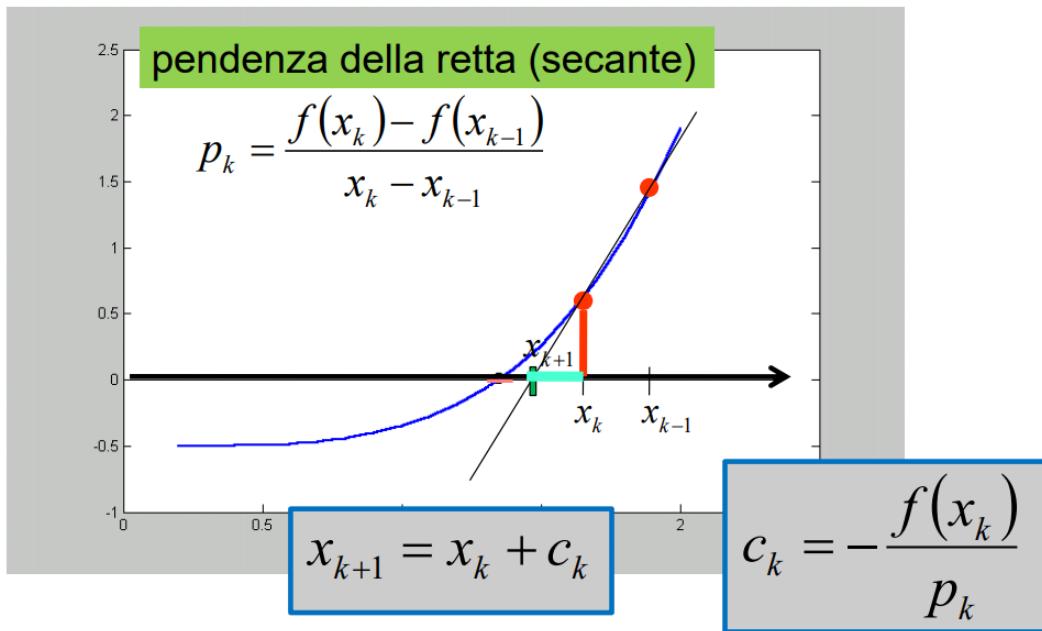
$$\textcolor{blue}{err}_{k+1} \leq C \cdot \textcolor{blue}{err}_k^{1.618}$$

Da un **punto di vista geometrico**, L'algoritmo è basato sull'idea di approssimare linearmente la f , a ogni passo k di iterazione, con la retta che passa per i due punti $(x_k, f(x_k))$ e $(x_{k-1}, f(x_{k-1}))$, il cui coefficiente angolare è $p_k = (f(x_k) - f(x_{k-1})) / (x_k - x_{k-1})$, e di considerare lo zero di tale retta come approssimazione dello zero di f .

Da notare come il metodo delle secanti è una **variante del metodo di Newton**, in quanto usa una retta secante al posto della retta tangente. Il vantaggio sta nel non dover usare la funzione derivata (usa solo i valori di $f(x)$ e approssima $f'(x)$ (usiamo il coefficiente angolare della retta secante p_k)); lo svantaggio è una minore velocità di convergenza.

Ad ogni passo k devo prendere il punto di intersezione della retta secante con l'asse delle x . Tale punto di intersezione rappresenta la successiva approssimazione x_{k+1} .

Al passo successivo $k+1$, considererò la retta che passa per i due punti $(x_{k+1}, f(x_{k+1}))$ e $(x_k, f(x_k))$, e questo per tutti i successivi passi k .



Per poterlo esprimere in **forma ricorrente** useremo la forma ricorrente del metodo di Newton, a cui però sostituiremo $f'(x)$ con p_k .

$$\text{Ovvero: } x_{k+1} = x_k - \frac{f(x_k)}{p_k}$$

$$\text{o anche più semplicemente: } x_{k+1} = x_k + c_k \quad , \text{ dove } c_k = -\frac{f(x_k)}{p_k}$$

(la nuova approssimazione è uguale alla vecchia approssimazione più una **correzione**).

Metodi ibridi

I metodi ibridi combinano in un unico algoritmo un metodo globale e uno (o più) metodi locali. L'obiettivo è di ottenere un algoritmo **globale** con velocità di convergenza **superlineare**.

Due **eempi di metodo ibrido** consistono nel combinare il metodo di bisezione (globale) con il metodo di Newton o delle secanti (locali).

Supponiamo di utilizzare la *combinazione bisezione/Newton* per individuare lo zero di una funzione $f(x)$: la strategia consiste nello stabilire un intervallo $[a,b]$ e una approssimazione iniziale x_k interno all'intervallo.

Per **prima cosa** si cerca di applicare il *metodo di Newton* individuando la retta tangente alla funzione $f(x)$ nel punto x_k e si analizza il punto di intersezione di tale retta con l'asse delle x .

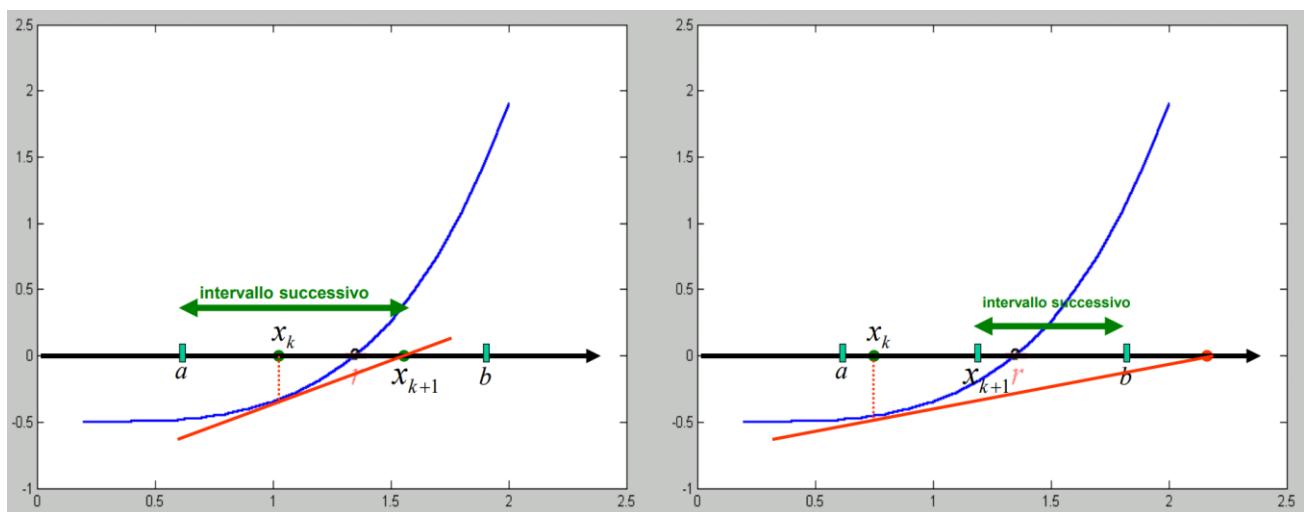
Se il punto d'intersezione appartiene all'intervallo $[a,b]$ del passo k , accetto tale punto come x_{k+1} . Al passo successivo $k+1$ restringo l'intervallo, considerando l'intervallo dove la funzione $f(x)$ cambia segno (es. $[a,x_{k+1}]$ o $[x_{k+1},b]$, quindi x_{k+1} diventa il nuovo a o b).

Se il punto d'intersezione NON appartiene all'intervallo $[a,b]$ del passo k , NON accetto tale punto come nuova approssimazione. Invece, la nuova approssimazione diventerà quella che avrei ottenuto utilizzando il *metodo di bisezione*.

Esempio:

retta tan. appartiene all'intervallo: uso Newton

retta tan. non appartiene all'intervallo: uso bis.



nb: l'algoritmo *fzero* implementato in MATLAB è un algoritmo ibrido.

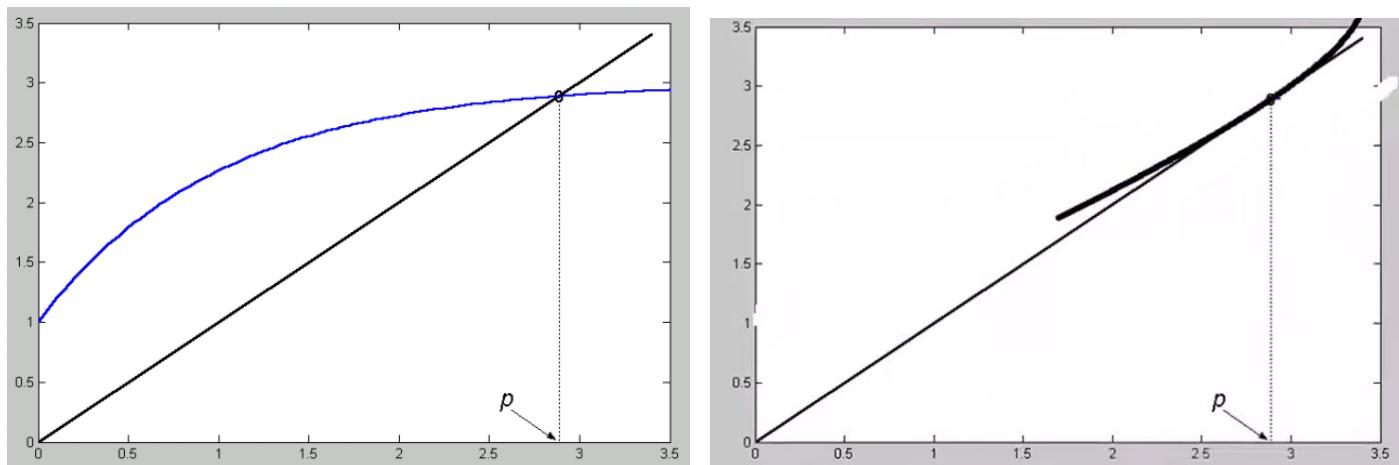
LEZIONE 6 – PROBLEMA DEL PUNTO FISSO ($p = g(p)$)

Il problema del punto fisso è un problema di grande rilevanza nelle applicazioni della matematica, e non deve essere confuso con il problema di $f(x)=0$, cioè dalla risoluzione di una equazione.

Il problema è che: data una funzione nota $g(x)$, trovare (se esiste) un'ascissa p tale che $p = g(p)$.

In tal caso, si dice che p è un **punto fisso** di g .

Dal punto di vista **grafico**, si tratta di trovare un punto sul grafico di g per il quale l'ascissa è **uguale** all'ordinata. Pertanto, tale punto deve necessariamente appartenere alla retta bisettrice del primo (e del terzo) quadrante, cioè alla retta di equazione $y = x$.



Il problema è **ben condizionato** (caso 1) se la pendenza del grafico della g è diversa dalla pendenza della bisettrice vicina al punto fisso.

Ciò implica che leggere variazioni della funzione portano a soluzioni simili.

Il problema è **mal condizionato** (caso 2) se la pendenza del grafico della g è uguale alla pendenza della bisettrice vicina al punto fisso.

Ciò implica che leggere variazioni della funzione portano a soluzioni molto diverse.

(infatti, ricordando che la pendenza della bisettrice è 1, notiamo come la retta tangente della funzione g del primo grafico sia molto diversa dalla pendenza della bisettrice, mentre la retta tangente della funzione g del secondo grafico è molto simile alla pendenza della bisettrice).

Alla luce di queste considerazioni, possiamo stabilire che l'**indice di condizionamento**

sia:

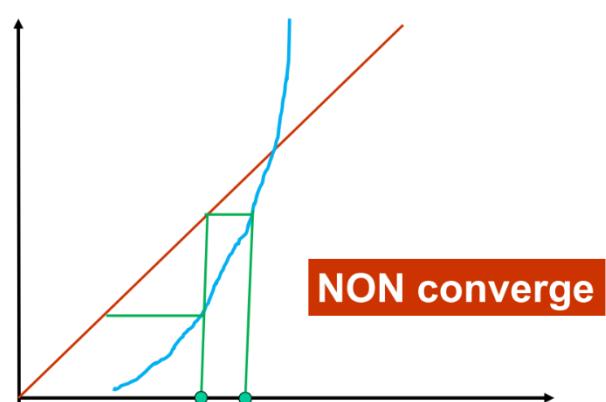
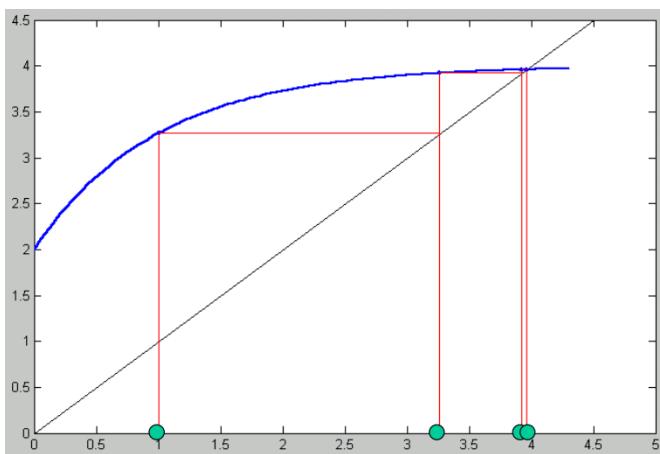
$$K_g = \left| \frac{1}{g'(p)-1} \right|$$

Infatti, se $g'(p) = 1$, il denominatore si annulla e l'indice di condizionamento diventa infinito. (ovvero, il problema potrebbe ammettere infinite soluzioni).

Il metodo del punto fisso

Il problema del punto fisso può essere risolto con il *metodo del punto fisso*, un algoritmo iterativo. A partire da un'approssimazione iniziale x_0 della soluzione, da localizzare per via grafica, il metodo genera una successione di approssimazioni $x_{k+1}=g(x_k)$ che converge alla soluzione (punto fisso) nel caso in cui (condizione sufficiente) $g(x)$ sia una funzione che gode della **proprietà di contrazione** in un intorno della soluzione p .

Dal punto di vista **grafico**, partiamo da x_0 e valutiamo la funzione g nel punto x_0 . Dal punto $g(x_0)$ ci muoviamo in orizzontale fino a toccare la bisettrice: l'ascissa di questo punto d'incontro diventerà la nuova approssimazione x_1 . Al passo successivo ripeteremo il processo, ovvero valutiamo la funzione g nel punto x_1 , individuando la nuova approssimazione x_2 , poi x_3 e così via... notiamo come tale successione di approssimazioni **converga alla soluzione**.



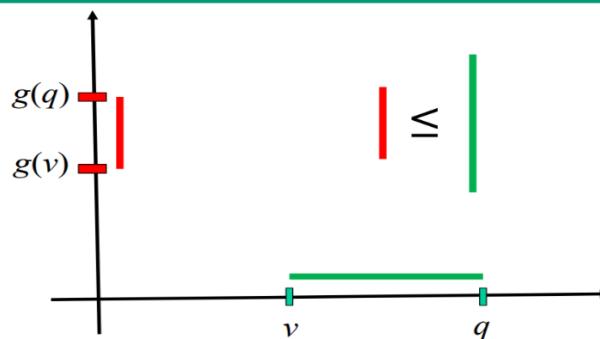
NON è sempre vero che il metodo converge (la funzione deve godere della proprietà di contrazione), quindi si tratta di un metodo **locale**.

(Nell'esempio 2 vediamo come infatti le approssimazioni si allontantino dalla soluzione).

Si dice che g è una **contrazione** in un intervallo I che contiene p se, per ogni coppia di numeri v e q appartenenti a I si ha che la differenza $|v-q|$ è strettamente maggiore della differenza $|g(v)-g(q)|$.

Quindi, diremo che **la funzione g è una contrazione se** trasforma il suo dominio in un codominio che è più piccolo del dominio (per questo si chiama contrazione).

$$\text{contrazione in } I: |g(v)-g(q)| \leq c|v-q| \quad , c < 1, \forall v, q \in I$$



Se la funzione è **derivabile** e ha derivata minore di 1 (in valore assoluto, $|g'(p)| < 1$) in p , allora il metodo del punto fisso **converge** a p , scegliendo x_0 vicino a p .

Per quanto riguarda l'**errore assoluto** al passo $k+1$, sarà dato dalla *approssimazione* al passo $k+1$ *meno* la soluzione p .

$$\textcolor{blue}{err}_{k+1} = |x_{k+1} - p|$$

Tuttavia sappiamo che $x_{k+1} = g(x_k)$, quindi possiamo riscrivere la formula come:

$$\textcolor{blue}{err}_{k+1} = |x_{k+1} - p| = |g(x_k) - g(p)|$$

Questa "forma" è comoda perché ci permette di collegarci facilmente al **teorema del valor medio**, il quale afferma che: dato il grafico di una funzione tra due estremi, esiste almeno un punto in cui la **tangente al grafico è parallela alla secante** passante per gli estremi.

Quindi, avremo che:

$$g(x_k) - g(p) = g'(\xi_k)(x_k - p)$$

teorema del Valor Medio ξ_k tra x_k e p

Da questa formula, notiamo come " $g(x_k) - g(p)$ " è l'errore al passo $k+1$, mentre " $x_k - p$ " è l'errore al passo k . Quindi ho collegato l'errore al passo $k+1$ con quello al passo k attraverso la derivata di g nel punto ξ_k .

$$\textcolor{blue}{err}_{k+1} \leq |g'(\xi_k)| \textcolor{blue}{err}_k$$

Da qui ci rendiamo conto che la convergenza sia **lineare** perché l'esponente di err_k è 1, e la velocità di convergenza dipende dall'andamento della derivata.

Se la derivata nel punto della soluzione è minore di 1 ($|g'(p)| < 1$), e se x_0 è vicino a p , allora esisterà un costante c compresa fra 1 e la derivata di g nel punto ξ_k (proprietà funzioni derivabili).

E allora:

$$\textcolor{blue}{err}_{k+1} \leq c \cdot \textcolor{blue}{err}_k \leq \dots \leq c^k \cdot \textcolor{blue}{err}_0 \quad \text{cioè } \textcolor{blue}{err}_k \rightarrow 0, k \rightarrow \infty$$

(Questo perché abbiamo detto che err_{k+1} è \leq di $c * \text{err}_k$, ma quindi questo vuol dire anche che err_k è \leq di $c * \text{err}_{k-1}$, e a sua volta err_{k-1} è \leq di $c * \text{err}_{k-2}$, e così via, fino ad arrivare ad err_0 .)

Ovvero, poiché c è minore di 1, la quantità c^k per erro₀ tenderà a 0.

Quindi, il fatto che $|g'(p)| < 1$ mi garantisce la convergenza del metodo, e la convergenza è appunto lineare.

Inoltre, notare come questa costante c sia proprio uguale alla derivata $g'(p)$, e quindi al **diminuire** di c **aumenta** la velocità di convergenza.

nb: se $g'(p) = 0$, allora la convergenza del metodo è **quadratica**.

Relazione fra il problema $f(x) = 0$ e il problema del punto fisso

Il legame tra il problema del punto fisso $g(x)=x$ e il problema della risoluzione di una equazione $f(x)=0$ consiste nel fatto che $f(x)=0$ può essere trasformato in un problema di punto fisso: per es., sommando x a entrambi i membri, $f(x)=0$ diventa $f(x)+x=x$, che è un problema di punto fisso per la funzione $g(x)=f(x)+x$.

esempio

$$f(x) = x^2 - x - 2 = 0 \quad r = 2$$

problemi di punto fisso equivalenti:

somma x $x^2 - 2 = x$ $g(x) = x^2 - 2$ diverge $g'(2) = 4$

dividi per x $1 + \frac{2}{x} = x$ $g(x) = 1 + \frac{2}{x}$ converge $g'(2) = -0.5$

somma x^2 $\frac{x^2 + 2}{2x - 1} = x$ $g(x) = \frac{x^2 + 2}{2x - 1}$ converge $g'(2) = 0$

$$p = 2 \qquad g'(x) = \frac{2x(2x-1) - 2(x^2+2)}{(2x-1)^2}$$

(nel primo caso diverge mentre negli altri due converge perché la derivata nel punto 2 deve essere minore di 2).

LEZIONE 7 – Problema del minimo di una funzione

minimi di una funzione

Il problema è detto “minimizzazione di una funzione” (chiamata **funzione obiettivo**) o “determinazione di un punto di minimo”.

Tale problema verrà considerato nell’accezione detta **ottimizzazione non vincolata**, nel senso che noi cerchiamo il minimo di una funzione nota su tutto il suo dominio di definizione.

In particolar modo, il nostro scopo sarà quello di determinare i punti di **minimo locale**.

Dal punto di vista formale, il problema del calcolo del minimo è descritto come:

$$\min_x(f(x)) \quad \arg \min_x(f(x))$$

Ovvero, voglio determinare il valore x , che appartiene al dominio della f , che rende minima la $f(x)$. argmin è una definizione usata meno spesso ma più precisa, in quanto sottolinea il fatto che si è interessati a conoscere l’ascissa in cui la funzione assume il valore minimo.

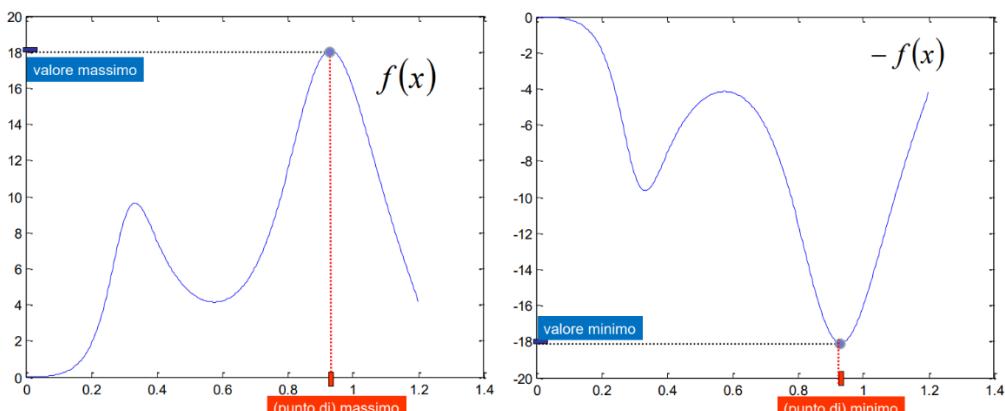
massimi di una funzione

Il problema di “determinazione di un punto di massimo” è praticamente equivalente a quello del minimo. Infatti, posso trasformare il problema del calcolo del massimo in uno del minimo semplicemente “negando” la funzione obiettivo.

(descrizione formale del pmax, e la sua successiva trasformazione nel pmin):

$$\begin{array}{ll} \max_x(f(x)) & \arg \max_x(f(x)) \\ \min_x(-f(x)) & \arg \min_x(-f(x)) \end{array}$$

Per questo motivo, in programmi come MATLAB non si trovano metodi che permettono di calcolare i punti di massimo, ma si devono applicare i metodi per calcolare i punti di minimo passando la funzione negata.



Metodi per determinare un punto di minimo

Se $f(x)$ è derivabile, la **condizione necessaria** affinché un certo punto di f sia un punto di minimo, è che in quel punto si annulli la derivata.

In tal caso, la ricerca dei punti di minimo coincide con i punti in cui si annulla derivata, che è un problema $f'(x)=0$. Quindi il problema della ricerca di un punto di minimo della funzione obiettivo diventa il problema $f'(x)=0$ della *derivata* di tale funzione.

Ergo, se **$f'(x)$ è nota**, possiamo utilizzare un qualunque metodo di risoluzione: bisezione, Newton (se è nota anche $f''(x)$), secanti, ibridi, metodo del punto fisso.

Gli algoritmi (metodi) che abbiamo trattato permettono di determinare **una sola soluzione alla volta**. Quindi, se il problema ammette più soluzioni, l'algoritmo deve essere eseguito tante volte quante sono le soluzioni che si vogliono calcolare.

Inoltre, i metodi presentati non sono in grado di stabilire se un minimo (locale) è un minimo globale della funzione.

Metodo di Newton per la minimizzazione

Se la funzione f è derivabile, allora **il problema $\min f(x)$ è riconducibile al problema $f'(x)=0$** , ovvero al calcolo dello zero della derivata prima di f . Se la funzione f è derivabile due volte, il problema $f'(x)=0$ può essere risolto mediante il metodo di Newton classico.

È opportuno notare che, in tal caso, il metodo di Newton agisce sulla derivata prima e sulla derivata seconda di f , ma non opera direttamente sulla f .

Il metodo di Newton per la minimizzazione è un algoritmo iterativo **locale** e a **velocità di convergenza quadratica**, e può convergere solo linearmente (se nel punto di minimo si annulla anche la derivata seconda).

È utilizzabile se f è una funzione continua e derivabile due volte (con derivata prima e seconda note) e se si conosce una buona approssimazione iniziale della soluzione.

Con locale si intende che l'algoritmo non è sempre convergente e che la convergenza dipende da come si sceglie la condizione iniziale. Con velocità di convergenza quadratica si intende che, a ogni passo di iterazione, l'errore relativo si riduce diventando il (proporzionale al) quadrato dell'errore precedente.

Si noti che non è necessario implementare il metodo di Newton mediante una nuova function Matlab, ma basta passare invece della funzione e la sua derivata prima, la derivata prima e la derivata seconda (altrimenti calcoleremmo lo zero invece del minimo).

Interpretazione geometrica (Newton minim.)

Data l'approssimazione: $x_{k+1} = x_k - \frac{f'(x_k)}{f''(x_k)}$

Ad ogni passo k , dire che la successiva interpretazione x_{k+1} sia uguale a "questo" equivale ad approssimare localmente, a ogni passo, la funzione f con il **polinomio di secondo grado** q , tale che:

$$\begin{aligned} q(x_k) &= f(x_k) \\ q'(x_k) &= f'(x_k) \\ q''(x_k) &= f''(x_k) \end{aligned}$$

Ove una tipico polinomio di secondo grado è del tipo: $q(x) = ax^2 + bx + c$

Queste 3 uguaglianze dicono rispettivamente che:

1. il grafico del polinomio q , nel punto x_k , ha lo stesso **valore** che ha la funzione f nello stesso punto.
2. il grafico del polinomio q , nel punto x_k , ha la stessa **pendenza** che ha la funzione f nello stesso punto.
3. il grafico del polinomio q , nel punto x_k , ha la stessa **curvatura** che ha la funzione f nello stesso punto.

Quindi q , nel punto $(x_k, f(x_k))$ ha lo stesso valore, pendenza e curvatura che ha la funzione nello stesso punto.

Il motivo per cui questa approssimazione q è importante è che per noi è facile calcolare il punto di minimo di una parabola. L'espressione del polinomio q sarà:

$$q(x) = f(x_k) + (x - x_k)f'(x_k) + \frac{1}{2}(x - x_k)^2 f''(x_k)$$

che altro non è che il *polinomio di Taylor di secondo grado*, il quale sviluppo è stato troncato alla potenza 2.

DIM APPROFONDITA...

Detto ciò, l'**interpretazione sarà quindi che**: il metodo di Newton ad ogni passo approssima localmente la funzione con un polinomio di secondo grado, calcola il minimo di quel polinomio di secondo grado, e tale minimo diventerà la nuova approssimazione.

Quindi, invece di risolvere una successioni di problemi lineari, risolverà una **successione di problemi quadratici** (minimizzazione di un polinomio di secondo grado).

Varianti del metodo di Newton

Possiamo distinguere 3 diverse varianti:

variante 1

Approssimazione della derivata prima e seconda (con differenze finite). Ha ispirato anche il metodo delle secanti.

La derivata prima può essere approssimata con la pendenza della retta che passa per due punti, mentre la derivata seconda può essere approssimata con la pendenza della retta che passa per *tre* punti.

variante 2

Determinazione (a ogni passo k) del minimo del polinomio di secondo grado interpolante su x_k, x_{k-1}, x_{k-2} .

Può essere vista come una generalizzazione del metodo di minimizzazione di Newton, dove, invece di utilizzare il polinomio di 2 grado di Taylor, si utilizza un polinomio che passa per 3 punti, ovvero $(x_k, f(x_k)), (x_{k-1}, f(x_{k-1}))$ e $(x_{k-2}, f(x_{k-2}))$.

Il minimo di questo polinomio diventerà la successiva approssimazione.

variante 3

Utilizzo dei metodi **damped** (molto utilizzati nel campo scientifico).

Ogni qualvolta abbiamo un metodo iterativo che esprime l'approssimazione da calcolare in una forma del tipo: $x_{k+1} = x_k + c_k$; i metodi damped prevedono che la correzione c_k venga moltiplicata per una costante a_k .

L'equazione diventerà quindi: $x_{k+1} = x_k + a_k c_k$

Il parametro a_k è scelto ad ogni passo in modo tale da **evitare** che la nuova approssimazione sia troppo lontana dalla precedente, e la tecnica utilizzata per la selezione è "a tentativi".

In particolare, il numero a_k è scelto ad ogni passo in modo che: $|f(x_{k+1})| < |f(x_k)|$.

Questo perché stiamo effettuando la ricerca del minimo, e quindi mi aspetto che il valore del punto all'approssimazione successiva sia minore rispetto a quella attuale.

L'idea è partire da $a_k = 1$, e se tale condizione non è verificata, si considera $a_k = 1/2$, $a_k = 1/4$, $a_k = 1/8\dots$ e così via, per un numero prefissato di tentativi.

Metodo del gradiente discendente (minimizzazione)

Si tratta di uno dei metodi più utilizzati al mondo per quanto riguarda il campo dell'IA.

Prima di parlare del metodo, ricordiamo le **derivate**, che altro non sono che il limite del rapporto incrementale (che può essere visto come una differenza finita).

Notiamo come se la derivata è *positiva* (va verso le x crescenti), per andare verso il punto di minimo devo andare nella direzione opposta, ovvero verso le x decrescenti, questo perché si allontana dal punto di minimo.

Viceversa, in modo simile, se la derivata è *negativa* (va verso le x decrescenti), per andare verso il punto di minimo devo andare nella direzione opposta, ovvero verso le x crescenti.

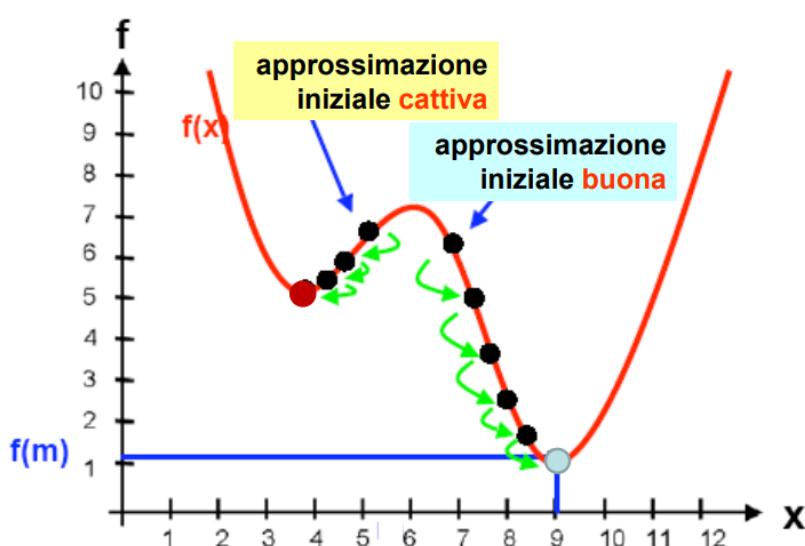
Questa è l'**idea** che sta dietro al **metodo del gradiente discendente**, dove ad ogni passo calcolo una nuova approssimazione e mi sposto all'indietro di tanto quanto è il valore della derivata corrente.

Tale metodo è descritto dall'**equazione**: $X_{k+1} = X_k - \alpha \cdot f'(X_k)$

ovvero, la nuova approssimazione è una correzione della vecchia approssimazione ottenuta andando nel verso opposto al segno della derivata di $f(x_k)$.

Mi sposto di una quantità che è proporzionale ad un fattore di proporzionalità α (learning rate) al valore della derivata nel punto x_k .

Nota bene che si deve avere una buona approssimazione iniziale, cioè la funzione deve essere convessa. Infatti, il rischio è che se non scegliamo una buona approssimazione iniziale potremmo finire per calcolare un punto di minimo diverso da quello che avevamo intenzione di calcolare.



Ricordando che:

- una funzione è **convessa** (concavità verso l'alto) se, comunque presi due punti sul suo grafico, il segmento che li unisce sta sopra, o al limite coincide, con il grafico della funzione.
Un insieme è **convesso** se, per ogni coppia di punti, il segmento che li unisce è completamente contenuto nell'insieme.
- una funzione è **concava** se (concavità verso il basso) se, comunque presi due punti sul suo grafico, il segmento che li unisce sta sotto, o al limite coincide, con il grafico della funzione.
Un insieme è **non convesso** se esiste almeno una coppia di punti il quale segmento che li unisce NON è completamente contenuto nell'insieme.

Per una **funzione convessa**, un minimo locale è un minimo globale, mentre per una **funzione non convessa** un minimo locale può non essere un minimo globale .

Funzione predefinita Matlab: fminbnd

La function Matlab fminbnd calcola il punto di minimo di una funzione.

La function fminbnd ha tre parametri di input: il primo è l'handle alla function Matlab che implementa la funzione f ; il secondo e il terzo argomento sono, rispettivamente, l'estremo sinistro e l'estremo destro di un intervallo che contiene la soluzione.

L'accuratezza di default è 10^{-6} . Se la volessimo aumentare dovremmo inserire un quarto metodo *optimset('TolX', nuova_acc)*.

Es. `>> xminimo = fminbnd(f,-2.5,-1,optimset('TolX',1e-14))`

LEZIONE 8 – Calcolo del minimo di una funzione unimodale

Una funzione $f(x)$ è **unimodale** in $[a,b]$ se esiste un **unico** numero p in $[a,b]$ tale che $f(x)$ è **strettamente decrescente** in $[a,p]$ ed è **strettamente crescente** in $[p,b]$, ovvero sono funzioni che hanno un **unico** punto di minimo.

Se la f è **unimodale** in un intervallo $[a,b]$, allora esistono algoritmi per il calcolo del minimo che godono della proprietà di **racchiudere la soluzione a ogni passo in un intervallo**, la cui ampiezza decresce all'aumentare dei passi. Si tratta di algoritmi di tipo *divide et impera*, basati su un'idea simile a quella dell'algoritmo di bisezione per $f(x)=0$.

È facile rendersi conto che se f è unimodale in $[a,b]$, considerando due qualunque numeri c,d appartenenti a $[a,b]$ e tali che $c < d$, allora è sicuramente possibile considerare un nuovo intervallo, incluso in $[a,b]$, che contiene la soluzione: infatti se $f(c) > f(d)$, allora la soluzione si trova certamente in $[c,b]$; mentre se $f(c) < f(d)$, allora la soluzione si trova certamente in $[a,d]$.

Possiamo utilizzare diversi metodi per determinare c,d ad ogni passo: il metodo più semplice consiste nel *generarli a caso*, o possiamo utilizzare il metodo della *ricerca di Fibonacci* o il metodo della *ricerca Aurea*, migliori in quanto hanno un miglior criterio per la scelta di c,d .

In particolar modo, diremo che c e d sono stati **scelti BENE** se il criterio di scelta utilizzato ci garantisce che la funzione obiettivo sia calcolata una sola volta (invece di due), risultando così molto più efficiente.

Inoltre, il nuovo intervallo dovrà essere il più piccolo intervallo che racchiude il minimo ($[a,d]$ o $[c,b]$) e il punto c o d deve essere un punto già considerato al passo precedente.

Metodo di Fibonacci

Il metodo di Fibonacci fa ovviamente utilizzo dei numeri di Fibonacci, i quali sono generati dalla formula ricorrente: $F_n = F_{n-1} + F_{n-2}$, dove $F_0 = 0$, $F_1 = 1$.

Innanzitutto si sceglie un intero N . Possiamo pensare di calcolare i numeri di Fibonacci attraverso una funzione “fibonacci”, e dovremo considerare tutti i numeri di Fibonacci che vanno da $\text{fibonacci}(N)$, $\text{fibonacci}(N-1)$, ..., $\text{fibonacci}(2)$, $\text{fibonacci}(1)$.

Calcoleremo c e d come:

$$c = a + \frac{\text{fibonacci}(N-2)}{\text{fibonacci}(N)} * (b-a)$$

$$d = a + \frac{\text{fibonacci}(N-1)}{\text{fibonacci}(N)} * (b-a)$$

Se $f(c) < f(d)$ il nuovo intervallo sarà $[a,d]$, mentre se $f(c) \geq f(d)$ il nuovo intervallo sarà $[c,b]$.

Notare come il *rapporto* in entrambi le assegnazione sia un numero < 1 . Questo è dovuto ad una **proprietà dei numeri di Fibonacci** per cui $F_n > F_{n-1} > \dots > F_1$, infatti essendo il nominatore sempre più piccolo rispetto al denominatore, tale rapporto darà sempre come risultato un numero < 1 .

Inoltre, poiché $\text{fibonacci}(N-2) < \text{fibonacci}(N-1)$ **implica certamente che $c < d$** .

Al **passo successivo**, non avremo più:
 fibonacci(N-2)/fibonacci(N) e fibonacci(N-1)/fibonacci(N),
ma fibonacci(N-3)/fibonacci(N-1) e fibonacci(N-2)/fibonacci(N-1).

Quindi al passo successivo andremo a considerare il numero di Fibonacci precedente, e **così via** fino al passo n-simo. Notare come anche in questo caso $c < d$ sempre, per la proprietà dei numeri di F.

Al **primo passo** l'intervallo che conteneva la soluzione diventa un intervallo di ampiezza $\text{fibonacci}(N-1)/\text{fibonacci}(N) * (b-a)$ mentre al **passo successivo** l'intervallo avrà ampiezza $\text{fibonacci}(N-2)/\text{fibonacci}(N-1)$ **del precedente**, e così via fino al passo n-simo.

I due nuovi intervalli $[a,d]$ e $[c,b]$ hanno **uguale ampiezza**.

DIM

Dimostriamo che i due nuovi intervalli hanno uguale ampiezza, ovvero che $d-a = b-c$, che è la stessa cosa di dire:

$$d - a = (b - a) \frac{F_{N-1}}{F_N}$$

Adesso consideriamo l'assegnazione di c:

$$c = a + \text{fibonacci}(N-2)/\text{fibonacci}(N) * (b-a)$$

a cui moltiplicherò entrambi i membri per -1 ed aggiungerò b , e **metto (b-a) in evidenza** ottenendo:

$$b - c = (b - a) - (b - a) \frac{F_{N-2}}{F_N} = (b - a) \left(1 - \frac{F_{N-2}}{F_N} \right)$$

che è uguale a:

$$(b - a) \frac{F_N - F_{N-2}}{F_N}$$

Ricordiamo la formula di Fibonacci: $F_n = F_{n-1} + F_{n-2}$, possiamo osservare quindi che in realtà $F_n - F_{n-2} = F_{n-1}$, e quindi posso riscrivere l'equazione come:

$$(b - a) \frac{F_{N-1}}{F_N}$$

Notare quindi come $d-a = b-c$, quindi **abbiamo dimostrato** che entrambi gli intervalli hanno la stessa ampiezza.

Per la precisione, l'**intervallo finale** sarà rappresentato dalla **successione**:

$$[b-a] \frac{F_{N-1}}{F_N} \frac{F_{N-2}}{F_{N-1}} \frac{F_{N-3}}{F_{N-2}} \dots \frac{F_2}{F_3} \frac{F_1}{F_2}$$

ma possiamo **semplificare** diagonalmente, quindi l'**intervallo finale** sarà:

$$[b-a] \frac{1}{F_N}$$

Dopo N passi, l'ampiezza dell'intervallo iniziale $[a,b]$ si riduce di un fattore $1/F_N$.

Ovviamente noi vogliamo che il nostro intervallo abbia un'ampiezza minore di **delta**, e per far ciò dobbiamo trovare una **N opportuna**.

Fissato delta, allora N deve essere tale che:

$$[b-a] \frac{1}{\delta} < F_N \approx \phi^N$$

Ovvero, Dopo N passi, l'ampiezza dell'intervallo iniziale $[a,b]$ si riduce di un fattore circa uguale a $1/\Phi^n$ (1/(sezione aurea)).

L'algoritmo di Fibonacci search è **globale** e con **velocità di convergenza lineare**.

OSSERVAZIONE

Al passo successivo, il *nuovo c* o il *nuovo d* coincidono certamente o con il *vecchio d* o con il *vecchio c*, consentendo di risparmiare una valutazione della funzione f a ogni passo (è necessario cioè effettuare 1 valutazione, invece di 2).

DIM

Tenendo in considerazione che:

$$c = a + [b-a] \frac{F_{N-2}}{F_N}, \quad d = a + [b-a] \frac{F_{N-1}}{F_N}$$

supponiamo che il nuovo intervallo sia $[a,d]$, noteremo come il *nuovo d* sia proprio uguale al *vecchio c*: (ovviamente, dimostriamo in modo analogo anche che il *nuovo c* = *vecchio d*)

$$d_{nuovo} = a + [d-a] \frac{F_{N-2}}{F_{N-1}} = a + [b-a] \frac{\cancel{F_{N-1}}}{F_N} \frac{F_{N-2}}{\cancel{F_{N-1}}} = c$$

Metodo della Sezione Aurea

Ricordiamo che la **sezione aurea** è un numero Φ (phi) dato da:

$$\phi = \lim_{n \rightarrow \infty} \frac{F_n}{F_{n-1}} = 1.61803398874989\dots$$

★ $\phi = \frac{1 + \sqrt{5}}{2}$

proprietà di ϕ

$$\phi - 1 = \frac{1}{\phi}$$

$$F_n = \frac{\phi^n - (1-\phi)^n}{\sqrt{5}}$$

osservazioni (in ordine):

- 1- Il rapporto è certamente maggiore di 1 perché $F_n > F_{n-1}$ (proprietà numeri di Fibonacci);
- 2- Il limite di tale rapporto è un numero irrazionale Φ che può anche essere espresso come *.
- 3- Φ è l'unico numero positivo che gode della seguente **proprietà**: $\Phi - 1$ è uguale all'inverso di Φ .
- 4- Un ulteriore proprietà afferma che possiamo utilizzare Φ per calcolare i numeri di Fibonacci, e per n sufficientemente grande, **tal equazione è circa uguale a Φ^n** .

Il **metodo della Sezione Aurea** è una variante del metodo di Fibonacci.

Consiste nel ridurre, a ogni passo, di un fattore costante l'ampiezza dell'intervallo che contiene la soluzione. Tale fattore è il numero $\Phi - 1 = 0.618$.

Dopo ogni passo, l'**intervallo** si riduce di un fattore $1/\Phi$. Dopo N passi l'ampiezza dell'intervallo iniziale $[a,b]$ si riduce di un fattore $1/\Phi^n$.

Il motivo per cui consideriamo tale valore è dovuto al fatto che:

$$\lim_{n \rightarrow \infty} \frac{F_n}{F_{n-1}} = \phi \quad 1.6180\dots$$

sezione aurea
(golden section)

$$\lim_{n \rightarrow \infty} \frac{F_{n-1}}{F_n} = \frac{1}{\phi} = \phi - 1 \quad 0.6180\dots$$

Il motivo per cui questo è **considerato una variante** è collegato al modo con cui calcoliamo c e d .

$$c = a + \frac{\text{fibonacci}(N-2)}{\text{fibonacci}(N)} * (b-a)$$

$$d = a + \frac{\text{fibonacci}(N-1)}{\text{fibonacci}(N)} * (b-a)$$

Nel **caso di d**, notiamo come $\frac{\text{fibonacci}(N-1)}{\text{fibonacci}(N)}$ sia proprio uguale a $\Phi - 1$.

Nel **caso di c**, basterà solo fare qualche passaggio in più, infatti:

$$\frac{F_{n-2}}{F_n} = \frac{F_n - F_{n-1}}{F_n} = 1 - \frac{F_{n-1}}{F_n}$$

$$\lim_{n \rightarrow \infty} \frac{F_{n-2}}{F_n} = \lim_{n \rightarrow \infty} \left(1 - \frac{F_{n-1}}{F_n} \right) = 1 - (\phi - 1) = 2 - \phi \quad 0.3819\dots$$

Quindi, ci basterà sostituire quei due rapporti con queste due nuove costanti, ed avremo dunque:

$$c = a + (2-\phi) * (b-a)$$

$$d = a + (\phi-1) * (b-a)$$

Anche l'algoritmo Golden search garantisce che al passo successivo, il *nuovo c* o il *nuovo d* **coincidono** certamente o con il *vecchio d* o con il *vecchio c*, consentendo di risparmiare una valutazione della funzione f a ogni passo (è necessario cioè effettuare 1 valutazione, invece di 2).

DIM

$$c = a + (b-a)(2-\phi), \quad d = a + (b-a)(\phi-1)$$

supponiamo che il nuovo intervallo sia $[a,d]$



$$d_{\text{nuovo}} = a + (d-a)(\phi-1) = a + (b-a)(\phi-1)^2 = a + (b-a)(2-\phi) = c$$

$$(\phi-1)^2 = \frac{(\phi-1)}{\phi} = 1 - \frac{1}{\phi} = 1 - (\phi-1) = 2 - \phi$$

LEZIONE 9 – Richiami di algebra lineare

VETTORI

Un vettore è un segmento orientato caratterizzato da una *direzione*, un *verso* e un *modulo*.

$$\begin{array}{ll} \text{vettore (colonna)} \color{red}{x} & x = \begin{pmatrix} 1.1 \\ -3.2 \\ \vdots \\ 0.9 \end{pmatrix} \quad x \in R^n \quad x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \\ \text{vettore (riga)} \color{red}{x^T} & x^T = (1.1 \quad -3.2 \quad \dots \quad 0.9) \end{array}$$

Con $x \in R^n$ intendiamo che il vettore x ha n componenti.
Un vettore riga si ottiene *trasponendo* un vettore colonna.

Operazioni di base

scalare per vettore

Dati un vettore x (di n componenti) ed un numero (scalare) α , otteniamo un nuovo vettore y (di n componenti) come il prodotto fra il vettore x e lo scalare α .

Più precisamente, la componente i -esima di y sarà data dal prodotto fra la componente i -esima di x e α .

$$\begin{array}{l} x \in R^n, \alpha \in R \quad \boxed{y = \alpha x \quad \in R^n} \\ y_i = \alpha x_i \quad i = 1:n \end{array}$$

somma di due vettori

Dati due vettori x e y , i quali devono avere lo **stesso numero n** di componenti, otteniamo un nuovo vettore z (di n componenti) sommando l' i -esima componente di x con la i -esima componente di y .

$$\begin{array}{l} x, y \in R^n \quad \boxed{z = x + y \quad \in R^n} \\ z_i = x_i + y_i \quad i = 1:n \end{array}$$

nb: regola del parallelogramma

somma saxpy

Si tratta della somma di un multiplo di un vettore a un vettore (è come se fosse una fusione).

$$x, y \in R^n , \alpha \in R$$

$$z = \alpha x + y \in R^n$$

$$z_i = \alpha x_i + y_i \quad i = 1:n$$

prodotto scalare di due vettori

Prodotto scalare in quanto il risultato è uno scalare.

Si calcola facendo il prodotto delle componenti di egual posto dei due vettori, e poi si sommano questi prodotti.

$$x, y \in R^n$$

$$s = x^T y \in R$$

$$s = \sum_{i=1}^n x_i y_i$$

The diagram illustrates the calculation of the dot product $s = x^T y$. It shows two vectors, x and y , represented by arrows pointing from the origin. The vector x is shown with its components x_1, x_2, \dots, x_n along its path. The vector y is shown with its components y_1, y_2, \dots, y_n along its path. The resulting scalar s is represented by a small red square at the end of the calculation process.

nb: $x^T y = y^T x$ (commuta)

prodotto esterno di due vettori

Un prodotto si esterno si ha quando si effettua un prodotto scalare tra due vettori x e y , ma il vettore trasposto è y . In tal caso il risultato non sarà uno scalare ma una matrice M .

Da notare inoltre che non è più necessario che i due vettori abbiano lo stesso numero di componenti.

La matrice avrà tante righe quante sono le componenti di x , e tante colonne quante sono le componenti di y .

Inoltre, la matrice risultato avrà per **colonne** i *multipli* del vettore x ottenuti scalandoli per le componenti del vettore y .

Tuttavia è anche vero che la matrice risultato avrà per **righe** i *multipli* del vettore y ottenuti scalandoli per le componenti del vettore x (abbiamo solo cambiato il punto di vista).

Fondamentalmente, quello che stiamo dicendo è che nel primo caso abbiamo che *ogni colonna di y moltiplica il vettore x* , e quindi abbiamo una situazione del tipo: y_1x, y_2x, \dots

Nel secondo caso abbiamo che *ogni riga di x moltiplica il vettore y* , e quindi abbiamo una situazione del tipo: yx_1, yx_2, \dots

Quindi la **matrice M** può essere **costruita per righe o per colonne**.

$$x \in R^m, y \in R^n$$

$$M = xy^T \in R^{m \times n}$$

$$M_{ij} = x_i y_j, \quad i=1:m, j=1:n$$



MATRICI

Una matrice è una tabella in cui sono riportati in modo ordinati gli elementi di un dato insieme.

La sua *trasposta* sarà una nuova matrice che si ottiene scambiando le righe con le colonne.

Una matrice può essere *partizionata* per **colonne** o per **righe**:

- per colonne: $A = (a_{\cdot 1}, a_{\cdot 2}, \dots, a_{\cdot n})$
- per righe: $A = (a_1, a_2, \dots, a_n)$

Una matrice può essere anche partizionata a blocchi.

Caratteristiche strutturali

Quando una matrice è costituita da moltissimi elementi, potrebbe essere conveniente sapere se la maggior parte degli elementi della matrice sia nulla o meno. In particolar modo distinguiamo due tipi di matrici:

- **matrici piene**: la maggior parte degli elementi sono *diversi* da zero.
- **matrici sparse**: la maggior parte degli elementi sono *uguali* a zero.

È possibile classificare una matrice utilizzando un **indice di sparsità**:

$$\text{indice di sparsità} = \frac{\text{numeri di elementi nulli}}{\text{numero totale di elementi}}$$

L'indice di sparsità sarà 0 se tutti gli elementi NON sono nulli, mentre sarà 1 se ci sono solo elementi nulli.

nb: il numero di elementi nulli è dato dal numero totale di elementi ($m \cdot n$) meno il numero di elementi diversi da zero (nnz, number of nonzeros).

$$sp = \frac{(m \cdot n) - nnz}{m \cdot n} \quad sp \approx 0, \text{ piena} \quad sp \approx 1, \text{ sparsa}$$

Possiamo distinguere vari **tipi comuni** di matrice, quali:

- **diagonali**: gli elementi extradiagonali sono nulli;
- **tridiagonali**: sono nulli tutti gli elementi al di fuori della diagonale principale e della diagonale immediatamente sopra e sotto;
- **triangolari sup**: sono nulli gli elementi del triangolo strettamente inferiore;
- **triangoli inf**: sono nulli gli elementi del triangolo strettamente superiore;
- **di Toeplitz**: elementi uguali sulle diagonali;
- **tridiagonali di Toeplitz**: combina tridiagonale con Toeplitz.

$D = \begin{pmatrix} x & 0 & 0 & 0 \\ 0 & x & 0 & 0 \\ 0 & 0 & x & 0 \\ 0 & 0 & 0 & x \end{pmatrix}$	$T = \begin{pmatrix} x & x & 0 & 0 \\ x & x & x & 0 \\ 0 & x & x & x \\ 0 & 0 & x & x \end{pmatrix}$	$U = \begin{pmatrix} x & x & \dots & x \\ 0 & x & \dots & x \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & x \end{pmatrix}$
matrice diagonale $a_{ij} = 0, i \neq j$	matrice tridiagonale $a_{ij} = 0, i - j > 1$	matrice triangolare superiore $a_{ij} = 0, i > j$
$L = \begin{pmatrix} x & 0 & \dots & 0 \\ x & x & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ x & x & \dots & x \end{pmatrix}$	$V = \begin{pmatrix} d & e & f & g \\ c & d & e & f \\ b & c & d & e \\ a & b & c & d \end{pmatrix}$	matrice Tridiagonale di Toeplitz $\begin{pmatrix} 2 & -1 & 0 & 0 \\ 1 & 2 & -1 & 0 \\ 0 & 1 & 2 & -1 \\ 0 & 0 & 1 & 2 \end{pmatrix}$
$a_{ij} = 0, i < j$	elementi uguali sulle diagonali	combinazione di proprietà elementari

Una **matrice sparsa non strutturata** è una matrice nella quale compaiono molti zeri ma questi non compaiono in posizioni che sono facilmente descrivibili (compaiono a caso), e quindi non è possibile individuarne la struttura.

Per questo tipo di matrice viene utilizzata la **memorizzazione a coordinate**, con la quale si memorizzano 3 vettori: un vettore che contiene tutti gli elementi *non nulli*, e due vettori che contengono gli indici degli elementi (uno per gli indici di colonna ed uno per quelli di riga).

L'obiettivo è quello di **NON memorizzare gli 0**.

$A = \begin{pmatrix} 6. & 0. & 0. & 3. & 0. \\ 4. & 1. & 0. & 0. & 2. \\ 7. & 0. & 6. & 0. & 8. \\ 0. & 0. & 9. & 5. & 0. \\ 0. & 0. & 0. & 0. & 3. \end{pmatrix}$	matrice sparsa non strutturata
	memorizzazione formato a coordinate
	è usato da MATLAB per la gestione delle matrici sparse
per righe	
$\left\{ \begin{array}{l} A_{\text{nonzero}} [6. \ 3. \ 4. \ 1. \ 2. \ 7. \ 6. \ 8. \ 9. \ 5. \ 3.] \\ \text{ind_col} [1 \ 4 \ 1 \ 2 \ 5 \ 1 \ 3 \ 5 \ 3 \ 4 \ 5] \\ \text{ind_riga} [1 \ 1 \ 2 \ 2 \ 2 \ 3 \ 3 \ 3 \ 4 \ 4 \ 5] \end{array} \right.$	nnz

Operazioni di base

scalare per matrice (multiplo di una matrice)

Il multiplo di una matrice si ha quando si moltiplica una matrice per uno scalare, e si ottiene una nuova matrice dove ogni componente è stata moltiplicata per quello scalare.

$$A \in R^{m \times n}, \alpha \in R$$

$$B = \alpha A \quad \in R^{m \times n}$$

$$b_{ij} = \alpha a_{ij}, i=1:m, j=1:n$$

somma di due matrici

La somma di due matrici si ottiene sommando gli elementi di ugual posto. Le matrici devono avere le stesse dimensioni.

$$A, B \in R^{m \times n}$$

$$C = A + B \quad \in R^{m \times n}$$

$$c_{ij} = a_{ij} + b_{ij}, i=1:m, j=1:n$$

prodotto matrice per vettore (IMPORTANTE)

Il prodotto matrice per vettore si ottiene moltiplicando una matrice qualsiasi ad un vettore **colonna**, il quale deve avere tante componenti quante sono le colonne della matrice.

nb: basta ricordare $m \times n^* \underline{n} \times 1$, dove i numeri interni devono essere uguali.

$$A \in R^{m \times n}, v \in R^n$$

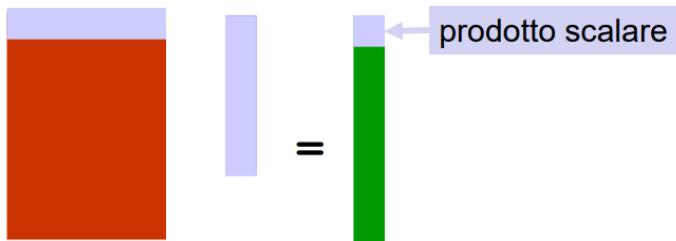
$$z = Av \quad \in R^m$$

$$z_i = \sum_{j=1}^n a_{ij} v_j, i=1:m$$

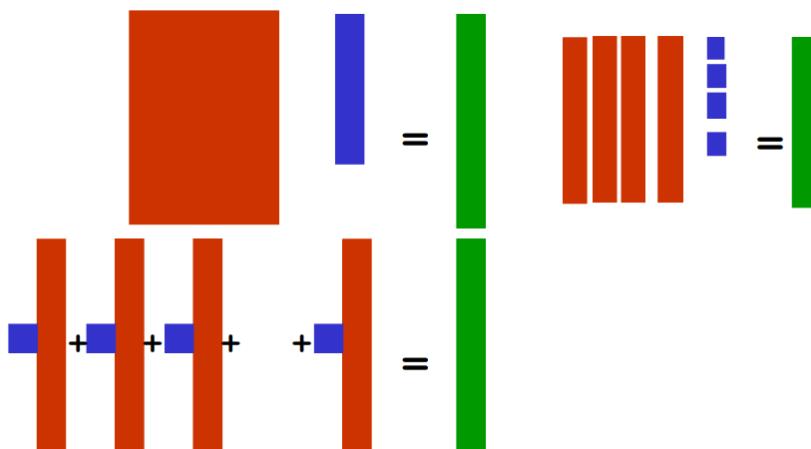
Effettuiamo un prodotto scalare fra la riga i -esima di A e il vettore v , ed otteniamo un nuovo vettore **colonna** z che avrà m componenti.

Questo prodotto è così importante perché è legato alla **trasformazione** di un vettore, e prende il nome di trasformazione lineare.

rappresentazione del primo passaggio: facciamo la somma fra il prodotto della prima riga della matrice con il vettore colonna, poi fra la seconda riga e il vettore colonna, poi la terza, ecc...



Da notare come il vettore risultato è una **combinazione lineare** delle colonne della matrice A.



Prodotto matrice per matrice

Il prodotto matrice per matrice si ottiene moltiplicando due matrici che rispettano il criterio di congruenza delle dimensioni (il numero di colonne della prima matrice = numero righe della seconda).

nb: basta ricordare $m \times n * n \times p$, dove i numeri interni devono essere uguali.

$$A \in R^{m \times n}, B \in R^{n \times p}$$

$$C = AB \in R^{m \times p}$$

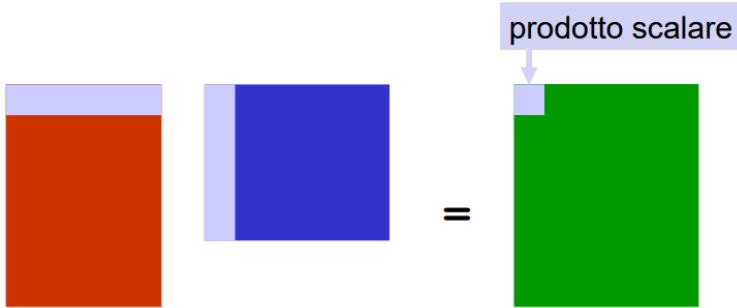
$$c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}, \quad i=1:m, j=1:p$$

Effettuiamo il prodotto scalare della i-esima riga di A per la j-esima colonna di B.
Il risultato sarà una nuova matrice C che avrà $m \times p$ componenti.

Il prodotto matrice x matrice può essere **INTERPRETATO** in 3 modi:

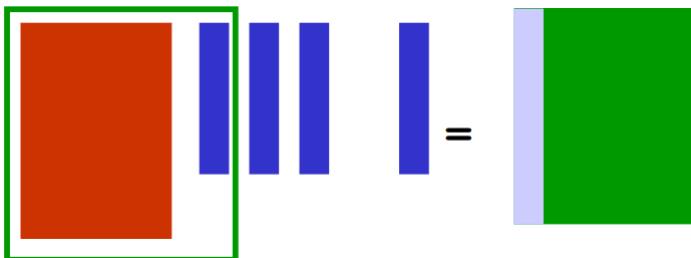
Successione di prodotti scalari di due vettori

Facciamo il prodotto scalare di due vettori: la riga i-esima di A e la colonna j-esima di B. Ripeterà il processo per tutte le righe rimanenti di A. (o colonne di B)



Successione di prodotti matrice x vettore

Ogni **colonna** della matrice C è il prodotto della matrice A per una colonna di B (si può vedere come una serie di *prodotti matrice x vettore*).

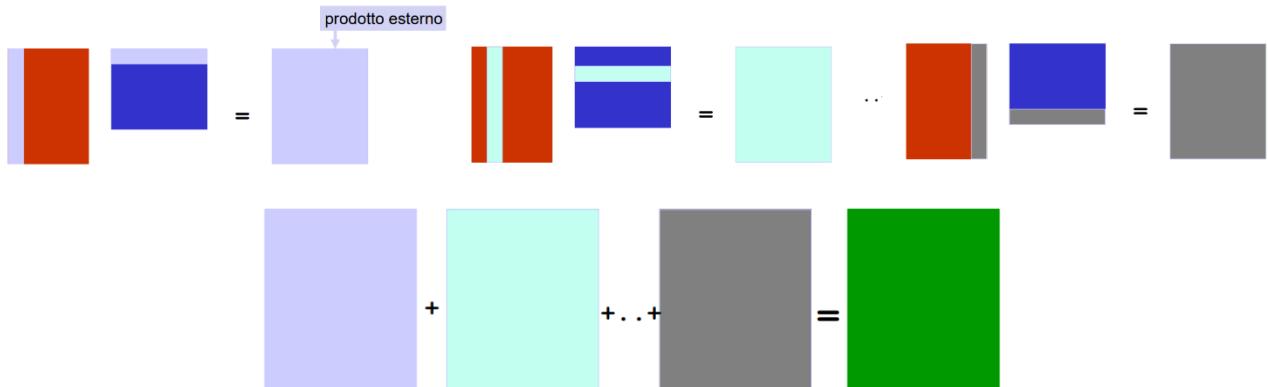


Successione di prodotti esterni

Facciamo il prodotto esterno fra due vettori: la colonna j-esima A e la riga i-esima di B (il risultato sarà una matrice).

Ripeterò il processo per tutte le colonne rimanenti di A (o righe di B).

La matrice risultato è la **somma di n matrici**, ognuna ottenuta come **prodotto esterno** di una colonna di A per una riga di B.



Trasformazione lineare

Una *trasformazione lineare* (prodotto matrice per vettore) è una trasformazione di un vettore in un altro vettore.

Una trasformazione lineare è chiamata “lineare” perché gode della proprietà di conservare il risultato di una **somma di due vettori** e di una **moltiplicazione di uno scalare per un vettore** (multiplo di un vettore).

Cioè, consideriamo due vettori v ed r , dove v viene trasformato in w , e r viene trasformato in s . La comodità stà nel fatto che se dobbiamo effettuare una trasformazione di più vettori, possiamo calcolarla per pezzi (calcoliamo la trasformazione dei singoli vettori).

$$v \rightarrow w \quad r \rightarrow s \\ (v + \alpha r) \rightarrow w + \alpha s$$

Si osserva quindi come i problemi lineari siano più semplici in quanto possono essere scomposti, cosa che non potremmo fare con un problema non lineare.

tipi di trasformazioni lineari:

- **rotazione**: l'immagine viene ruotata in senso orario o antiorario;
- **riflessione** (rispetto a un asse);
- **scalatura**: l'immagine viene allungata sull'asse x o sull'asse y.
- **shearing**: lascia inalterati tutti i punti su un asse; tutti gli altri punti sono traslati parallelamente all'asse in modo proporzionale alla loro distanza dall'asse (conserva le aree).

Quello che accadrà in linea generale è che **moltiplicheremo** un **vettore** (x,y) (che rappresenta un generico punto) con una **matrice** che sintetizza una certa informazione (es. rotazione), ed otterremo un **nuovo vettore** (x',y') .





rotazione
oraria

ogni punto (pixel) (x,y)
dell'immagine a sinistra è
trasformato in un punto (x',y')
dell'immagine a destra

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

rotazione
oraria

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

rotazione
antioraria



ogni punto (pixel) (x,y)
dell'immagine a sinistra è
trasformato in un punto (x',y')
dell'immagine a destra

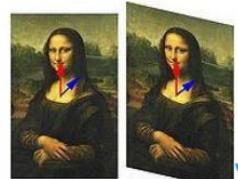
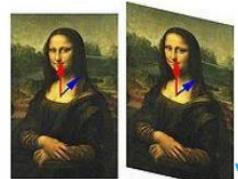
scalatura

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

$$\begin{aligned} x' &= s_x x \\ y' &= s_y y \end{aligned}$$

$s_x > 1, s_y < 1$

$s_x < 1, s_y > 1$



shearing

Lascia inalterati tutti i punti su un asse; gli altri
punti sono traslati parallelamente all'asse in
modo proporzionale alla loro distanza dall'asse
(conserva le aree)

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & k \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

shearing parallelo all'asse x

$$x' = x + ky$$

$$y' = y$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ k & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

shearing parallelo all'asse y

$$x' = x$$

$$y' = y + kx$$

NORMA DI UN VETTORE

La **norma** di un vettore è un numero **non negativo** che misura la grandezza (delle componenti) di un vettore.

La norma più utilizzata è la **norma euclidea (norma 2)** di un vettore, la quale rappresenta la lunghezza geometrica di un vettore.

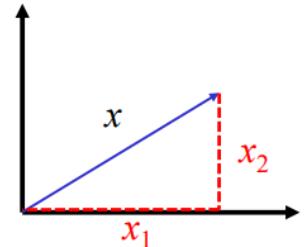
$$\|x\|_2 = \sqrt{x_1^2 + x_2^2 + \dots + x_n^2}$$

Da notare come questa successione possa essere vista come:

$$\|x\|_2 = \sqrt{x_1 x_1 + x_2 x_2 + \dots + x_n x_n} = \sqrt{x^T x}$$

e possiamo quindi **affermare che**:

$$\|x\|_2 = \sqrt{x^T x}$$



Versore: se divido un vettore per la sua lunghezza, ottengo un vettore che ha la stessa direzione del vettore ma con modulo 1.

$$\frac{x}{\|x\|_2}$$

ALTRE NORME IMPORTANTI

Norma di Manhattan (norma 1): è la somma dei valori assoluti delle componenti del vettore:

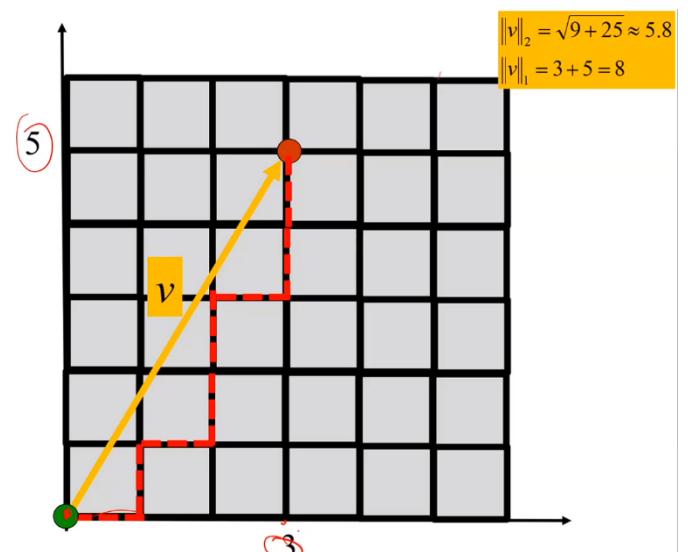
$$\|x\|_1 = |x_1| + |x_2| + \dots + |x_n|$$

Norma del massimo (norma ∞): è il massimo delle componenti del vettore in valore assoluto.

$$\|x\|_\infty = \max \{|x_1| + |x_2| + \dots + |x_n|\}$$

Nella stragrande maggioranza dei casi è sufficiente utilizzare la norma 2, ma non risulta sempre applicabile.

Ad esempio supponiamo di essere nella città di Manhattan e volessimo conoscere la lunghezza di un percorso. Non possiamo ovviamente tagliare in linea d'aria (norma 2) perché è come se passassimo attraverso i palazzi, invece dobbiamo usare la norma 1.



ANGOLO TRA DUE VETTORI

Il prodotto scalare tra due vettori è legato all'angolo Θ che si forma fra i due vettori, infatti:

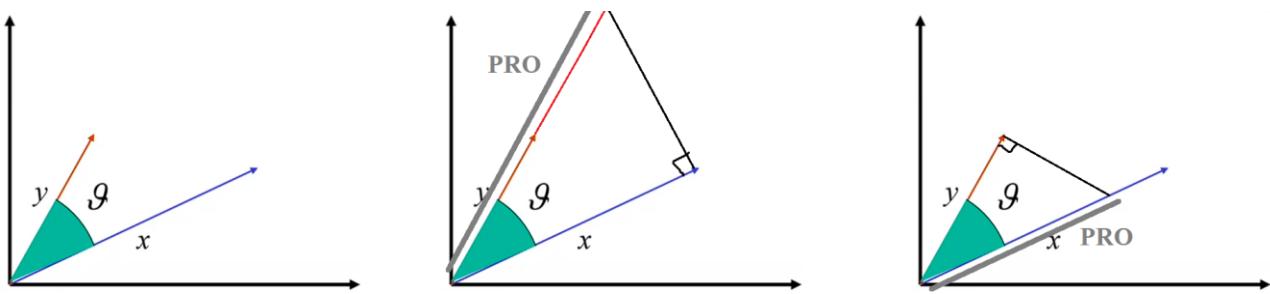
$$x^T y = \|x\|_2 \|y\|_2 \cos(\theta)$$

quindi possiamo ottenere l'angolo fra due vettori come (figura 1):

$$\cos(\theta) = \frac{x^T y}{\|x\|_2 \|y\|_2}$$

OSSERVAZIONI:

- se $x^T y = 0$, Θ è 90 gradi. Quindi i due vettori x e y sono *perpendicolari*. (condizione di perpendicolarità, molto importante).
- $\|x\|_2 \cos(\theta)$ = lunghezza della proiezione ortogonale di x su y . (figura 2)
- $\|y\|_2 \cos(\theta)$ = lunghezza della proiezione ortogonale di y su x . (figura 3)



NORMA DI UNA MATRICE

La norma di una matrice è un numero **non negativo** che misura la grandezza (delle componenti) di una matrice.

La norma di una matrice è definita allo stesso modo della norma di un vettore, ma dobbiamo fare attenzione ad alcune cose:

- la norma 1 di una matrice A consiste nella somma dei valori assoluti delle colonne di A .
- la norma infinito di una matrice A fa la stessa cosa della norma infinito di un vettore, ma lavora su righe.
Fa la somma in valore assoluto di tutte le componenti delle righe, e fra queste seleziona la più grande.

norma di Frobenius: è una norma che si ottiene come: $\|A\|_F = \sum_{ij} a_{ij}^2$

La **norma** di una matrice misura il potere amplificante/contraente di una matrice sul vettore che trasforma. Infatti se la norma di $A > 1$, quando A è applicato ad un vettore, otterremo un nuovo vettore che sarà più grande di quello di partenza (se $A < 1$ succede il contrario).

nb: $\|Ax\| \leq \|A\| \cdot \|x\|$

LEZIONE 10 – Risoluzione di sistemi lineari

Un **sistema di equazioni lineare** è un sistema composto da più equazioni lineari le quali devono tutte essere risolte contemporaneamente.

In un S.L. il numero di equazioni è uguale al numero di incognite.

Risolvere un sistema di eq. lineari significa risolvere:

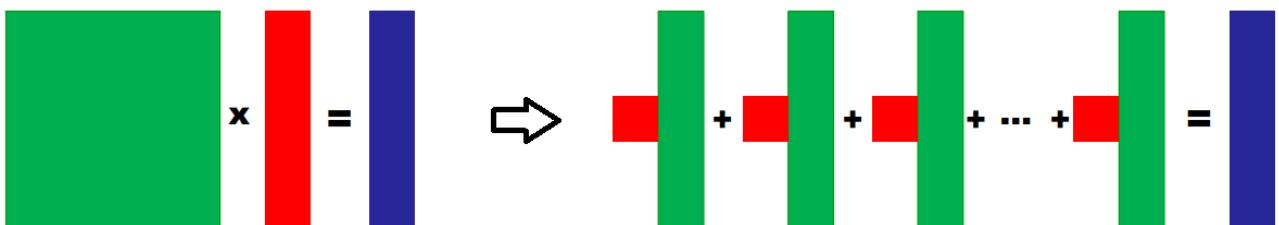
$$Ax = b$$

dove A = matrice dei coefficienti, x = vettore quantità incognite, b = vettore dei termini noti.

Il nostro scopo è quello di individuare x :

$$x: Ax = b \quad x: b - Ax = 0$$

il vettore $b - Ax$ prende il nome di vettore **residuo**. Quindi possiamo dire che “cerco un vettore x che azzera il residuo”.



Il problema ammette soluzione se:

- le colonne di A sono linearmente indipendenti, ovvero se **A è non singolare**;
- Se $\det(A) \neq 0$;
- Se il problema $Ax = 0$ è risolto solo al vettore nullo ($x=0$);
- se $\text{rank}(A) = n$, cioè le colonne sono linearmente indipendenti (ha rango massimo);

nb: sono tutte equivalenti. Se si verifica una condizione, si verificheranno anche le altre.

nb: una matrice **A è singolare** se ha tutti 0 sulla diagonale.

Risolvere $Ax = b$ tramite Inversa

Se una matrice è non singolare, allora tale matrice ammette anche **inversa**.

Questo significa che dato il problema $Ax = b$, posso moltiplicare ambo i membri per A^{-1} . Ricordo che per definizione $A^{-1} * A = \text{matrice identità}$, e la *matrice identità* * $x = x$. Ergo:

$$x = A^{-1}b$$

Questo potrebbe essere un metodo per risolvere il problema $Ax = b$, tuttavia gli algoritmi più efficienti non fanno uso di questa formula (es. algoritmo di Gauss).

Sistemi per cui il problema $Ax = b$ è facile da risolvere

Matrice diagonale:

$$\begin{aligned} a_{11}x_1 &= b_1 \\ a_{22}x_2 &= b_2 \\ a_{33}x_3 &= b_3 \\ a_{44}x_4 &= b_4 \end{aligned}$$

$$\begin{aligned} x_1 &= \frac{b_1}{a_{11}} \\ x_2 &= \frac{b_2}{a_{22}} \\ x_3 &= \frac{b_3}{a_{33}} \\ x_4 &= \frac{b_4}{a_{44}} \end{aligned}$$

$$\begin{pmatrix} a_{11} & & & \\ & a_{22} & & \\ & & a_{33} & \\ & & & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

Può essere risolta attraverso i seguenti **algoritmi**, dove A è una matrice, DA è il vettore degli elementi diagonali, oppure attraverso l'operazione componente per componente (nello specifico si parla di versione **vettorializzata**, cioè a parallelismo su dati).

La complessità è: n.

```
for i=1:n
    x(i)=b(i)/A(i,i)
end
```

```
for i=1:n
    x(i)=b(i)/DA(i)
end
```

x=b./DA

Matrice triangolare inferiore:

Si tratta di un sistema in cui le equazioni sono **debolmente accoppiate**, infatti ad ogni equazione ci basterà calcolare una sola incognita. Questo perché alla prima riga calcoliamo x_1 , alla seconda dovremo calcolare x_1 e x_2 , tuttavia conosciamo già x_1 e quindi dovremo calcolare solo x_2 . Lo stesso discorso possiamo applicarlo per tutte le righe successive.

Matrice triangolare superiore:

Si tratta di una situazione analoga alla matrice triangolare inferiore, con la differenza che dovremo procedere partendo dall'ultima operazione a risalire.

Quindi parto dall'ultima operazione calcolando l'ultima incognita, che andrò a sostituire nella penultima operazione così che debba calcolarmi solo la penultima incognita, e così via...

(IMPLEMENTAZIONE “NORMALE” DEI DUE METODI DI RISOLUZIONE →)
(c’è la versione **vettorializzata** sul file MATLAB).

$$\begin{pmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ a_{31} & a_{32} & a_{33} & \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

$$\begin{array}{lcl} a_{11}x_1 & & = b_1 \\ a_{21}x_1 + a_{22}x_2 & & = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 & & = b_3 \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 & = b_4 \end{array}$$

$$1) x_1 = \frac{b_1}{a_{11}}$$

$$2) x_2 = \frac{b_2 - a_{21}x_1}{a_{22}}$$

$$3) x_3 = \frac{b_3 - a_{31}x_1 - a_{32}x_2}{a_{33}}$$

$$4) x_4 = \frac{b_4 - a_{41}x_1 - a_{42}x_2 - a_{43}x_3}{a_{44}}$$

```

x(1)=b(1)/A(1,1)
for i=2:n
    x(i)=b(i)
    for j=1:i-1
        x(i)=x(i)-A(i,j)*x(j)
    end
    x(i)=x(i)/A(i,i)
end

```

$$T(n) = \frac{n(n+1)}{2} = O(n^2),$$

molt.,div.

$$\begin{array}{lcl} a_{11}x_1 & & = b_1 \\ a_{21}x_1 + a_{22}x_2 & & = b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 & & = b_3 \\ a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 & = b_4 \end{array}$$

Spiegazione dell'algoritmo

Il primo passo consisterà nel trattare la prima equazione, cioè risolvere “ $a_{1,1}x_1 = b_1$ ” . Per calcolare x_1 basterà moltiplicare ambo i membri per $1/a_{1,1}$, ottenendo $x_1 = b_1 / a_{1,1}$. Nel nostro caso specifico diremo che la *prima componente del vettore x* è uguale al rapporto fra la *prima componente del vettore b* e il *primo elemento della matrice A*.

Successivamente entreremo in un **ciclo for innestato** con il quale ci sposteremo lungo la matrice. Indicheremo con **i** l'indice di riga e con **j** l'indice di colonna.

Avendo già calcolato la prima riga al primo passo (e quindi anche la prima incognita), ci basterà solo calcolare l'incognita 2 della seconda riga. Conoscendo anche la seconda incognita potremo risolvere la terza equazione, e di conseguenza la quarta, e di conseguenza tutte le altre.

Il primo ciclo for andrà dalla seconda all'ultima riga (n), e ad ogni iterazione calcoleremo una nuova incognita. Porremo $x(i) = b(i)$ e andremo poi ad effettuare la sottrazione attraverso il **secondo ciclo for**.

Il secondo ciclo for andrà da 1 a $i-1$, e con questo calcoleremo il prodotto scalare fra il vettore riga $A(i,1:i-1)$ e il vettore colonna $x(1:i-1)$. Fondamentalmente stiamo moltiplicando ogni componente per la sua incognita e ne facciamo la somma.

Alla fine del primo ciclo, otterremo $x(i)$ come il rapporto fra $x(i)$ e l'elemento i -esimo della diagonale.

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{22} & a_{23} & a_{24} & \\ a_{33} & a_{34} & \\ a_{44} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{pmatrix}$$

A triangolare sup

$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = b_1$
$a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = b_2$
$a_{33}x_3 + a_{34}x_4 = b_3$
$a_{44}x_4 = b_4$

1) $x_4 = \frac{b_4}{a_{44}}$

2) $x_3 = \frac{b_3 - a_{34}x_4}{a_{33}}$

3) $x_2 = \frac{b_2 - a_{23}x_3 - a_{24}x_4}{a_{22}}$

4) $x_1 = \frac{b_1 - a_{12}x_2 - a_{13}x_3 - a_{14}x_4}{a_{11}}$

```

x(n)=b(n)/A(n,n)
for i=n-1:-1:1
    x(i)=b(i)
    for j=i+1:n
        x(i)=x(i)-A(i,j)*x(j)
    end
    x(i)=x(i)/A(i,i)
end

```

$$T(n) = \frac{n(n+1)}{2} = O(n^2),$$

molt.,div.

$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = b_1$
$a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = b_2$
$a_{33}x_3 + a_{34}x_4 = b_3$
$a_{44}x_4 = b_4$

Spiegazione dell'algoritmo

Il **primo passo** consisterà nel trattare l'ultima equazione, cioè risolvere “ $a_{n,n}x_n = b_n$ ”. Per calcolare x_n basterà moltiplicare ambo i membri per $1/a_{n,n}$, ottenendo $x_n = b_n / a_{n,n}$. Nel nostro caso specifico diremo che l'*ultima componente del vettore x è uguale al rapporto fra l'ultima componente del vettore b e l'ultimo elemento della matrice A*.

Successivamente entreremo in un **ciclo for innestato** con il quale ci sposteremo lungo la matrice. Indicheremo con **i** l'indice di riga e con **j** l'indice di colonna.

Avendo già calcolato l'ultima riga al primo passo (e quindi anche l'ultima incognita), ci basterà solo calcolare la penultima incognita della penultima riga. Conoscendo anche la penultima incognita possiamo calcolare la terzultima equazione, e di conseguenza anche la quartultima, e di conseguenza tutte le altre.

Il **primo ciclo for** andrà dalla penultima riga ($n-1$) alla prima riga (1), e ad ogni iterazione calcoleremo una nuova incognita. Porremo $x(i) = b(i)$ e andremo poi ad effettuare la sottrazione attraverso il **secondo ciclo for**.

Il **secondo ciclo for** andrà da $i+1$ fino ad n , e ad ogni iterazione sottrae da **b** la “parte a destra” della componente sulla diagonale, e possiamo farlo perché abbiamo già calcolato precedentemente quelle incognite.

Alla fine del primo ciclo, otterremo $x(i)$ come il rapporto fra $x(i)$ e l'elemento i -esimo della diagonale.

LEZIONE 11 – Risoluzione di sistemi lineari con Gauss

L'idea generale per risolvere un sistema lineare di **tipo generale** è quello di trasformarlo in un *sistema equivalente* (cioè con la stessa soluzione) *semplice da risolvere*.

È possibile fare ciò utilizzando il **metodo di eliminazione di Gauss**, con cui trasformiamo una matrice generale A in una matrice *triangolare superiore* equivalente.

$$Ax = b \rightarrow Ux = p$$

Ripetizione del metodo di eliminazione di Gauss per risolvere un sistema

Il primo elemento su ogni riga diverso da 0 è detto **pivot**.

La matrice generale viene trasformata in una matrice a scalini (trian. superiore). Per fare questo vengono utilizzate due mosse:

- se necessario, vengono scambiate due righe fra di loro;
- è possibile sostituire una riga con la somma di quella riga e un opportuno multiplo di un'altra riga. Cioè, un'altra riga moltiplicata per un'opportuna costante.

Lo **scopo** del metodo è quello di far compare uno 0 sotto ogni pivot.

Quadratic Equations System:

$$\begin{cases} x-y-z+w=0 \\ 2x+2z=8 \\ -y-2z=-8 \\ 3x-3y-2z+4w=7 \end{cases}$$

Augmented Matrix:

$$\left(\begin{array}{cccc|c} 1 & -1 & -1 & 1 & 0 \\ 2 & 0 & 2 & 0 & 8 \\ 0 & -1 & -2 & 0 & -8 \\ 3 & -3 & -2 & 4 & 7 \end{array} \right)$$

Labels:

- COLONA DEI TERMINI NOTI
- MATRICE COMPLETA ASSOCIATA AL SISTEMA LINEARE
- MATRICE DEI COEFFICIENTI

Row Operations:

$$\begin{aligned} R_2 &\rightarrow R_2 - 2R_1 \\ R_4 &\rightarrow R_4 - 3R_1 \\ R_3 &\rightarrow R_3 + \frac{1}{2}R_2 \\ R_3 &\leftrightarrow R_4 \end{aligned}$$

Simplified Matrix:

$$\left(\begin{array}{cccc|c} 1 & -1 & -1 & 1 & 0 \\ 0 & 2 & 4 & -2 & 8 \\ 0 & -1 & -2 & 0 & -8 \\ 0 & 0 & 1 & 1 & 7 \end{array} \right)$$

Solution:

$$\begin{aligned} -w &= -4 \implies w = 4 \\ z + w &= 7 \implies z = 7 - w = 7 - 4 = 3 \\ 2y + 4z - 2w &= 8 \implies y = \dots = 2 \\ x - y - z + w &= 0 \implies x = \dots = 1 \end{aligned}$$

Quando la matrice è stata ridotta a scala, ci sono 3 possibilità:

- se tutte le colonne della matrice dei coefficienti hanno un pivot, il sistema ha *un'unica soluzione*;
- se compare una riga del tipo 000...n con n ≠ 0, allora il sistema *non ammette soluzioni* (impossibile);
- se compare una riga del tipo 000...0, allora il sistema ammette *infinite soluzioni*.

Operazioni che garantiscono l'equivalenza

Come accennato prima, le operazioni che garantiscono l'**equivalenza** sono:

- moltiplicazione di una equazione per uno scalare (riga i-esima di **A**, componente i-esima di **b**);
- somma di una equazione con un'altra equazione (**nuova** riga di **A**, nuova componente i-esima di **b** = somma riga i-esima e riga k-esima di **A**, somma componente i-esima e k-sima di **b**).

L'algoritmo di Gauss combina queste due operazioni:

- somma/**sottrazione** di un multiplo di un'equazione da un'altra equazione (**nuova** riga di **A**, nuova componente i-esima di **b** = sottrazione di un multiplo della riga k-esima dalla riga i-esima di **A**, sottrazione di un multiplo della componente k-esima dalla componente i-esima di **b**).

ovvero:

$$\begin{aligned} A(i) &= A(i) - \text{multiplo}_A(k) \\ b(i) &= b(i) - \text{multiplo}_b(k) \end{aligned}$$

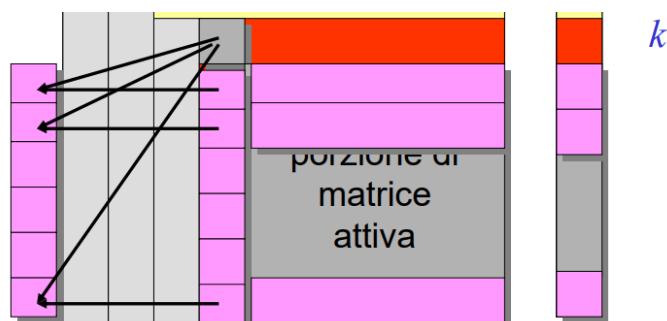
Si procede per passi: al passo k , l'equazione k-sima è detta **equazione pivotale** e l'elemento A_{kk} è detto **pivot**. Lo scopo è di rendere nullo in ognuna delle nuove equazioni il coefficiente della k-sima incognita, scegliendo opportunamente il multiplo dell'equazione pivotale. Tale multiplo è detto **moltiplicatore**.

Per **ogni** equazione da sostituire, si deve calcolare il relativo moltiplicatore. Si noti che il moltiplicatore è moltiplicato per l'equazione pivotale e non per l'equazione da sostituire. Il moltiplicatore relativo alla i-sima equazione da sostituire è il rapporto A_{ik}/A_{kk} .

Moltiplicando tale moltiplicatore per la riga pivotale, si ottiene un multiplo dell'equazione pivotale che sottratto dall'i-sima equazione fornisce una nuova equazione, in cui la k-sima incognita ha coefficiente 0. L'effetto totale del k-simo passo dell'AG è ottenere un sistema equivalente in cui gli elementi della matrice dei coefficienti che si trovano in colonna k e sotto la diagonale sono nulli.

Quindi, ad ogni passo modifico la riga i-esima e la matrice sottostante. La colonna i-esima, da sotto il pivot (di posizione (k,k)), avrà solo 0.

Da notare che viene modificata anche la colonna dei termini noti. La parte di matrice di matrice che viene modificata al passo k prende il nome di **porzione di matrice attiva**.



Operazioni fondamentali

```
1. for k=1:n-1
2.   for i=k+1:n
3.     molt= A(i,k)/A(k,k);
4.     for j=k+1:n
5.       A(i,j)=A(i,j)-A(k,j)*molt;
6.     end
7.     b(i)=b(i)-molt*b(k);
8.   end
9. end
```

Per ogni riga k della matrice, per k che vada da 1 a $n-1$, ci occuperemo di modificare le righe sottostanti per trasformare la matrice in una matrice triangolare superiore [1-9].

La riga al passo k diventa l'**equazione pivotale**, e per tutte le righe che vanno da $k+1$ a n effettueremo una serie di operazioni [2-8]:

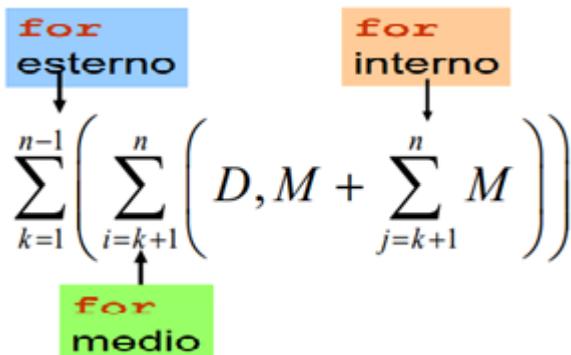
- calcolare il **moltiplicatore** per la i -sima riga. [3]
Esso sarà calcolato come il **rapporto** fra il *coefficiente dell'incognita da eliminare* dell' i -sima riga **A(i,k)** e il **pivot A(k,k)**.
- **sottrarre** dall' i -sima riga la riga pivotale (riga k -esima) moltiplicata per il suo moltiplicatore [4-5].
NB: $A(i,j)=A(i,j)-A(k,j)*molt$; effettua l'operazione solo sull'elemento di posizione $A(i,j)$. Noi ripeteremo il processo per ogni colonna della riga (da $k+1$ a n).
- **sottrarre** da $b(i)$ il moltiplicatore moltiplicato per $b(k)$ [7].

L'algoritmo agisce **in place**, quindi perdo le vecchie componenti della matrice A e del vettore colonna b .

Inoltre, lavora solo sulla parte attiva, quindi non azzerà in maniera esplicita il triangolo strettamente inferiore (tuttavia non ci interessa perché risulta essere un'azione non necessaria).

Complessità

L'algoritmo contiene 3 cicli for uno dentro l'altro, dove nel *for medio* è presente una moltiplicazione ed una divisione, e nel *for interno* è presente solo una moltiplicazione. Se sostituiamo i *for* con delle sommatorie otterremo che:



Partendo dal *for interno* noto che non c'è nulla che dipende da j (c'è solo M), quindi è come se facessi $\sum_{j=k+1}^n 1$, il cui risultato è $n-k$. Avrò quindi:

$$\sum_{k=1}^{n-1} \left(\sum_{i=k+1}^n (D, M + (n-k)M) \right) = \sum_{k=1}^{n-1} ((n-k)D, M + (n-k)^2 M)$$

Infine, noterò che quando $(n-k) = (n - (n-1))$ quando arriviamo al passo $n-1$. Quindi possiamo riscrivere $(n-k)$ come k .

$$\sum_{k=1}^{n-1} k D, M + \sum_{k=1}^{n-1} k^2 M$$

Per semplicità non facciamo più distinzione fra moltiplicazioni e divisioni (consideriamo $D=M$). Quindi alla prima sommatoria avremo $2M$, dove porteremo il 2 fuori.

$$2 \sum_{k=1}^{n-1} k M + \sum_{k=1}^{n-1} k^2 M$$

La **prima sommatoria** corrisponde alla *formula di Gauss bambino*, mentre la **seconda sommatoria** corrisponde alla *somma dei quadrati dei primi n numeri*. Avrò quindi:

$$\begin{aligned} & \left[(n-1)n + \frac{(n-1)n(2n-1)}{6} \right] M \\ & O(n^2) + O\left(\frac{n^3}{3}\right) M = O\left(\frac{n^3}{3}\right) M \end{aligned}$$

LIMITI DELL'ALGORITMO DI GAUSS

L'algoritmo di Gauss, nella sua versione standard, non è completamente affidabile. Infatti:

- se il **pivot = 0** l'algoritmo non è applicabile. Questo perché si verificherebbe una divisione per 0.
- se si presentano **pivot piccoli** rispetto agli altri elementi della matrice l'algoritmo è *instabile*, questo perché dei pivot piccoli danno origine a moltiplicatori grandi, i quali amplificano gli errori che esistono sui valori degli elementi della matrice.
(si parla di *eccessiva sensibilità alla precisione finita*, e rende l'algoritmo *inapplicabile*).

$$A = \begin{pmatrix} 0 & 2 & 1 \\ 0 & 0 & 4 \\ 1 & 3 & 22 \end{pmatrix} \quad A = \begin{pmatrix} 0.000001 & 2 & 1 \\ 0 & 0.0000001 & 4 \\ 1 & 3 & 22 \end{pmatrix}$$

Per rendere **stabile** l'algoritmo possiamo scambiare l'ordine delle righe, in modo che venga scelto sempre (ad ogni passo k) il pivot più opportuno. La scelta adottata in questo caso è quella di selezionare l'elemento più grande sulla colonna del pivot e di scambiare la riga di quell'elemento con la riga corrente.

Tale tecnica prende il nome di strategia del **pivoting parziale**.

Da un punto di vista **implementativo**, basta aggiungere all'algoritmo della triangolarizzazione di Gauss un'azione preliminare, che consiste in uno scambio opportuno delle righe.

nb: se una matrice contiene tutti 0 dal pivot in giù, si tratta sicuramente di una matrice singolare (quindi se il pivot_max = 0, mi fermo).

Per quanto riguarda la **complessità di tempo**, nonostante si aggiungano operazioni di ricerca del massimo e di scambio, non si aggiungono operazioni aritmetiche. Quindi la complessità rimane $O(n^3/3)$.

Problema ben e mal condizionato

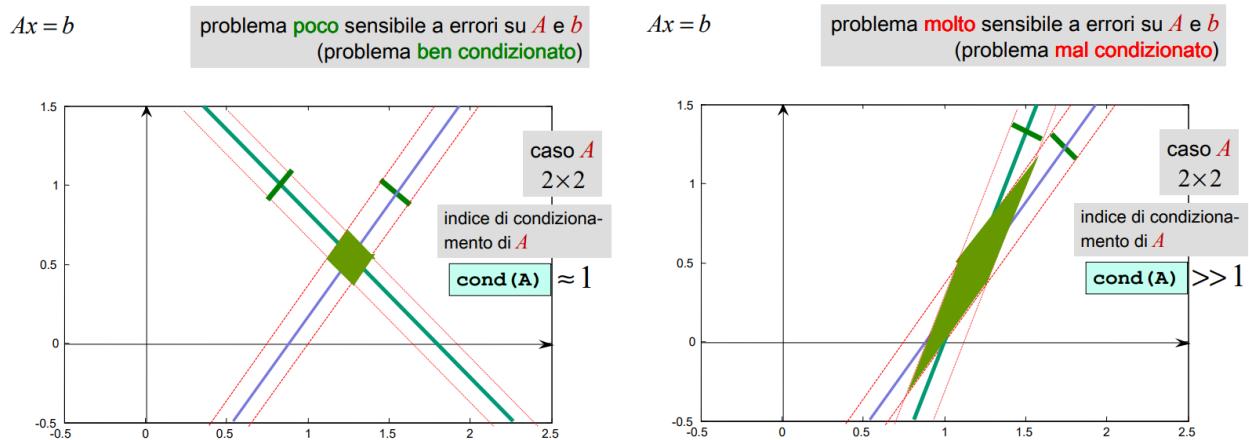
Supponiamo che A sia una matrice 2×2 (sistema di due eq. in due incognite). Da un punto di vista geometrico avremo due rette, e la soluzione del sistema sarà data dal loro punto di intersezione.

Ognuna delle due rette avrà un certo *intervallo di incertezza*. L'**incertezza sulla soluzione** sarà dato dalla regione che si viene a creare dall'intervallo di incertezza delle rette.

In MATLAB è possibile calcolare l'incertezza sulla soluzione attraverso il comando `cond(A)`.

Detto ciò, diremo che:

- un problema $Ax=b$ è **ben condizionato** se le rette sono (o quasi) perpendicolari, o se $\text{cond}(A) \approx 1$.
- un problema $Ax=b$ è **mal condizionato** se le rette sono quasi parallele, o se $\text{cond}(A)$ ci restituisce un numero molto maggiore di 1.



OSSERVAZIONI:

- l'indice di condizionamento di un problema $Ax=b$ NON dipende da b , ma solo dalla matrice A ;
- se le rette sono parallele, il problema non ammette soluzione;
- se le rette sono parallele e coincidenti, il problema ammette infinite soluzioni.

Proprietà dell'algoritmo di Gauss (con pivoting):

- è il più efficiente tra i metodi diretti per $\mathbf{Ax} = \mathbf{b}$;
- la soluzione calcolata dall'algoritmo (eseguito in precisione finita) garantisce un **residuo piccolo**;
- la soluzione calcolata dall'algoritmo (eseguito in precisione finita) garantisce un **errore piccolo** solo se il **problema è ben condizionato**;
- se l'**indice di condizionamento** della matrice \mathbf{A} è dell'ordine di 10^q allora l'**errore** non può essere minore di 10^{-p+q} , dove 10^{-q} è l'accuratezza delle componenti di \mathbf{A} e di \mathbf{b} .

Se \mathbf{A} e \mathbf{b} hanno accuratezza massima ($p=16$), allora q non può essere > 16 . (in tal caso, dovremmo cambiare calcolatore con uno con un'accuratezza maggiore).

- se l'**indice di condizionamento** della matrice \mathbf{A} è dell'ordine di 10^q allora la soluzione calcolata ha un numero di cifre corrette (rispetto alla soluzione) non maggiore di $p-q$, dove p è il numero di cifre corrette delle componenti di \mathbf{A} e di \mathbf{b} . (si **persono q cifre**).

Sostanzialmente, più q è grande, più cifre perdiamo.

Le osservazioni sull'indice di condizionamento dovrebbe essere fatte come azioni preliminare per assicurarci quante cifre delle soluzioni siano affidabili.

residuo: $r = \mathbf{b} - \mathbf{Ax}_{\text{Gauss}}$

errore: $E = \mathbf{x} - \mathbf{x}_{\text{Gauss}}$

LEZIONE 12 – Fattorizzazione LU

Ogni matrice A , quadrata di ordine n e non-singolare, può essere ottenuta come **prodotto** tra una matrice triangolare superiore (indicata con U) e una matrice triangolare inferiore con diagonale unitaria, ovvero ha tutti 1 sulla diagonale (indicata con L), ovvero $A = LU$.

Si tratta di una **proprietà** propria delle matrici, e non solo dei sistemi di equazioni. Infatti, attraverso il calcolo della fattorizzazione $A = LU$ possiamo calcolare altre informazioni relative ad A (determinante, matrice inversa).

Per **risolvere il sistema** mi basterà riscrivere $Ax = b$ come $LUX = b$.

UX è il prodotto fra una matrice ed un vettore, che mi restituirà un altro vettore p .

Quindi il problema $LUX = b$ diventa il sistema:

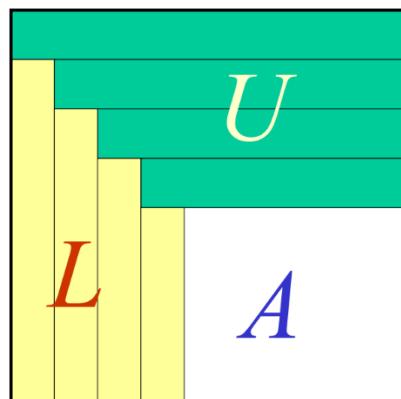
- $Lp = b$ (si calcola prima p attraverso l'algoritmo di sostituzione in avanti)
- $UX = p$ (poi si calcola x attraverso l'algoritmo di sostituzione all'indietro)

Una volta calcolata p , ovvero quando p diventa **nota**, nella seconda equazione rimane solo x come incognita.

La matrici L e U sono ottenute attraverso un algoritmo che è semplicemente una **variazione dell'algoritmo di Gauss**.

Quando trasformavamo la matrice A in una matrice a scalini, inserivamo tutti 0 nel triangolo inferiore. In questo caso invece andremo lì a **memorizzare i moltiplicatori**, sicché quando arrivo alla fine dell'algoritmo non avrò più la matrice A (se è *in-place*) ma la matrice **U nel triangolo superiore** e la matrice **L nel triangolo strettamente inferiore**. Da notare che NON memorizzo la diagonale di L in quanto già so essere una diagonale formata da tutti 1.

Attraverso questo algoritmo *in-place* memorizzo solo le quantità fondamentali da memorizzare e non quelle inutili. Inoltre è da notare che le matrice L ed U le otteniamo solo ALLA FINE dell'algoritmo.



La **complessità di tempo** dell'algoritmo di fattorizzazione LU è $O(n^3/3)$.

Tecnica del Pivoting per la Fattorizzazione LU

Anche l'algoritmo della fattorizzazione, nella sua versione standard, non è completamente affidabile, per gli stessi motivi che abbiamo già visto con l'algoritmo di Gauss.

Per rendere **stabile** l'algoritmo dovremo dunque utilizzare la strategia del **pivoting parziale** (scambio opportuno di righe), ma con una leggera differenza: infatti non avremo che $A=LU$ ma che $PA=LU$, perché dobbiamo tenere conto degli scambi che abbiamo fatto.

P è detta **matrice di permutazione**, e “ricorda” gli scambi di righe del pivoting.

Fondamentalmente, si tratta di aggiungere solo un passaggio in più. Da notare che se effettuo uno scambio sulla matrice dei coefficienti dovrò fare lo stesso scambio anche sulla colonna dei termini noti, per questo motivo dovrò moltiplicare sia A che b per P .

Avrò dunque che:

1. $Ax=b$
2. $(PA)x=Pb$
3. $(LU)x=Pb$ //siccome $PA=LU$

E quindi otterremo il seguente sistema (**nb**: per evitare ambiguità diremo che $Ux=y$ invece che p):

- $Ly = Pb$
- $Ux = y$

Possiamo **ottenere la matrice di Permutazione** dalla Matrice Identità, sulla quale effettueremo delle operazioni:

- PA : scambia le righe di A ;
- AP : scambia le colonne di A .

Supponiamo di avere una matrice A 4×4 , e di voler fare due operazioni: scambiare le righe 1 e 4 e scambiare le colonne 1 e 4. Quindi dovremo scambiare le righe 1 e 4 della Matrice Identità e fare PA per scambiare le righe, e poi AP per scambiare le colonne.

$$I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad \xrightarrow{\hspace{1cm}} \quad P = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

PA scambia le righe 1 e 4 di A

AP scambia le colonne 1 e 4 di A

Da notare tuttavia che da un punto di vista pratico è **inutile** conservare la matrice di Permutazione (si tratta di un accorgimento teorico), in quanto si utilizza la **tecnica di indirizzamento attraverso indici puntatori**.

In MATLAB si utilizza un vettore **ipiv** che conterrà degli indici, che vanno da 1 al numero di righe/colonne della matrice. Ogni qualvolta effettuerò uno scambio andrò a memorizzare lo stesso scambio nel vettore **ipiv**.

Esempio (scambio prima e ultima riga):

```
A([k r], :) = A([r k], :);
ipiv([k r]) = ipiv([r k]);
```

$$A = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ l & m & n & o \\ q & r & s & t \end{pmatrix} \quad \xrightarrow{\hspace{1cm}} \quad A = \begin{pmatrix} q & r & s & t \\ e & f & g & h \\ l & m & n & o \\ a & b & c & d \end{pmatrix}$$

$$\text{ipiv} = \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \quad \xrightarrow{\hspace{1cm}} \quad \text{ipiv} = \begin{pmatrix} 4 \\ 2 \\ 3 \\ 1 \end{pmatrix}$$

PROPRIETÀ DELLE MATRICI

Queste sono proprietà che valgono in generale:

- calcolo del determinante;
- risoluzione dei sistemi multipli;
- calcolo della matrice inversa.

CALCOLO DEL DETERMINANTE

Il **determinante di un prodotto** è il prodotto dei determinanti. Quindi:

$$\det(A) = \det(LU) \quad \text{dove } \det(LU) = \det(L)\det(U).$$

Il **determinante di una matrice triangolare** è il prodotto degli elementi diagonali. Quindi:

$$\det(L)=1 \text{ (perché abbiamo tutti 1 sulla diagonale)}, \quad \det(U)=u_{11} \cdot u_{22} \cdot \dots \cdot u_{nn}$$

$$\det(A) = \det(LU) = u_{11} \cdot u_{22} \cdot \dots \cdot u_{nn}$$

Tale calcolo è $T(n)=O(n^3/3)$

RISOLUZIONE DEI SISTEMI MULTIPLI

La risoluzione di sistemi multipli riguarda la risoluzione di più sistemi con la stessa matrice dei coefficienti e differenti vettori di termini noti.

$$AX = B \quad , \quad A \in \mathbb{R}^{n \times n}; \quad X, B \in \mathbb{R}^{n \times s}$$



$$Ax_{\bullet j} = b_{\bullet j} \quad , \quad j = 1 : s$$

X e B sono matrici che hanno lo stesso numero di righe di A ed hanno S colonne, e dove la prima colonna di X è la soluzione del sistema

$A^*(\text{prima colonna di } X) = (\text{prima colonna di } b)$, e così via...

La complessità è $T(n)=O(n^3/3) + O(s \cdot n^2)$, che per n molto grande sarà: $T(n)=O(n^3/3)$

CALCOLO DELLA MATRICE INVERSA

$$AA^{-1} = I \quad , \quad A, A^{-1}, I \in \mathbb{R}^{n \times n}$$

Le colonne della matrice inversa si determinano risolvendo n sistemi con matrice dei coefficienti A e vettori di termini noti e_j . La complessità è $T(n)=O(n^3/3)$.

LEZIONE 13 – Fitting di dati: interpolazione di Lagrange

Il **fitting di dati** è il problema della ricostruzione di un **fenomeno continuo**, espresso da una funzione incognita, a partire da **m dati** (punti del piano) che **costituiscono** un **campionamento discreto** di tale fenomeno.

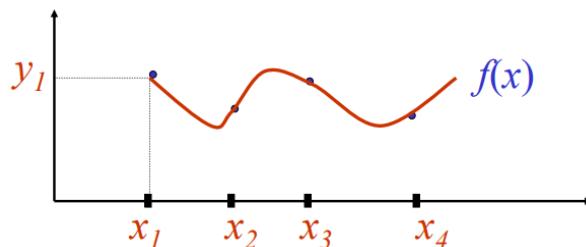
In base al tipo di dati possiamo effettuare due tipi di ricostruzioni:

- **ricostruzione esatta**, detta **interpolazione**, se i dati sono esatti e **si vuole ricostruire il fenomeno in modo esatto**;
- **ricostruzione approssimata**, detta **approssimazione**, se i dati sono affetti da errore e **si vuole ricostruire un trend**.

I **dati di input** di un problema di interpolazione sono le **m coppie di numeri** (x_i, y_i) , che rappresentano l'**ascissa (nodo)** e l'**ordinata (valore) dell'i-simo dato** (i-simo punto p_i del piano). Inoltre, si deve preventivamente **fissare** un insieme (infinito) **F** di funzioni (**modello**), in cui si ricerca, se esiste, una **funzione $f(x)$** che ricostruisce il fenomeno (in **F**) e che è l'output, cioè la **soluzione del problema di interpolazione**.

Se un certo punto (x_i, y_i) appartiene al grafico, allora $y_i = f(x_i)$. Quindi lo **scopo** sarà trovare (se esiste) **una f in F tale che: $f(x_i) = y_i, i=1:m$** ;

Queste **m** equazioni sono dette **condizioni di interpolazione di Lagrange**.



L'insieme **F (modello lineare)** è costituito da funzioni che sono univocamente determinate da **n coefficienti**.

Quello che **vogliamo fare** è costruire uno spazio di funzioni modello che sia una **combinazione delle funzioni prefissate b (di base)** con dei **coefficienti a** .

Queste **funzioni prefissate** devono essere **già conosciute**, e possiamo definirle noi stessi individualmente oppure possiamo definirle in maniera più **generica**, ad esempio può risultare conveniente considerare: $b_i(x) = x^{i-1}$, ovvero **F** è l'insieme dei polinomi algebrici di grado al più **$n-1$** .

La funzione $f(x)$ sarà data dal polinomio:
$$f(x) = a_1 b_1(x) + a_2 b_2(x) + \dots + a_n b_n(x)$$

funzioni prefissate (di base)

Quindi per trovare la $f(x)$, il nostro **obiettivo** sarà quello di calcolare gli n coefficienti a_1, a_2, \dots, a_n .

Quello che andrò a fare sarà **imporre le m condizioni di interpolazione**, cioè cerco di mescolare la forma dell'interpolante con i vincoli (ovvero le informazioni che devo rispettare).

Mi ritroverò con un **sistema di m equazioni**, del tipo:

$$f(x_1) = y_1$$

$$f(x_2) = y_2$$

... ...

$$f(x_m) = y_m$$

Tuttavia io so che $f(x_i) = a_1 b_1(x_i) + a_2 b_2(x_i) + \dots + a_n b_n(x_i) = y_i$

Quindi posso **riscrivere il sistema** come:

$$a_1 b_1(x_1) + a_2 b_2(x_1) + \dots + a_n b_n(x_1) = y_1$$

$$a_1 b_1(x_2) + a_2 b_2(x_2) + \dots + a_n b_n(x_2) = y_2$$

.....

$$a_1 b_1(x_m) + a_2 b_2(x_m) + \dots + a_n b_n(x_m) = y_m$$

Si tratta di un **sistema lineare** con m equazioni e n incognite.

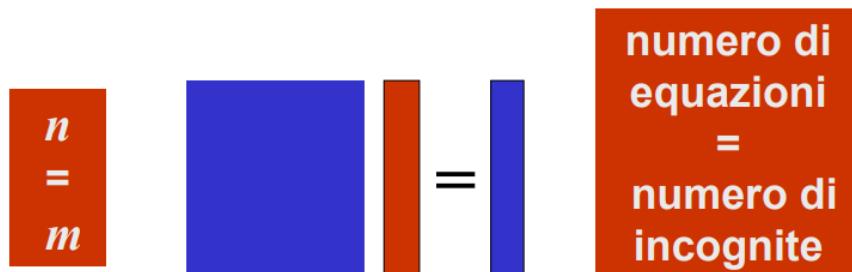
Da notare che le colonne $b(x)$ e y sono note mentre le colonne a sono **incognite**.

Se consideriamo B come la **matrice dei coefficienti**, y il **vettore dei termini noti** ed a il **vettore incognita**, ci troveremmo di fronte al **problema della risoluzione di un sistema lineare: $Ba = y$** .

$$B = \begin{pmatrix} b_1(x_1) & b_2(x_1) & \cdots & b_n(x_1) \\ b_1(x_2) & b_2(x_2) & \cdots & b_n(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ b_1(x_m) & b_2(x_m) & \cdots & b_n(x_m) \end{pmatrix} \quad y = (y_1, y_2, \dots, y_m)^T$$

$a = (a_1, a_2, \dots, a_n)^T$ le incognite

Da notare che se il sistema è rettangolare potrebbe non avere un'unica soluzione.
Per far sì che sia unica il numero di condizioni $m = \text{numero di coefficienti } n$.



Se il sistema ammette un'unica soluzione quindi dipende da **B**, tuttavia **B** non dipende né dai coefficienti a né dai valori, ma **dipende esclusivamente dai nodi**.
Cambiando i nodi e cambiando le funzioni di base potremmo avere matrici B che hanno soluzione unica o più soluzioni.

Tuttavia esistono delle **funzioni base**, definite come $b_i(x) = x^{i-1}$, le quali **affermano** che **esiste un unico polinomio di grado al più $n-1$ che interpola su n punti**.

Infatti sappiamo perfettamente che su *due punti passa una retta (pol. 1° grado)*, su *tre punti passa una parabola (pol. 2° grado)*, e così via...

$$F = \langle 1, x, x^2, x^3, \dots, x^{n-1} \rangle = \Pi_{n-1} \quad (\text{spazio dei polinomi algebrici})$$

nb: con $\langle \rangle$ si indica lo spazio generato

La **matrice dei coefficienti B** che andremo a generare prenderà il nome di **matrice di Van der Monde**, in quanto ha la caratteristica particolare che **per ogni colonna i , i nodi sono elevati alla i** .

$$B = \begin{pmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ 1 & x_2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{n-1} \end{pmatrix}$$

matrice di
**Van der
Monde**

Secondo un determinato teorema, sappiamo che tale matrice è **sempre non singolare** fintanto che i nodi siano tutti diversi fra di loro, ovvero **se $x_i \neq x_j$** .

Ciò significa che non devono esserci due coppie di nodi uguali.

OSSERVAZIONE: Diciamo che la soluzione può essere di grado al più $n-1$ e non $n-1$ perché la soluzione può anche essere 0.

Pensiamo che la soluzione sia data dal polinomio $f(x) = a_1 + a_2x$. Si tratta quindi di una retta del tipo $f(x) = mx+q$, che ha grado 1.

Tuttavia se il coefficiente angolare q fosse 0 ci ritroveremmo con una costante, ovvero $a_2 = 0$. Quindi la soluzione potrebbe anche essere data solo da $f(x) = a_1$, che è un polinomio di grado 0.

In modo simile pensiamo che la soluzione sia data dal polinomio $f(x) = a_1 + a_2x + a_3x^2$. Si tratta di una parabola, che ha grado 2. Tuttavia se tutti e 3 i punti della parabola fossero allineati ci ritroveremmo con una retta, che ha grado 1.

OSSERVAZIONE: Talvolta possiamo trovare anche la notazione: $B_{ij} = b_j(x_i)$
Ovvero che la cella (i,j) della matrice B è data dalla *funzione base j-sima* calcolata nel *nodo i-esimo*.

ALTRI TIPI DI FUNZIONI BASE

Ovviamente non siamo costretti ad utilizzare funzioni potenza (caso di prima), ma **potremmo voler utilizzare** tipi diversi come i **polinomi trigonometrici**.

Il motivo per cui vorremmo fare un particolare modello di funzioni è perché $f(x)$ eredita le **qualità di questi polinomi**. Ad esempio, se tutte le funzioni sono pari allora anche $f(x)$ sarà pari, se le funzioni sono trigonometriche allora $f(x)$ sarà una funzione *oscillante e periodica*.

esempio:

interpolate con un **polinomio trigonometrico**
(pari) di **secondo grado** su **tre punti**

$$f(x) = a_1 + a_2 \cos(x) + a_3 \cos(2x)$$

$$\begin{pmatrix} 1 & \cos(x_1) & \cos(2x_1) \\ 1 & \cos(x_2) & \cos(2x_2) \\ 1 & \cos(x_3) & \cos(2x_3) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

Un altro modello lineare può essere il seguente, ove la funzione $f(x)$ assumerà le caratteristiche delle funzioni base. e^x è concava mentre \sqrt{x} è convessa (quindi dovremmo ottenere un effetto intermedio) con $x \geq 0$.

$$f(x) = a_1 + a_2 e^x + a_3 \sqrt{x}$$

$$\begin{pmatrix} 1 & e^{x_1} & \sqrt{x_1} \\ 1 & e^{x_2} & \sqrt{x_2} \\ 1 & e^{x_3} & \sqrt{x_3} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

LEZIONE 14 – Fitting di dati: interpolazione di Hermite & Qualità ricostruzione

Il problema dell'interpolazione di Lagrange può essere esteso imponendo ulteriori condizioni e ottenendo così problemi di interpolazione più avanzati, come quello di Hermite.

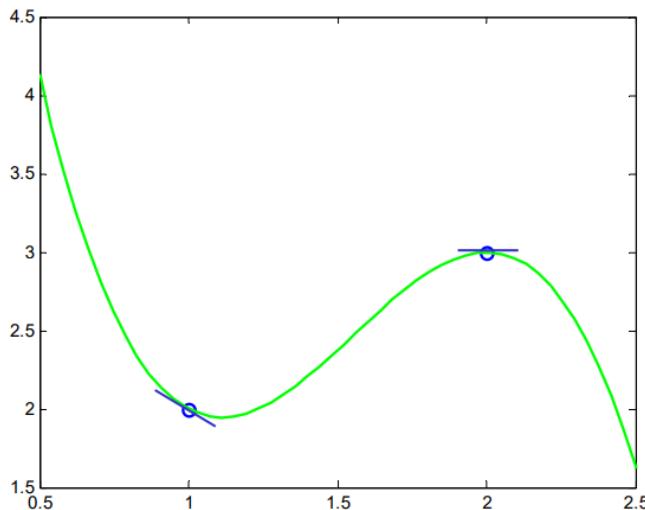
L'**interpolazione di Hermite** è un tipo di interpolazione con la quale non solo richiediamo che la funzione passi per determinati punti, ma anche che la **pendenza** in questi punti sia assegnata.

I **dati di input** di un problema di interpolazione sono le **m coppie di numeri** (x_i, y_i) , che rappresentano l'**ascissa (nodo)** e l'**ordinata (valore)** dell'i-simo dato (i-simo punto p_i del piano) e **m numeri** d_i che rappresentano le pendenze. Inoltre, si deve preventivamente fissare un insieme (infinito) **F** di funzioni (**modello**), in cui si ricerca, se esiste, una **funzione $f(x)$** che ricostruisce il **fenomeno** (in **F**) e che è l'**output**, cioè la soluzione del problema di interpolazione.

Se un certo punto (x_i, y_i) appartiene al grafico, allora $y_i = f(x_i)$. Quindi lo **scopo** sarà trovare (se esiste) una **f** in **F** tale che: $f(x_i) = y_i, \quad i=1:m$
 $f'(x_i) = d_i, \quad i=1:m$

Queste $2m$ equazioni sono dette **condizioni di interpolazione di Hermite**.

Quindi oltre a dover soddisfare le **condizioni di interpolazione di Lagrange**, vorremmo che la **pendenza** (cioè il coeff. angolare della retta tangente, cioè la *derivata della funzione*) in questi punti x_i sia assegnata. Quindi per ogni punto conosciamo sia il valore della funzione che la pendenza.



Se abbiamo $2m$ equazioni, possiamo presumere che **F** deve essere un **insieme modello** caratterizzato da **2m coefficienti** (per meglio dire, da $2m$ funzioni note $b_j(x)$, $j=1:2m$). Infatti, in tal caso si ottiene un sistema di equazioni lineari in cui il numero delle equazioni è uguale al numero delle incognite; infine, se la matrice di tale sistema è **non-singolare**, allora il **problema di interpolazione di Hermite ammette soluzione unica** (in **F**).

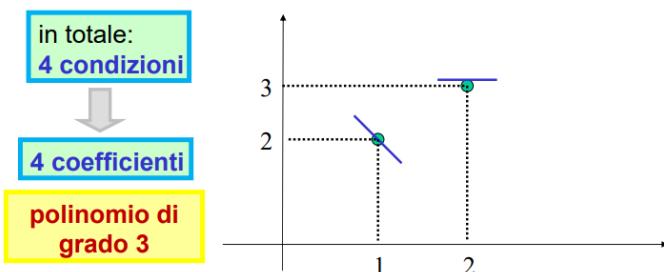
Da questa analisi discerne il **Teorema dell'interpolazione di Hermite con polinomi**, il quale afferma che: *dati m punti distinti del piano ed m pendenze, esiste un unico polinomio di grado al più $2m-1$ che interpola tali punti assumendo in essi le pendenze assegnate.*

Esempio di interpolazione di Hermite dove abbiamo 2 punti e 2 pendenze, quindi ricaviamo un polinomio di grado 3.
 (nb: p_1 e p_2 sono due punti, p_3 intende “polinomio di grado 3”)

Esercizio: risolvere il problema di interpolazione di Hermite con polinomi
 punti di interpolazione $p_1 = (1, 2)$
 $p_2 = (2, 3)$

2 condizioni di pendenza:

- ✓ la **pendenza** nel primo nodo del polinomio interpolante è fissata ($d_1 = -1$)
- ✓ la **pendenza** nel secondo nodo del polinomio interpolante è fissata ($d_2 = 0$)



$$\begin{cases} p_3(x_1) = y_1 & x_1 = 1, x_2 = 2 \\ p_3(x_2) = y_2 & y_1 = 2, y_2 = 3 \\ p'_3(x_1) = -1 & \end{cases} \quad \Rightarrow \quad \begin{cases} a_3 x_1^3 + a_2 x_1^2 + a_1 x_1 + a_0 = y_1 & x_1 = 1, x_2 = 2 \\ a_3 x_2^3 + a_2 x_2^2 + a_1 x_2 + a_0 = y_2 & y_1 = 2, y_2 = 3 \\ 3a_3 x_1^2 + 2a_2 x_1 + a_1 = -1 & \\ 3a_3 x_2^2 + 2a_2 x_2 + a_1 = 0 & \end{cases}$$

- il polinomio **interpola** nei 2 nodi
- la **pendenza** nel primo nodo del polinomio interpolante è fissata ($d_1 = -1$)
- la **pendenza** nel secondo nodo del polinomio interpolante è fissata ($d_2 = 0$)

$$\begin{pmatrix} x_1^3 & x_1^2 & x_1 & 1 \\ x_2^3 & x_2^2 & x_2 & 1 \\ 3x_1^2 & 2x_1 & 1 & 0 \\ 3x_2^2 & 2x_2 & 1 & 0 \end{pmatrix} \begin{pmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ -1 \\ 0 \end{pmatrix}$$

nb: il polinomio risultato è quello della pagina precedente.
 $d=-1$ sarebbe la bisettrice che passa per 2 e 4 quadrante. $d=0$ sarebbe la retta costante

QUALITÀ DELLA RICOSTRUZIONE

Supponiamo di avere due funzioni. La **funzione $f(x)$ esatta**, la quale sarà costituita da un insieme di punti (x_i, y_i) dove $y_i = f(x_i)$, e una **funzione interpolante di $f(x)$** , la quale sarà costituita da un insieme di punti (x_i, y_i) dove $y_i = p(x_i)$.

Con **qualità della ricostruzione** ci domandiamo quanto sia *simile* la funzione interpolante alla funzione esatta.

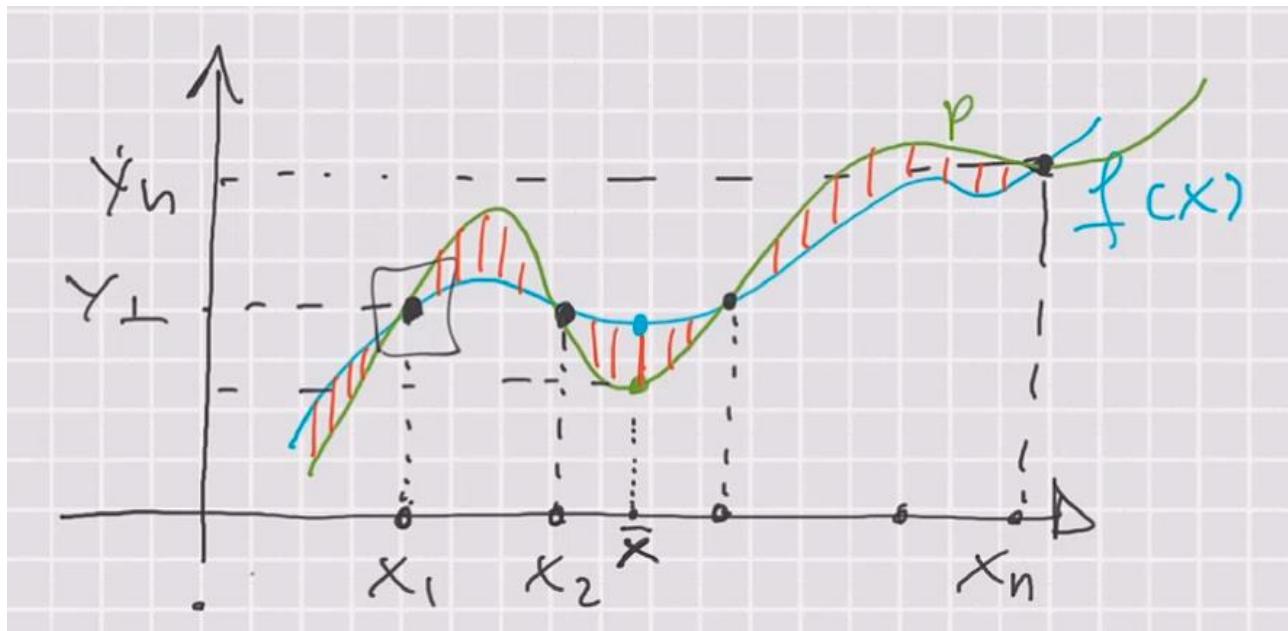
L'interpolante e la funzione si somiglieranno in tutte le zone intorno ai punti (x_i, y_i) , in quanto entrambe passano per lo stesso punto. Tuttavia più ci allontaneremo da questi punti e più è probabile che le due funzioni differiscano.

La differenza in un punto \bar{x} fra queste due funzioni è dato dalla **differenza delle due funzioni calcolate nel punto**.

Tale differenza prende il nome di **Errore**: $E(\bar{x}) = |p(\bar{x}) - f(\bar{x})|$

Nei punti (x_i, y_i) l'errore sarà uguale a 0, in quanto $p(x_i) = f(x_i)$.

Infatti $p(x_i) = f(x_i) \rightarrow p(x_i) - f(x_i) = 0$.



Per sapere se la funzione interpolante sia una **buona approssimazione** di quella esatta dobbiamo chiederci se, calcolato l'errore delle funzioni in un insieme di punti, il **massimo dell'Errore sia piccolo**.

Se l'Errore è piccolo dipende da almeno due aspetti:

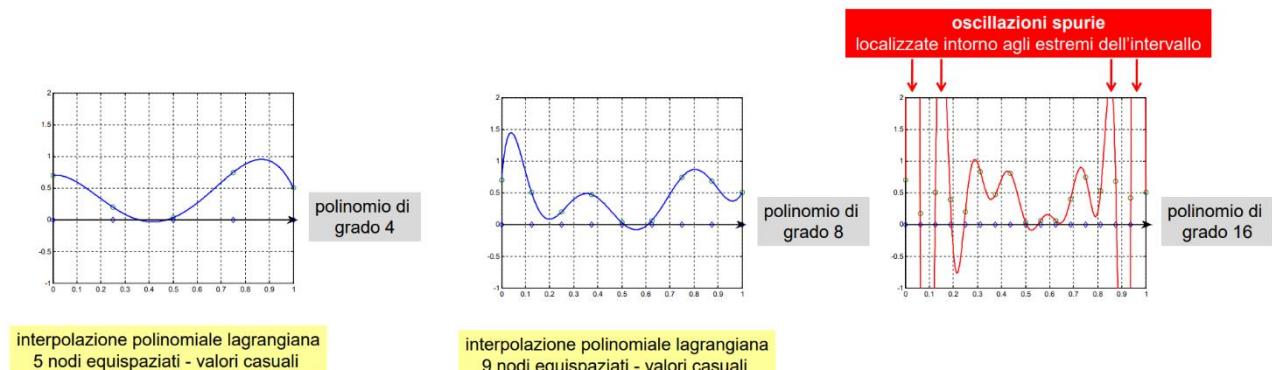
- distribuzione dei nodi;
- "qualità" di f (inteso come le sue proprietà analitiche).

Il numero di nodi influisce sulla qualità della ricostruzione, come influenza dipende dalle qualità della funzione e sulla disposizione dei nodi.

Un maggior numero di nodi produce un polinomio di grado maggiore, il quale avrà un maggior numero di massimi e di minimi, e quindi un **maggior numero di oscillazioni**.

Se i nodi sono **equispaziali**, la ricostruzione effettuata dal polinomio interpolante **peggiora** al crescere del numero di nodi, questo perché si verificano delle **oscillazioni spurie** (la funzione oscilla molto) localizzate intorno agli estremi dell'intervallo.

Invece, al centro la funzione viene approssimata abbastanza bene.



Una **soluzione** a questo problema **consiste nell'avere più nodi ai bordi e meno nodi al centro**, e possiamo farlo usando i **nodi di Chebyshev**.

I **nodi di Chebyshev** verranno generati partendo da una semicirconferenza (angolo π) la quale verrà divisa in **n-1 angoli uguali** (perché abbiamo n-1 settori).

L'ampiezza di tale settori sarà appunto data da: $\frac{\pi}{n-1}$.

Supponendo che l'ampiezza dell'angolo sia α , tale **punto di Chebyshev** sarà dato dalla **proiezione** delle coordinate $(\cos(\alpha), \sin(\alpha))$, ovvero dall'**ascissa** di tale punto.

Quindi i **punti di Chebyshev** saranno dati dalle ascisse di tali punti **moltiplicati per un certo valore $k = 0:n-1$** che rappresenta il “numero” del nodo a cui stiamo facendo riferimento.

Da notare che il **calcolo dei nodi deve essere fatto in senso antiorario**.

Dato un certo intervallo $[-1, 1]$, partiremo da 1 e andremo verso -1.

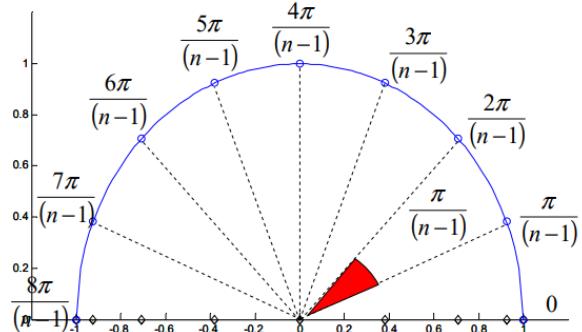
Possiamo prendere tali valori in senso **orario** negando i valori, ovvero negando i coseni. ($1, 0.9, 0.8\dots$ diventano $-1, -0.9, -0.8\dots$). Si tratta dello **standard utilizzato**.

In caso di un intervallo diverso da $[-1, 1]$ dovremo fare una **trasformazione lineare**. In generale consideriamo gli estremi a e b e il **punto medio** $\frac{a+b}{2}$.

Quello che osserviamo è che i nodi di Chebyshev sono **più densi** agli estremi dell'intervallo.

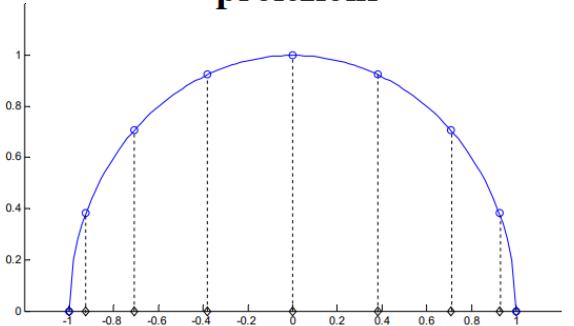
generazione di una griglia di n nodi di Chebychev

dividere l'angolo π in $n-1$ angoli uguali



$$x_k^{Cheb} = \cos\left(\frac{k\pi}{n-1}\right), k = 0 : n-1$$

proiezioni



le ascisse di tali punti sono i nodi di Chebychev

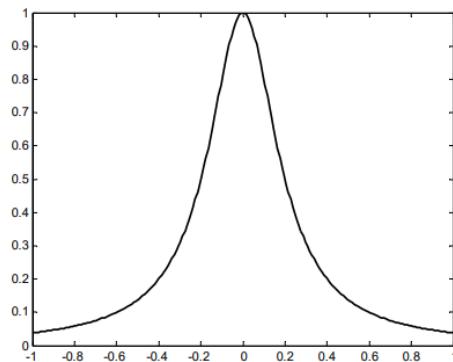
nodi di Chebychev
per un intervallo $[a,b]$

$$\rightarrow \frac{a+b}{2} + \frac{b-a}{2} x_k^{Cheb}, k = 0 : n-1$$

Verifichiamo quanto somigliano le due funzioni facendo il grafico dell'Errore.

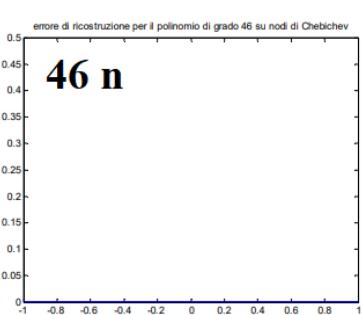
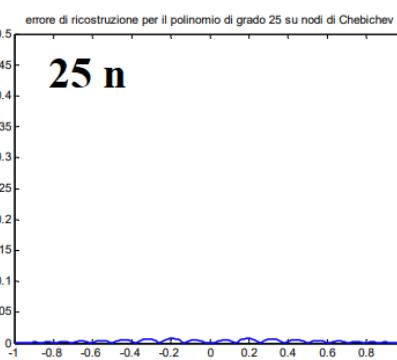
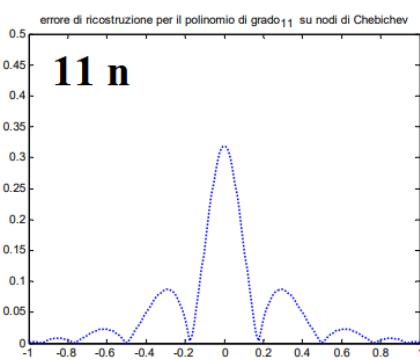
Notiamo come, se si usano i nodi di Chebychev, la ricostruzione effettuata dal polinomio interpolante **migliora** al crescere del numero di nodi.

Esempio:



$$f(x) = \frac{1}{1+25x^2}$$

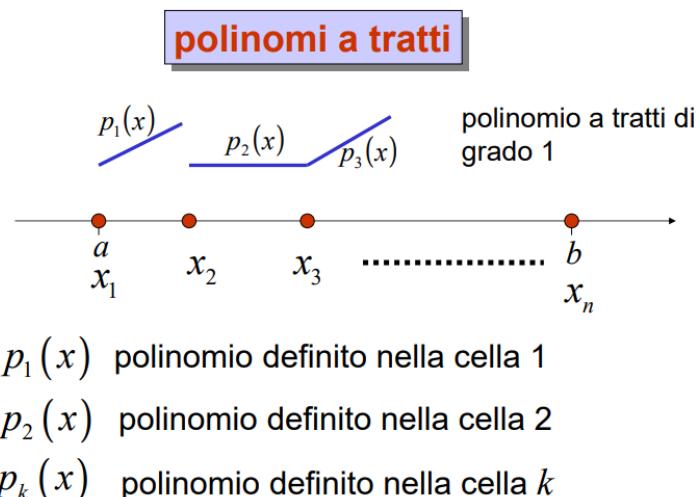
funzione di Runge



LEZIONE 15 – Interpolazione con polinomi a tratti

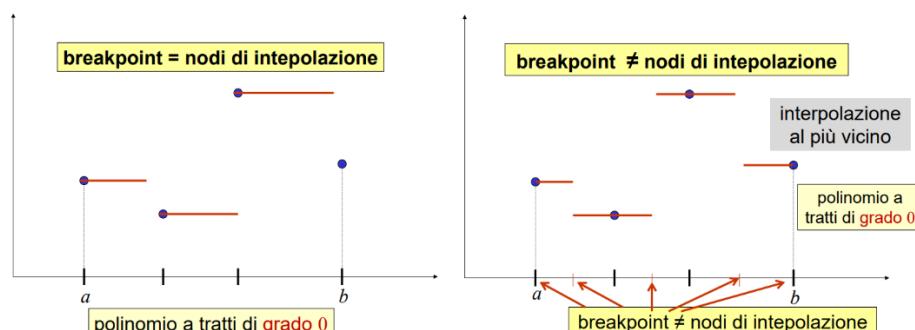
Per definire un **polinomio a tratti** in un intervallo $[a,b]$ bisogna preventivamente definire un insieme ordinato di m punti di $[a,b]$ detti **breakpoints**, e dove a e b devono essere breakpoints; ciò significa suddividere l'intervallo $[a,b]$ in $m-1$ (sotto-)intervalli (perché abbiamo n punti), detti **celle**, che hanno per estremi due breakpoints consecutivi.

Allora, un **polinomio a tratti $pp(x)$ di grado k** è una funzione definita in $[a,b]$ che in **ogni cella è un polinomio di grado al più k** . In altri termini, pp è una funzione costituita da $m-1$ pezzi, ognuno dei quali è un polinomio. Il grado di regolarità di pp dipende da come sono *attaccati* tra loro i vari pezzi che lo compongono. Si noti che, in tutte le ascisse che non sono breakpoints, un pp ha massima regolarità (è infinitamente derivabile, essendo un polinomio); le eventuali irregolarità, cioè le discontinuità di $pp(x)$ e delle sue derivate, sono limitate ai soli breakpoints.



Notiamo diverse cose: tale polinomio a tratti, in ogni cella, è di grado al più (1). Infatti tutti i polinomi a tratti sono di grado 1 o di grado 0. Notare inoltre che i polinomi a tratti sono regolari (continui) mentre può essere **discontinuo** (irregolare) solo nei breakpoint (è quello che accade nel punto x_2 , dove c'è un "salto").

Se consideriamo funzioni di grado 0, i breakpoint possono sia **uguali** ai nodi di interpolazione che **diversi**, in quest'ultimo caso si effettua una interpolazione al più vicino. Se i breakpoint sono uguali ai punti di interpolazione possiamo anche considerare funzioni di grado superiore.



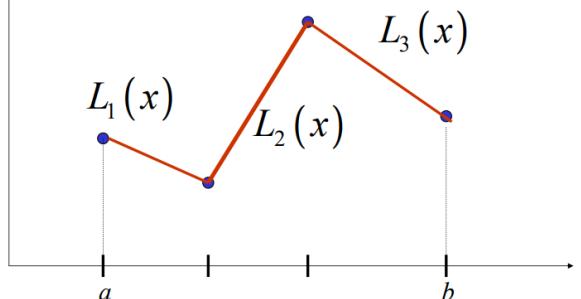
Un **polinomio a tratti di primo grado** è costituito da più polinomi lineari $L_1, L_2 \dots L_{n-1}$. Tali polinomi lineari possono essere trovati come: (nb: L_i corrisponde all'eq della retta).

$$L_i(x) = a_i + b_i(x - x_i) \quad i = 1 : n-1$$

$$a_i = y_i \quad b_i = \frac{(y_{i+1} - y_i)}{(x_{i+1} - x_i)}$$

$$L(x) = \begin{cases} L_1(x) & se \quad x_1 \leq x < x_2 \\ L_2(x) & se \quad x_2 \leq x < x_3 \\ L_3(x) & se \quad x_3 \leq x < x_4 \end{cases}$$

grafico del polinomio a tratti di grado 1 $L(x)$



b_i è calcolato in questo modo perché b_i è il coefficiente angolare di una retta, e per calcolarla basta ricordare che è uguale al rapporto fra la differenza delle y e la differenza delle x .

$L(x)$ rappresenta l'espressione del polinomio lineare a tratti.

SPLINE LINEARE

Quello che noi chiamiamo comunemente “**polinomio lineare a tratti**” prende anche il nome di **spline lineare** (spline di primo grado), la quale è una **funzione continua** in $[a,b]$ con **derivata prima** (eventualmente) **discontinua** (la discontinuità può essere solo nei breakpoint).

NB:

- Il **grado** della spline interpolante **non** dipende dal numero dei nodi.
- se la spline lineare avesse derivata prima continua, allora la spline lineare sarebbe un polinomio di grado 1.

SPLINE CUBICA

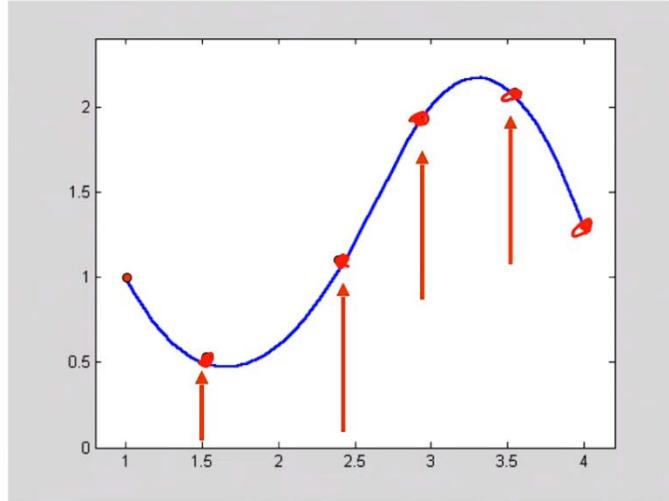
La **spline cubica** è un polinomio a tratti di **terzo grado**. Si tratta di una **funzione continua** in $[a,b]$, e con **derivata prima e derivata seconda continua** in $[a,b]$.

La **derivata terza** è (eventualmente) **discontinua** (la discontinuità può essere solo nei breakpoint).

NB:

- L'insieme delle spline cubiche contiene strettamente l'insieme dei polinomi di grado 3.
- una spline cubica con derivata terza continua è un polinomio di terzo grado.

spline cubica interpolante (su 6 punti)



Da notare come l'occhio umano non sia in grado di "apprezzare" una discontinuità della derivata terza. Per tale motivo il grafico di una spline cubica sembra avere un 'andamento molto liscio per l'occhio umano.

espressione della spline cubica $s(x)$

$$s(x) = \begin{cases} s_1(x) & \text{se } x_1 \leq x < x_2 \\ s_2(x) & \text{se } x_2 \leq x < x_3 \\ \dots & \dots \\ s_k(x) & \text{se } x_k \leq x < x_{k+1} \end{cases}$$

$$s_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

OSSERVAZIONI:

- Una **spline** può non essere un polinomio;
- un **polinomio** è un particolare tipo di spline;
- la **spline cubica** è il più regolare dei polinomi a tratti di grado 3 (esclusi i polinomi).

L'**obiettivo** sarà trovare s_i , ma per far ciò dobbiamo prima trovare a_i, b_i, c_i e d_i .

LEZIONE 16 – Continuo Spline cubiche e Cubiche di Hermite

Una **spline cubica** viene rappresentata come un polinomio a tratti, ha **4 coefficienti** per ogni cella (sottointervallo) ed ha **n breakpoint** (n-1 celle). Tenendo in considerazione ciò, avremo **4*(n-1)** soluzioni, che possiamo conservare all'interno di una matrice $4^*(n-1)$.

(Ciò significa che se abbiamo 3 punti avremo solo $s_1(x)$ e $s_2(x)$, ovvero $3-1 = 2$ celle.
Abbiamo 4 coefficienti per celle, quindi $4*2 = 8$ incognite).

a	b	c	d
	b_3		

coefficienti di $s_1(x)$
coefficienti di $s_2(x)$
coefficienti di $s_3(x)$
.....
coefficienti di $s_k(x)$

CONDIZIONI DA IMPORRE

Le condizioni da imporre riguardano: l'**interpolazione**, la **continuità della derivata prima** e la **continuità della derivata seconda**.

Possiamo osservare che nel caso dell'**interpolazione**, ci ritroviamo con dei punti "ridondanti". Ad esempio se avessimo 3 punti (quindi due $s(x)$) per interpolare tale spline ci ritroveremmo in questa situazione:

$$s_1(x_1)=y_1, s_1(x_2)=y_2, s_2(x_2)=y_2, s_2(x_3)=y_3 \dots$$

notiamo come $s_1(x_2)$ e $s_2(x_2)$ abbiano lo stesso valore. Questo vale in generale per tutti i punti ad eccezione dei due punti sugli estremi. (**2n-2 condizioni**);

Imponendo le condizioni di **continuità della derivata prima** (nei breakpoints interni), si ottiene:

$$s_1'(x_2)=s_2'(x_2), s_2'(x_3)=s_3'(x_3) \dots \quad (\text{n-2 condizioni});$$

Avviene la stessa cosa anche imponendo le condizioni di **continuità della derivata seconda** (nei breakpoints interni), infatti si ottiene:

$$s_1''(x_2)=s_2''(x_2), s_2''(x_3)=s_3''(x_3) \dots \quad (\text{n-2 condizioni});$$

In totale, tali condizioni danno luogo a un **sistema di 4n-6 condizioni** in **4n-4 incognite**.

Collegandoci all'esempio di prima, se $n=3$ (3 punti), allora avremo 4 condizioni per l'interpolazione, 2 condizioni per la continuità della derivata prima e 2 condizioni per la continuità della derivata seconda, per un TOTALE di 6 condizioni.

È anche possibile aggiungere, in modo arbitrario, altre **2 condizioni**: ovvero che la derivata terza sia continua nel **secondo** e nel **penultimo** breakpoint.
Così facendo renderemmo quadrato il sistema.

Si tratterà di risolvere il sistema: **$A_c = v$** .

L'ordine del sistema sarà $4(n-1) \times 4(n-1)$. (non si tratta di un sistema piccolo)

Se si scelgono le condizioni $s_1''(x_1) = s_{m-1}''(x_m) = 0$, si hanno le cosiddette **spline naturali**.
In Matlab invece si usano due diverse condizioni (dette **not a knot**). (esempio sul file MATLAB).

Le *spline naturali* hanno la proprietà di rendere minimo l'integrale del quadrato della derivata seconda, ed hanno quindi una proprietà di **non oscillazione** (o meglio, meno oscillazioni) rispetto a tutte le altre spline cubiche.

FUNZIONE CUBICA DI HERMITE A TRATTI

Una funzione cubica di Hermite a tratti è un **polinomio a tratti di terzo grado**. Si tratta di una funzione **continua** in $[a,b]$ e con **derivata prima continua** in $[a,b]$.

La **derivata seconda** è (eventualmente **discontinua**) (la discontinuità può presentarsi solo nei breakpoint).

Si tratta di una spline che richiede ancora meno vincoli, e per tal motivo è più ambigua.

L'insieme delle cubiche di Hermite a tratti **contiene strettamente** l'insieme delle spline cubiche (cioè le spline cubiche sono un sottoinsieme delle cubiche di Hermite).

Una **cubica di Hermite con derivata seconda continua** è una **spline cubica**.

L'idea delle cubiche di Hermite è quella di **conservare la forma** (shape preserving), ovvero si tratta di un'interpolazione a **conservazione di forma**.

L'obiettivo è: dati n punti da interpolare, si vuole costruire una cubica di Hermite a tratti che **non oscilli** tra nessuna coppia di nodi consecutivi.

Per ottenere tale risultato i valori delle **pendenze** dei nodi devono essere **automaticamente determinati** in modo che la funzione interpolanti non oscilli intorno ad ogni nodo.

Se teniamo in considerazione le *condizioni di interpolazioni* e le *condizioni della derivata prima*, in entrambi i casi 2 per ogni nodo interno e 1 per i nodi esterni ($n+n-2$), e NON dovendo tenere in considerazione le due condizioni aggiuntive, avremo che:

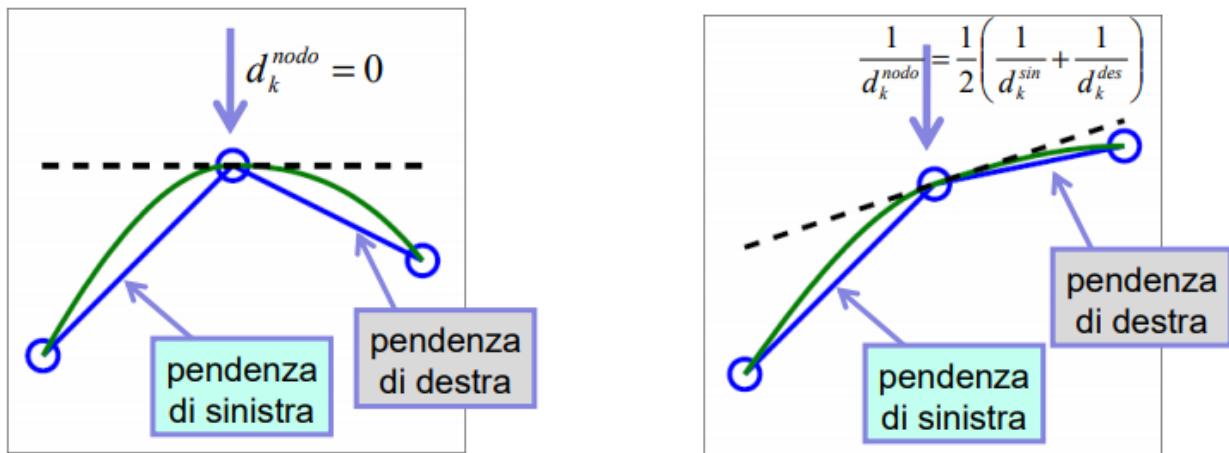
In totale, tali condizioni danno luogo a un **sistema di $4n-4$ condizioni in $4n-4$ incognite**.

Si tratterà di risolvere il sistema: **$H_c = b$** .

L'ordine del sistema sarà $4(n-1) \times 4(n-1)$.

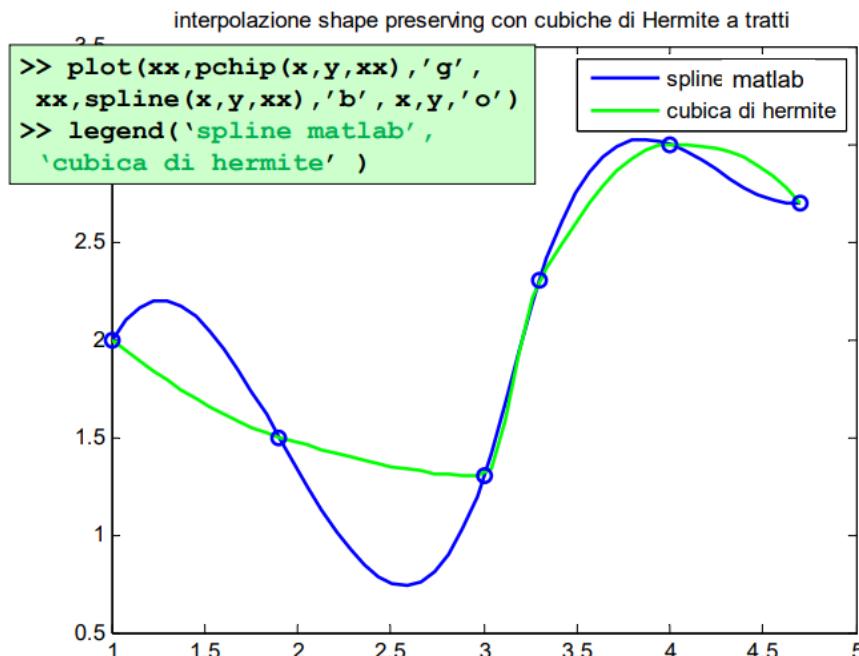
Per **fissare le derivata** dobbiamo tenere in conto due diverse situazioni:

- presi 3 valori, se la pendenza della retta cambia passando dal 2 punto allora possiamo considerare tale punto come un massimo, e la **pendenza** di una retta in un punto di massimo è **0**.
Quindi considerate le due curve (tratti) $s_1(x)$ e $s_2(x)$, se queste due hanno pendenze con **segno opposto**, dovremo avere che $s_1'(x_2) = 0$ e $s_2'(x_2) = 0$.
- presi 3 valori, se la pendenza della retta non cambia passando dal 2 punto allora considereremo una pendenza **intermedia** fra le pendenze delle due curve.
Quindi considerate le due curve (tratti) $s_1(x)$ e $s_2(x)$, se queste due hanno pendenze con **segno uguale**, dovremo fare la **media armonica** fra le due pendenze (è la *media dell'inverso delle pendenze*).



Notare come un'interpolazione effettuata in questo modo provochi **meno oscillazioni** rispetto ad una normale spline cubica.

(nb: In MATLAB si usa la funzione `pchip(nodi, ordinate, griglia)`).



Per renderci conto della fedeltà delle varie approssimazioni possiamo prendere in considerazione la **funzione di Runge**.

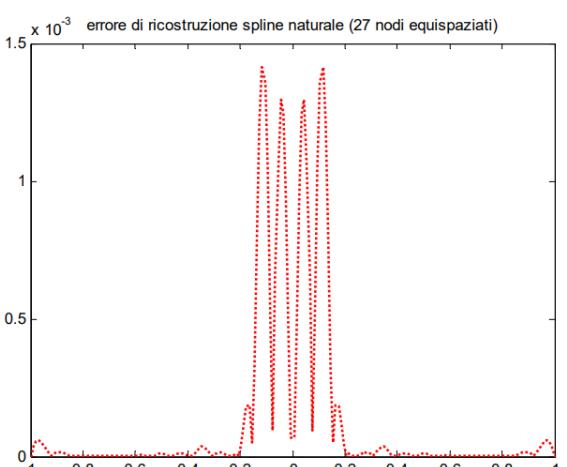
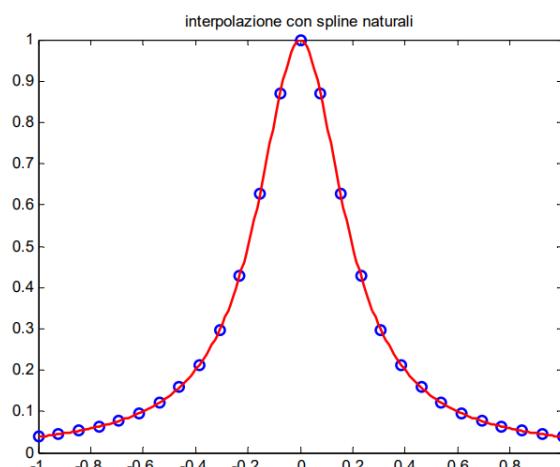
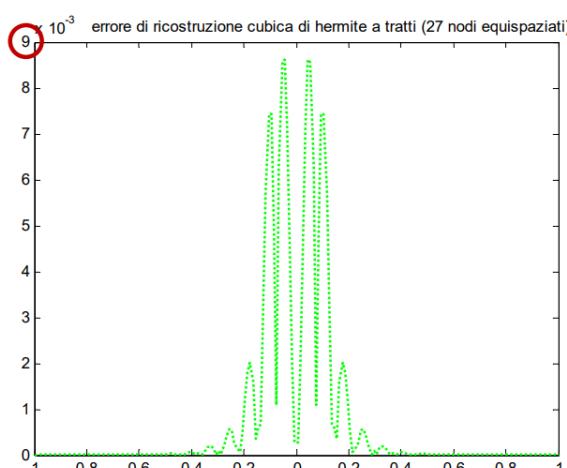
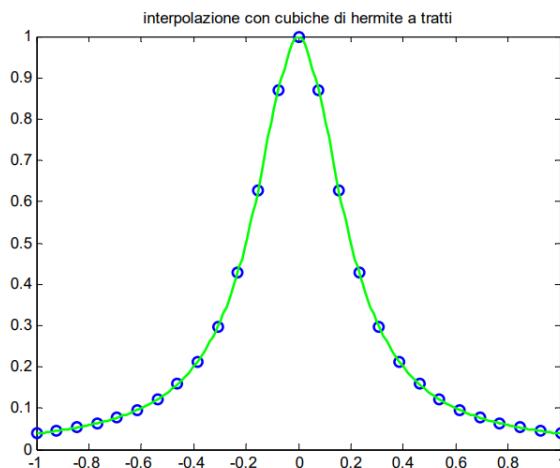
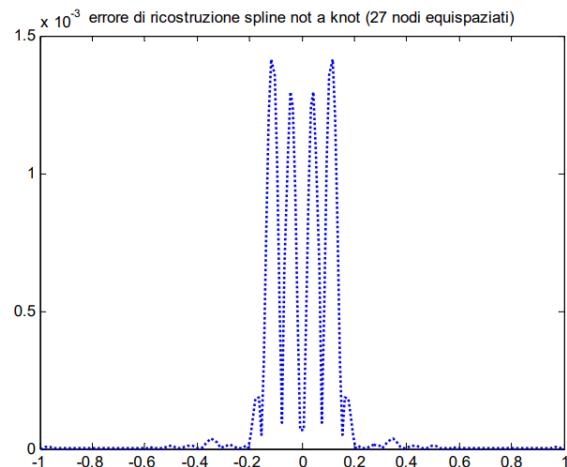
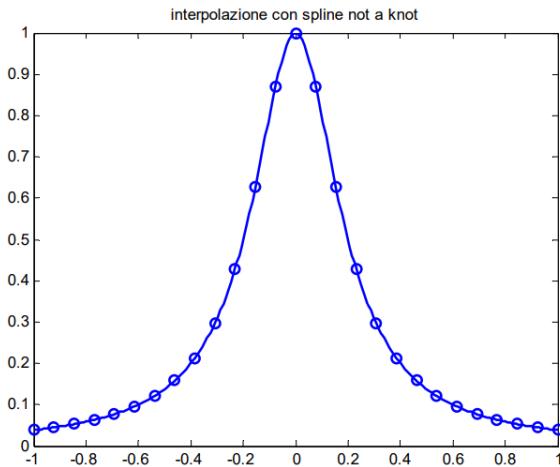
Misuriamo l'errore di *interpolazione* come la norma della differenza fra la funzione esatta e la funzione approssimata (oppure, possiamo considerare l'*errore puntuale* (è uguale)).

Il valore da considerare sarà l'**errore di valore massimo**.

$$\text{err} = \| f(x) - s(x) \|$$

oppure

$$\text{err}_i = f(x_{xi}) - s(x_{xi})$$



Notare come la cubica di Hermite conservi meglio la forma ma non è capace di approssimare bene la funzione quanto fa invece la spline. Tuttavia, entrambe possono andare bene.

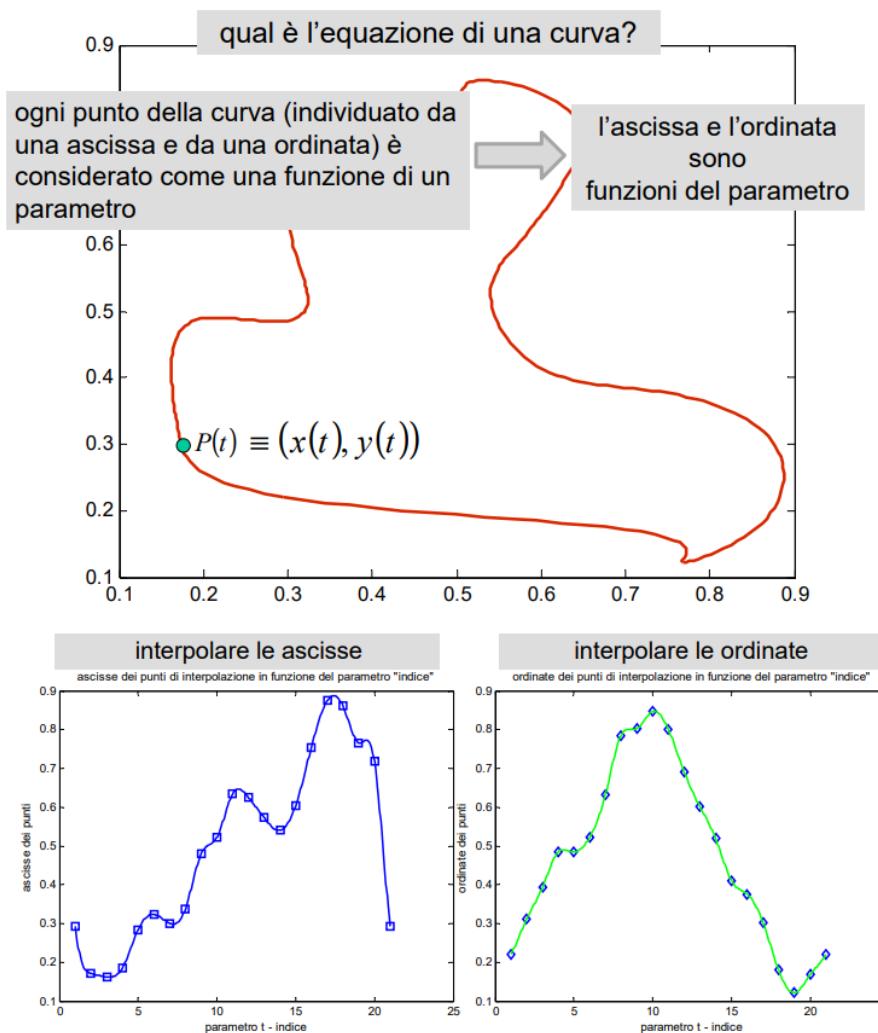
COSTRUIRE CURVE PARTENDO DA PUNTI NEL PIANO DELLO SPAZIO

Le funzioni **spline** e le funzioni **cubiche di Hermite** a tratti sono tra i principali strumenti della **computer graphics**, la quale è basata sull'interpolazione 1D e sull'interpolazione 2D (per questa viene utilizzata una funzione di *due* variabili (x,y) , il cui grafico è una superficie (la variabile dipendente è la *quota z*).

In un **caso 1D**, il **problema** è che: dati n punti nel piano, ordinati in sequenza, determinare una **curva continua e liscia** che passi per n punti (ove $p=(x,y)$).

Ricordiamo che con **gamma** facciamo riferimento al sostegno della curva, e $x = x(t)$ e $y = y(t)$, e dove $t \in [a,b]$. Ricordiamo inoltre che $p(\bar{t}) = (p(\bar{x}), p(\bar{y}))$ identifica un punto della curva e che $p(a) = p(b)$ (la curva si "chiude").

Per ottenere **l'interpolazione di una curva parametrica** dovrò interpolare sia le **ascisse** che le **ordinate**.



Notare come quando termina il “giro”, la x torna dov’era (parte da 0.3 e torna a 0.3) e anche la y torna dov’era (parte da circa 0.2 e torna a circa 0.2).

nb: l’asse verticale rappresenta la x e la y rispettivamente nel primo e nel secondo grafico, mentre l’asse orizzontale rappresenta la t (abbiamo 21 punti).

L’interpolazione delle ascisse e delle ordinate può essere effettuata con **tutti** i metodi visti precedentemente (spline, pchip, ecc...).

Una spline con condizione di periodicità agli estremi, ovvero è una spline in cui la derivata prima e seconda in x_1 deve essere uguale alla derivata prima e seconda in x_n , e dove alla fine dobbiamo raccordarci.

(nb: in MATLAB dovremmo fare l’interpolazione delle ascisse e delle ordinate, quindi dovremo avere due spline. Quando andremo a fare il plot invece di passare una sola spline ne passeremo 2).

LEZIONE 17 – Fitting di dati: l'approssimazione (ricostruzione di trend)

La **ricostruzione approssimata** viene utilizzata quando abbiamo **a disposizione molti dati**, eventualmente con **errore**, i quali **rivelano un trend**.

Tale ricostruzione **prende il nome di approssimazione**.

L'idea è simile a quella dell'interpolazione, in quanto il nostro **scopo** sarà sempre quello di **trovare** (se esiste) un **descrittore f** nell'insieme **modello F** tale che $f(x_i) - y_i$ sia **piccolo**, ovvero voglio un descrittore f , che calcolato in x_i , abbia una distanza da y_i piccola (al contrario dell'interpolazione dove $f(x_i) = y_i$). Fondamentalmente **vogliamo una linea che passa vicino ai punti**.

$$f \text{ in } F \text{ tale che: } |f(x_i) - y_i| \text{ è piccolo, } i=1:m;$$

Diamo per **assunto** che l'errore dipenda solo nella variabile **dipendente**. Dallo **scostamento** dei m punti di una curva $y=f(x)$ otteniamo il **vettore degli scostamenti elementari** (o vettore scostamento o vettore *residuo*) $r = (r_1, r_2 \dots r_m)$.

$$r_i = f(x_i) - y_i, \quad \text{ove } \|r\|_2 \text{ è piccolo}$$

ovvero, lo **scostamento** di una funzione da un **insieme di punti** è **misurato** dalla **grandezza** del **vettore degli scostamenti elementari**, ovvero dalla **norma** di tale vettore.

La ricerca della minima norma, su spazi vettoriali di dimensione **finita**, ha **sempre soluzione**. Le scelte più comuni di norma sono la norma 2, la norma 1 e la norma infinito (di cui preferiamo la norma 2).

Questo tipo di approssimazione prende il nome di approssimazione dei **Minimi Quadrati**, in quanto voglio che sia *più piccola possibile* la somma dei quadrati degli scostamenti elementari. $(r_1^2 + r_2^2 + \dots + r_m^2)$.

Lo scopo sarà quindi trovare (se esiste) f in F che **rende minima** la norma 2. Ricordando inoltre che " $r_i = f(x_i) - y_i$ " avremo che:

$$\|r\|_2 = \sqrt{(f(x_1) - y_1)^2 + (f(x_2) - y_2)^2 + \dots + (f(x_m) - y_m)^2}$$

è equivalente a trovare f che **rende minima**:

$$\|r\|_2^2 = (f(x_1) - y_1)^2 + (f(x_2) - y_2)^2 + \dots + (f(x_m) - y_m)^2$$

Questo significa che se un residuo, per una certa f , è il più piccolo possibile, allora per la stessa f , quel residuo al quadrato sarà il più piccolo possibile.

Cioè il punto di minimo della funzione *su* coincide con il punto di minimo della funzione *giù*.

Quindi, per semplificarci i conti, invece di fare il minimo della norma 2 di r (che comporta anche la presenza di una radice quadrata) facciamo il punto di **minimo del quadrato della norma**.

L'insieme F (modello lineare) è costituito da funzioni che sono univocamente determinate da **n coefficienti**.

Quindi supponiamo che la **funzione da trovare** sia data dalla combinazione di più funzioni di **base** delle quali dobbiamo individuare i **coefficienti**.

Quindi **per trovare la f** che rende minima la norma due **dobbiamo individuare i coefficienti**.

In questo caso, in generale, **$m >> n$** , ovvero potrei avere migliaia di punti e solo pochi coefficienti (es. $m=1000$, $n=2$).

Quindi, al contrario dell'interpolazione dove $m = n$ in quanto necessitavamo di un sistema quadrato, **in questo caso**, in generale, ne avremo uno **rettangolare**. (**sistema rettangolare**)

Da notare che i coefficienti **influiscono** sulla ricerca del minimo della norma 2 al quadrato (*infatti potremmo vedere $\|r\|_2^2$ come una sommatoria di sommatorie. Infatti $\|r\|_2^2$ è individuabile sommando tutte le r_i^2 , e dove per ogni r_i^2 dobbiamo individuare un $f(x_i)$ che a sua volta richiede una sommatoria (e nel quale compare il coefficiente a).*

Le incognite del problema sono gli **n** valori dei coefficienti che minimizzano la **funzione** $\|r\|_2^2 = g(a_1, a_2, \dots, a_n)$, che si tratta di una funzione **convessa**, da cui sappiamo poter trovare il punto di minimo nel punto in cui il gradiente si annulla. (derivate parziali uguali a 0).

Quindi **dovremo porre le derivate parziali del residuo, rispetto ai coefficienti, uguali a 0:**

$$\frac{\partial}{\partial a_i} \|r\|_2^2 = 0 \quad , i = 1, \dots, n$$

sistema di **n** equazioni in **n** incognite

maggiore è il numero di coefficienti **n**, maggiore sarà il numero di derivate parziali da fare. **Il punto in cui si annullano tutte le derivate parziali sarà il punto di minimo.**

Esempio: **retta** dei minimi quadrati

✓ **$n = 2$**

✓ modello $\underline{f(x)} = \underline{a_1} + \underline{a_2}x$

obiettivo:

calcolare i **2** coefficienti a_1, a_2 tali che

$$\frac{\partial}{\partial a_1} \|r\|_2^2 = 0 \quad , \quad \frac{\partial}{\partial a_2} \|r\|_2^2 = 0$$

$b_1(x) = 1$
 $b_2(x) = x$

$\nabla r^2 = (f(x_n) - y_n)^2 =$
 $(a_1 + a_2 x_n - y_n)^2$

$\left\{ \begin{array}{l} \frac{\partial g}{\partial a_1} = 0 \\ \frac{\partial g}{\partial a_2} = 0 \end{array} \right.$

(approccio di **analisi matematica**).

$$\frac{\partial}{\partial a_1} \sum_{i=1}^m [(a_1 + a_2 x_i) - y_i]^2 = 0$$

$$\frac{\partial}{\partial a_2} \sum_{i=1}^m [(a_1 + a_2 x_i) - y_i]^2 = 0$$



per la proprietà della sommatoria possiamo prima fare la derivata e poi la sommatoria:

$$\cancel{\sum_{i=1}^m} [(a_1 + a_2 x_i) - y_i] = 0$$

$$\cancel{\sum_{i=1}^m} [(a_1 + a_2 x_i) - y_i] (x_i) = 0$$

Da cui possiamo ottenere il **sistema in 2 equazioni m in 2 incognite n** :

$$\left(\sum_{i=1}^m a_1 + \sum_{i=1}^m a_2 x_i \right) = \sum_{i=1}^m y_i$$

$$\left(\sum_{i=1}^m x_i a_1 + \sum_{i=1}^m a_2 x_i^2 \right) = \sum_{i=1}^m y_i x_i$$

Ergo, la **retta** dei minimi quadrati può essere descritta in **forma matriciale** come:

$$\begin{pmatrix} m & \sum_{i=1}^m x_i \\ \sum_{i=1}^m x_i & \sum_{i=1}^m x_i^2 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^m y_i \\ \sum_{i=1}^m y_i x_i \end{pmatrix}$$

sistema lineare
2 equazioni
2 incognite

nb: notare come il numero dei nodi entri in gioco attraverso m , e dal fatto che dovremo effettuare le sommatorie di m nodi.

IN GENERALE, un **modello** a n coefficienti è un sistema lineare a n equazioni e n incognite, e prende il nome di **sistema delle equazioni normali** (in quanto definiscono una condizione di perpendicolarità).

nb: l'ordine del sistema non dipende dal numero dei dati.

OSSERVAZIONE: se abbiamo $n = m$, il modello più vicino che passa per quei punti è proprio il **modello interpolante**. (es. dati 3 coefficienti e 3 punti, il modello che passerà più vicino a questi punti sarà una parabola).

Per tale motivo possiamo dire che il **modello interpolante è il modello approssimante ma con residuo nullo**.

(Infatti, in MATLAB polyfit non nasce per risolvere i problemi di interpolazione ma quelli di approssimazione).

È anche possibile approcciarsi al problema dei **Minimi Quadrati** dal punto di vista dell'**algebra lineare**.

Esempio: **retta** dei minimi quadrati

$$\checkmark \quad n = 2$$

$$\checkmark \quad \text{modello } f(x) = a_1 + a_2 x$$

componenti
del
**vettore
residuo**

$$\begin{aligned} r_1 &= f(x_1) - y_1 \\ r_2 &= f(x_2) - y_2 \\ \dots &\dots \\ r_m &= f(x_m) - y_m \end{aligned}$$

$$\begin{aligned} r_1 &= a_1 + a_2 x_1 - y_1 \\ r_2 &= a_1 + a_2 x_2 - y_2 \\ \dots &\dots \\ r_m &= a_1 + a_2 x_m - y_m \end{aligned}$$

$$a_1 + a_2 x_1 = y_1$$

$$a_1 + a_2 x_2 = y_2$$

.....

$$a_1 + a_2 x_m = y_m$$

sistema lineare
 m equazioni
2 incognite

in generale,
non ammette
soluzione

e non ammette soluzione (in generale) perché il nostro scopo non è che tale valori siano uguali ma che l'errore sia piccolo.

Possiamo invece **ragionare con i vettori**.

$$\begin{aligned} r_1 &= a_1 + a_2 x_1 - y_1 \\ r_2 &= a_1 + a_2 x_2 - y_2 \\ r_3 &= a_1 + a_2 x_3 - y_3 \end{aligned}$$

↓ ↓ ↓

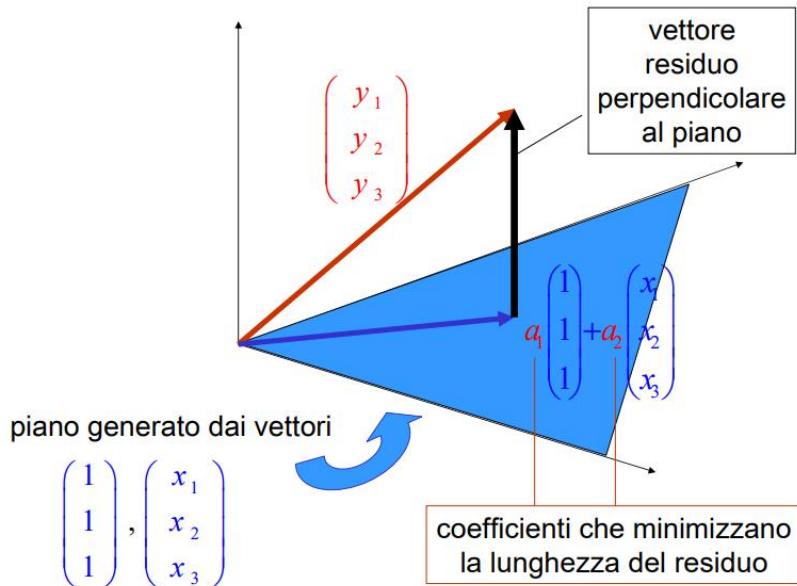
vettore
nello
spazio 3D

vettore
in un piano

vettore
nello
spazio 3D

Li chiamiamo, in ordine, vettori **z, v e w**.

Il vettore con il **residuo minore** sarà quello perpendicolare al piano. Avremo la seguente situazione:



2 vettori sono **perpendicolari** se e solo se il loro prodotto scalare è nullo. Ciò significa che il vettore **r** e il vettore **v** devono essere perpendicolari.

Notare come il vettore **v** possa essere visto come:

$$a_1 \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} + a_2 \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Da ciò, possiamo notare inoltre come il vettore **v** possa essere scritto come **Ba**, con:

$$B = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{pmatrix} \quad a = (a_1, a_2)^T$$

allora, il **vettore residuo** è:

$$r = B a - y$$

Se ricordiamo che 2 vettori sono **perpendicolari** se e solo se il loro prodotto scalare è nullo, significa che il vettore r deve essere **perpendicolare** alle colonne di B (ovvero, alle righe di B^T).

Tutto ciò comporta che:

$$B^T (B \mathbf{a} - \mathbf{y}) = 0$$

ovvero:

$$B^T B \mathbf{a} = B^T \mathbf{y}$$

dove $B^T B$ è una **matrice quadrata**, esattamente fatta dal numero di equazioni ed incognite del numero di funzioni base del numero di colonne di B .

Ergo, la **retta** dei minimi quadrati può essere descritta in **forma matriciale** come:

$$B^T B \mathbf{a} = B^T \mathbf{y}$$



$$\begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_m \end{pmatrix} \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_m \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_m \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$



$$\begin{pmatrix} m & \sum_{i=1}^m x_i \\ \sum_{i=1}^m x_i & \sum_{i=1}^m x_i^2 \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^m y_i \\ \sum_{i=1}^m y_i x_i \end{pmatrix}$$

LEZIONE 18 – Continuo approssimazione, SL Sovradeterminati e Quadratura

Esempio: **parabola** dei minimi quadrati

✓ $n = 3$

✓ modello $f(x) = a_1 + a_2 x + a_3 x^2$

$$B = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_m & x_m^2 \end{pmatrix} \quad \mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

Notiamo come sia equivalente al caso della *retta dei minimi quadrati*, dove le funzioni base sono $b_1 = 1$, $b_2 = x$ e $b_3 = x^2$, e per ogni funzione base abbiamo m nodi.

Anche la **risoluzione** del problema è equivalente.

(Come nel caso della retta) Diamo per **assunto** che l'errore dipenda solo nella variabile **dipendente**. Dallo **scostamento** dei m punti di una curva $y=f(x)$ otteniamo il **vettore degli scostamenti elementari** (o vettore *scostamento* o vettore *residuo*) $r = (r_1, r_2 \dots r_m)$.

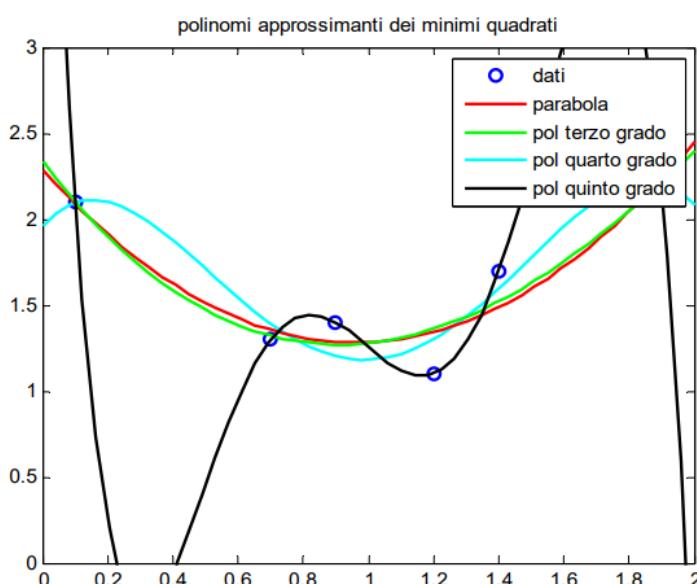
$$r_i = f(x_i) - y_i, \quad \text{ove } \|r\|_2 \text{ è piccolo}$$

ovvero, lo **scostamento** di una funzione da un **insieme di punti** è **misurato** dalla **grandezza** del **vettore degli scostamenti elementari**, ovvero dalla **norma** di tale vettore.

Non è detto che **AUMENTANDO IL GRADO** sia abbia una miglior rappresentazione.

Infatti più è grande il grado maggiore sarà il numero di oscillazioni.

Il grado non può essere maggiore del *numero di punti* + 1, in quanto se $n = \text{grado} + 1$ non **abbiamo** più un'approssimazione ma una **interpolazione**. Invece se $n > \text{grado} + 1$ il problema è **mal posto** in quanto non abbiamo una soluzione sola.



È anche possibile approssimare i dati con il **modello trigonometrico**, solo che per loro non abbiamo una funzione analoga a *polyfit*.

Ciò significa che dovremmo utilizzare la **procedura generale**: $B^T B \mathbf{a} = B^T \mathbf{y}$.

RIEPILOGANDO

modello $f(x) = a_1 b_1(x) + a_2 b_2(x) + \dots + a_n b_n(x)$

Interpolazione

B è quadrata $n \times n$



risolvere

$$B \mathbf{a} = \mathbf{y}$$

Minimi Quadrati

B è rettangolare $m \times n$



risolvere

$$B^T B \mathbf{a} = B^T \mathbf{y}$$

SISTEMI LINEARI SOVRADETERMINATI

Come ben ricordiamo, in **generale**, un sistema lineare $\mathbf{Ax} = \mathbf{b}$ di m equazioni ed n incognite, dove $m > n$, non ammette soluzione.

Tuttavia posso richiedere che $Ax = b$ sia **piccolo** (in valore assoluto), e quindi avrò l'**obiettivo di trovare la x dei minimi quadrati** (x_{MQ}).

L'idea è quella di risolvere il sistema in **modo approssimato**. Per far ciò dovremo usare i **minimi quadrati**, ed avrò dunque:

$$A^T A x_{MQ} = A^T b$$

Così facendo avremo la **soluzione nel senso dei minimi quadrati**.

La scelta dell'insieme **modello F** deve essere fatta in modo tale da:

- utilizzare le informazioni **a priori** sul fenomeno descritto dai dati (scelta delle *funzioni base*);
 - **filtrare** (eliminare) **l'errore** sui dati (scelta del *numero n dei coefficienti incogniti* del modello).

$$A^T A x_{MQ} = A^T b$$

La **complessità di tempo** è $T(m,n) = O(m \cdot n^2 / 2) + O(n^3 / 3)$.

QUADRATURA (Integrazione Numerica)

Ricordando che un **Integrale definito** dipende dalla funzione f e dai due estremi a, b , avremo che:

$$I [f, a, b] = \int_a^b f(x) dx$$

Un possibile **approccio** per calcolare l'area sotto il grafico della funzione consiste nell'**approccio geometrico**, che riguarda l'utilizzo di **figure geometriche elementari**. (In questo corso almeno NON utilizziamo le primitive F , ma solo la funzione stessa).

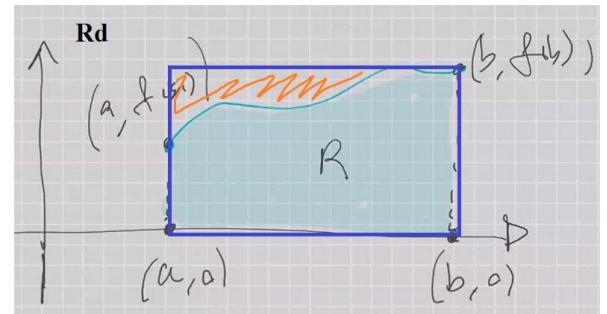
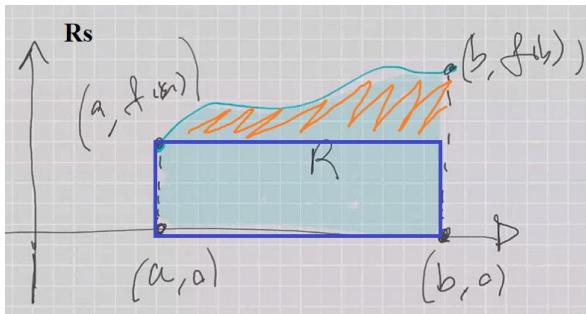
In generale, diremo che $I(f) \approx Q(f)$ (*quadratura di f*).

Una quadratura ha dei **nodi x** e dei **pesi w**.

es. Rettangolo: $\int_a^b f(x) dx \approx (b - a) * f(a)$

$\int_a^b f(x) dx \approx w_1 f(x_1)$ formula rettangolare sinistra = R_s $x_1 = a$ $w_1 = b-a$

$\int_a^b f(x) dx \approx w_1 f(x_1)$ formula rettangolare destra = R_d $x_1 = b$ $w_1 = b-a$



Ogni qualvolta utilizziamo una formula di quadratura viene generato un *errore*, detto **errore di quadratura**, che corrisponde all'area dell'integrale non compresa nella quadratura: $E(f) = I(f) - Q(f)$

In generale, $Q(f)$ è definito come: $Q(f) = w_1 f(x_1) + w_2 f(x_2) \dots w_n f(x_n)$

Un altro approccio è l'**approccio interpolante**. Infatti se sappiamo che $f(x)$ passa per n punti, possiamo **creare una funzione interpolante $g(x)$ che passa per quei punti**.

Avremo che: $f(x_i) = g(x_i)$ e $f(x) \approx g(x)$

Ciò significa che l'integrale di $f(x)$ sarà *simile* a quello di $g(x)$. Ciò è conveniente perché possiamo costruire $g(x)$ utilizzando solo quei n numeri, e di conseguenza l'integrale di $g(x)$ utilizzerà solo quelle n quote.

FORMULE DI QUADRATURA

Le **formule di quadratura** traggono la loro giustificazione teorica direttamente dalla definizione di integrale definito, seconda la costruzione di Riemann (**somme di Riemann**).

$$\int_a^b f(x) dx \approx \sum_{i=1}^n w_i f(x_i)$$

Un'idea è quella di passare per l'**interpolazione**, quindi si tratta di **approssimare** l'integrale di f (area con segno del rettangoloide di f) con l'integrale di una **funzione g che interpola** la f , e che sia semplice da integrare.

$$\int_a^b f(x) dx \approx \int_a^b g(x) dx$$

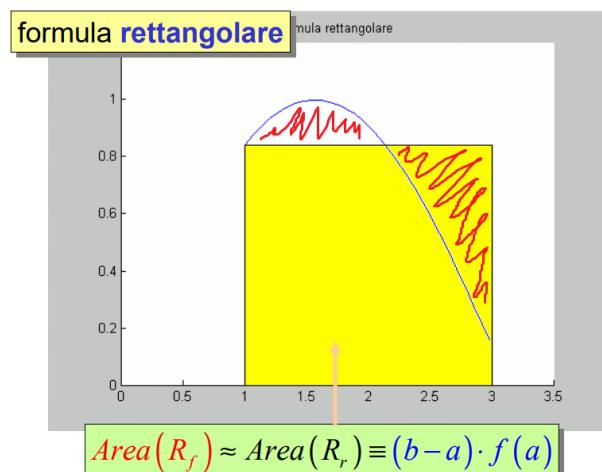
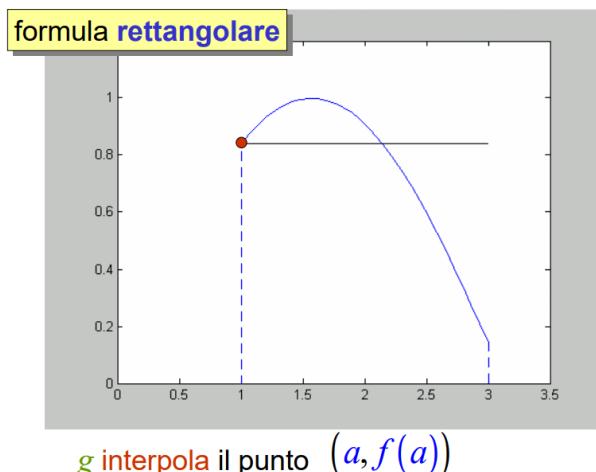
Ovviamente le due funzioni saranno (probabilmente) simili ma NON uguali, quindi si genererà un errore.

FORMULA RETTANGOLARE

Prendiamo in considerazione un solo punto $(x_1, f(x_1))$, per il quale passerà il polinomio interpolante $g(x) = p_0(x)$. Quindi avremo che $p_0(x_1) = f(x_1)$.

Prendiamo in considerazione l'**area** sotto la costante p_0 , che altro non è che un rettangolo.

Lo scopo sarà quindi calcolare l'area di tale rettangolo.



(La zona gialla rappresenta la stima dell'integrale, l'area tratteggiata in rosso rappresenta l'errore).

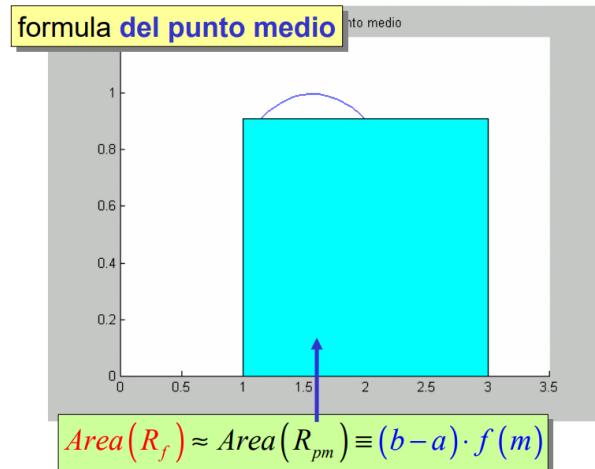
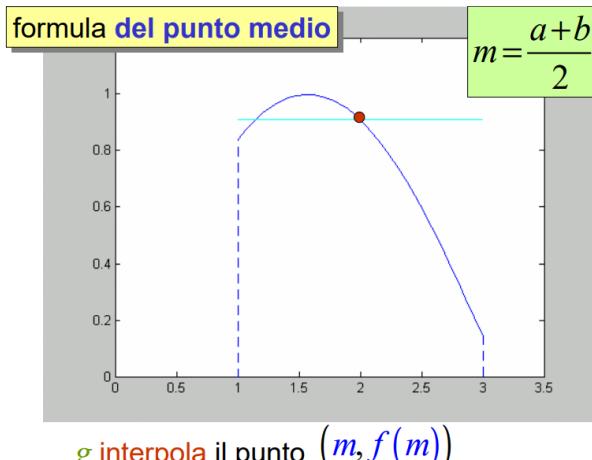
L'**errore** sarà rappresentato dalla zona in eccesso più la zona in **difetto**, e talvolta si compensano (a volte più, a volte meno).

Da notare che talvolta la stima dell'integrale potrebbe differire di molto rispetto all'integrale vero e proprio in base ai valori di a o di b (nell'esempio prima, se avessimo preso b il rettangolo sarebbe stato molto piccolo). Per tal motivo potremmo usare il *punto medio*.

FORMULA DEL PUNTO MEDIO

Prendiamo in considerazione un solo punto che sarà dato dal **punto medio m** dell'intervallo. Quindi prendiamo in considerazione il punto $(x_m, f(x_m))$, per il quale passerà il polinomio interpolante $g(x) = p_0(x)$.

L'integrale di tale nodo interpolante ha una maggior possibilità di esser simile all'integrale della funzione di partenza.

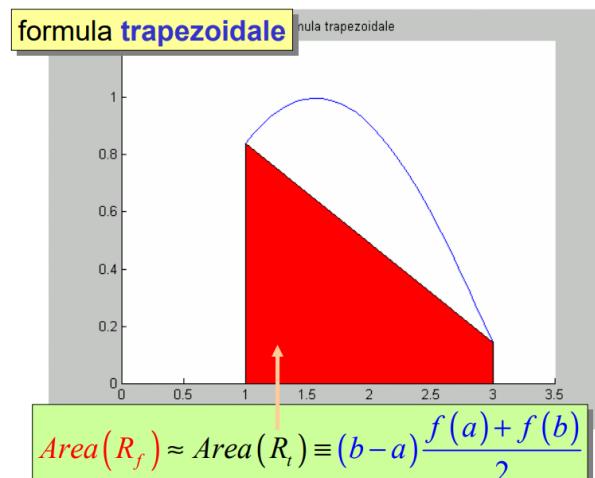
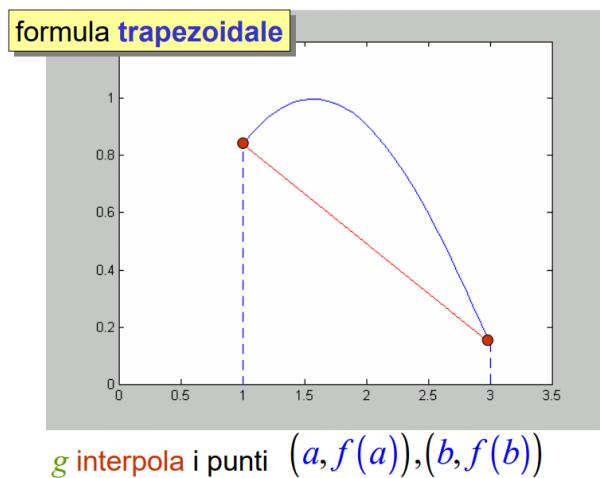


FORMULA TRAPEZOIDALE

Prendiamo in considerazione i due punti $(a, f(a))$ e $(b, f(b))$, per i quali passerà il polinomio interpolante $g(x) = p_1(x)$. Quindi avremo che $p_1(a) = f(a)$ e $p_1(b) = f(b)$.

Si tratta di una **retta**, la quale area sottostante sarà un **trapezio**.

Ricordando che: $A(\text{trapezio}) = \frac{(B + b) * h}{2}$, dove h è il **segmento orizzontale**.



Base maggiore = $f(a)$, base minore = $f(b)$, $h = b-a$.

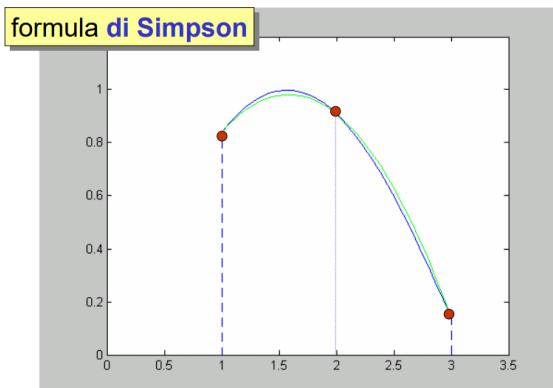
FORMULA DI SIMPSON

Prendiamo in considerazione tre punti equispaziati, ovvero i due estremi e il punto medio. Si tratta di una **parabola**. Per calcolarne l'**area** sottostante dobbiamo considerare che i 3 punti non hanno tutti lo stesso **peso**, infatti la maggior parte dell'informazione è contenuta nel **centro** (p. medio).

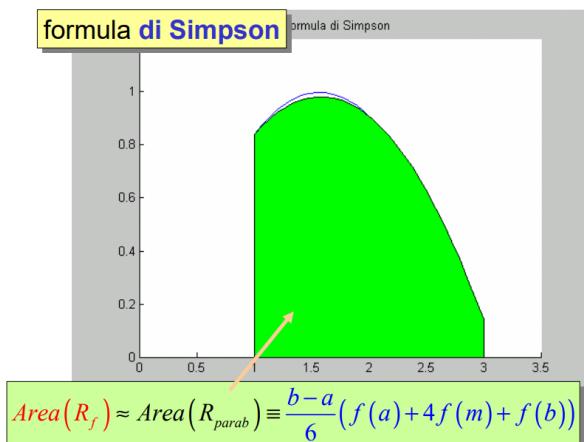
Per far ciò dobbiamo scegliere un **criterio**:

- interpoliamo e integriamo (quella che usiamo);
- facciamo in modo che l'errore sulle parabole sia nullo.

Il peso di **a** sarà 1/6, il peso di **b** sarà 1/6 e il peso di **m** sarà 4/6. Da cui otteremo la seguente formula dell'area.



g intercala i punti $(a, f(a)), (m, f(m)), (b, f(b))$



$$\text{Area}(R_f) \approx \text{Area}(R_{\text{parab}}) \equiv \frac{b-a}{6} (f(a) + 4f(m) + f(b))$$

Per capire il **perché di questi pesi**, considereremo la seguente situazione da cui otteniamo un sistema lineare. Ci basterà sottrarre la prima equazione con la terza ed otteniamo i coefficienti (il secondo non lo calcolo perché la funzione è dispari quindi scompare).

A questo punto mi basterà calcolare l'**integrale della parabola interpolante**.

$$g(x) = a_1 + a_2x + a_3x^2$$

$$g(-h) = f(-h) \equiv f(a)$$

$$g(0) = f(0) \equiv f(m)$$

$$g(h) = f(h) \equiv f(b)$$

$$a_1 - a_2h + a_3h^2 = f(a)$$

$$a_1 = f(m)$$

$$a_1 + a_2h + a_3h^2 = f(b)$$

$$a_1 = f(m)$$

$$a_2 = \dots$$

$$a_3 = \frac{1}{2h^2}(f(a) - 2f(m) + f(b))$$

integrale della parabola interpolante

$$\int_{-h}^h g(x) dx = \int_{-h}^h a_1 + a_2x + a_3x^2 dx$$

$$= a_1x + \frac{a_2}{2}x^2 + \frac{a_3}{3}x^3 \Big|_{-h}^h$$

$$= 2a_1h + \frac{2}{3}a_3h^3$$

$$= \frac{h}{3}(f(a) + 4f(m) + f(b))$$

poiché l'intervallo di integrazione ha ampiezza $2h$
in generale, per l'intervallo $[a,b]$ si ha:

$$= \frac{b-a}{6}(f(a) + 4f(m) + f(b))$$

formula di Simpson

LEZIONE 19/20 – Formule Composite e Andamento dell'Errore

Nelle formule composite guardiamo le figure come una unione di più pannelli diversi (divido il problema). Applicherò la formula di quadratura per ogni pannello. I pannelli non devono avere necessariamente tutti la stessa ampiezza.

L'idea è quella di suddividere l'intervallo di integrazione in **p** sottointervalli ($n=p+1$ estremi) e applicare la formula di base ad ogni sottointervallo.

Infatti avremo che l'integrale nell'intervallo di partenza sarà uguale alla somma degli integrali dei sottointervalli.

$$\int_a^b f(x) dx = \sum_{j=1}^p \int_{x_j}^{x_{j+1}} f(x) dx$$

FORMULA RETTANGOLARE COMPOSITA

Supponiamo di avere n punti equispaziati, e che $p = n-1$. L'ampiezza di un sottointervallo, che rappresenta anche l'altezza **h** del rettangolo, sarà $\frac{(b-a)}{p}$ (o in modo equivalente $\frac{(b-a)}{n-1}$).

La base **b** in un punto x_i sarà data dal valore della funzione in quel punto. Quindi avremo:

$$Q_r f = h \sum_{i=1}^{n-1} f(x_i)$$

Per calcolare x_i possiamo usare la formula generale “ $x_i = a + h(i-1)$ ” (oppure in MATLAB il comando *linspace*).

FORMULA DEL PUNTO MEDIO COMPOSITA

Supponiamo di avere n punti equispaziati, e che $p = n-1$. Ci ritroviamo nella stessa situazione della formula rettangolare composita.

$$Q_{pm} f = h \sum_{i=1}^{n-1} f(m_i)$$

In questo caso consideremo non gli estremi a, b ma i punti medi, i quali possono essere calcolati come: $m_i = \frac{x_i + x_{i+1}}{2}$. Quindi avremo:

FORMULA TRAPEZOIDALE COMPOSITA

Al contrario della Q_r e della Q_{pm} , questa formula **non è a pesi uguali**. Infatti i nodi interni li utilizziamo ben due volte (nel calcolo delle aree) mentre quelli esterni una sola volta.

I nodi esterni saranno a e b , mentre i nodi interni saranno i punti x_i per $i=2$ a $n-1$.

$$Q_tf = h\left(\frac{1}{2}f(a) + \sum_{i=1}^{n-1} f(x_i) + \frac{1}{2}f(b)\right)$$

FORMULA DI SIMPSON COMPOSITA

Suddividiamo l'intervallo $[a,b]$ in p pannelli di ugual ampiezza, e prendiamo in considerazione tre punti equispaziati, ovvero i due estremi e il punto medio, per ogni pannello.

Costruirò l'interpolante della funzione per ogni sottointervallo, e sostituirò l'integrale dell'interpolante a quello della funzione.

La formula da applicare, per ogni sottointervallo, sarà:

$$Q_sf = \frac{h}{6}(f(a) + 4\sum_{i=1}^p f(m_i) + 2\sum_{i=2}^{p-1} f(x_i) + f(b))$$

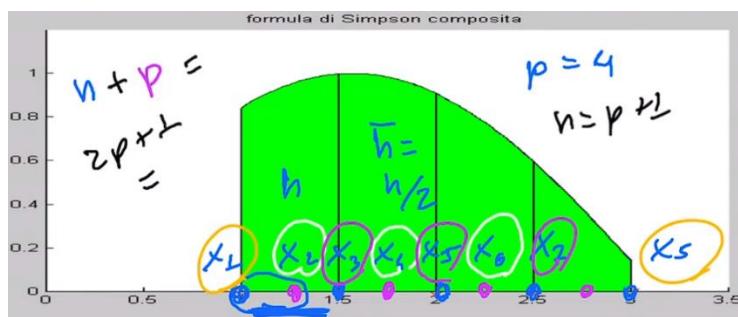
In questo caso i nodi interni li utilizziamo ben due volte (nel calcolo delle aree) mentre quelli esterni e i punti medi una sola volta.

I nodi esterni saranno a e b , mentre i nodi interni saranno i punti x_i .

nb: in questo caso consideriamo h e non $b-a$ perché è inutile calcolare l'ampiezza per ogni sottointervallo in quanto è sempre la stessa, cioè h .

La formula di Simpson composita su p sottointervalli usa $2p+1$ valori della funzione f . Questo perché dobbiamo aggiungere agli n punti anche i punti medi (che ne sono p), quindi ricordando che $n = p+1$, avremo un totale di $2p+1$ punti.

OSSERVAZIONI:



Identifichiamo ognuno dei $2p+1$ punti con una certa x_i . Noteremo che tutti i punti medi avranno indice pari mentre tutti gli altri nodi avranno indice dispari.

Osserviamo inoltre che l'**ampiezza** che ci interessa non è tutto h , ma la sua metà, ovvero l'intervallo che va da un certo nodo x_i al nodo x_{i+1} . Chiamiamo tale ampiezza \bar{h} *segnato*.
 $\bar{h} = h/2$.

In base alla loro posizione i nodi avranno un **peso** diverso:

I nodi con indice **pari** avranno peso 4, i nodi con indice **dispari** avranno peso 2, ad eccezione degli estremi (quindi il primo e l'ultimo nodo) che avranno peso 1.

Fatte tali osservazioni, possiamo **reinterpretare la formula**, utilizzando solo i nodi x , come:

$$Q_s f = \frac{\bar{h}}{3} \left(f(x_1) + 4 \sum_{i \text{ pari}}^{n-1} f(x_i) + 2 \sum_{i \text{ dispari}}^{n-2} f(x_i) + f(x_n) \right)$$

CALCOLO DELL'ERRORE PER LE F.Q. COMPOSITE

Quanto maggiore è p , ovvero quanti più pannelli sono presenti, minore sarà h , ovvero l'ampiezza dei sottointervalli. Più è piccolo h , minore sarà l'**errore** che grava sull'intervallo. Da ciò possiamo dire che l'**errore è una funzione che dipende da h** , e $E(h) \rightarrow 0$ quando $h \rightarrow 0$.

L'errore non dipende però solo da h , ma anche dalla *funzione integranda* e dalla *formula di quadratura* utilizzata. Tuttavia, a noi interessa l'**andamento generale dell'errore** per una qualsiasi **funzione che sia abbastanza regolare** e fissata la formula di quadrata.

Possiamo indicare tale andamento con un α .

Una **tipica** funzione dell'errore avrà una forma del tipo:

$$E(h) \cong \alpha \cdot h^{\text{esp}}$$

ove:

- **α è un numero** che dipende dalla funzione integranda e dalla formula di quadrata;
- **esp è un numero ≥ 1** (è importante purché diminuisca l'errore).

Le varie formule di quadratura hanno diversi **andamenti**, in particolare se *dimezziamo il passo* (ovvero raddoppiamo p) l'errore si riduce di una certa quantità, che prende il nome di **ordine di convergenza**:

- **rettangolare**: 2;
- **punto medio**: 4;
- **trapezoidale**: 4;
- **simpson**: 16;

È anche possibile considerare l'andamento dell'errore come una **curva**. Se ne osserviamo la pendenza noteremo che:

rettangolare: -1; **punto medio**: -2; **trapezoidale**: -2; **simpson**: -4;

Esempio: approssimiamo l'integrale di $f(x) = 1/x$ compreso in $[1,2]$.

APPROXIMAZIONI DELL'INTEGRALE				
n	Rettang	PMedio	Trapez	Simpson
3	0.8333333	0.6857143	0.7083333	0.6944444
5	0.7595238	0.6912199	0.6970238	0.6932540
9	0.7253719	0.6926606	0.6941219	0.6931545
17	0.7090162	0.6930252	0.6933912	0.6931477
33	0.7010207	0.6931167	0.6932082	0.6931472
65	0.6970687	0.6931396	0.6931624	0.6931472



ERRORE DI APPROXIMAZIONE				
n	Rettang	PMedio	Trapez	Simpson
3	0.1401862	0.0074329	0.0151862	0.0012973
5	0.0663766	0.0019273	0.0038766	0.0001068
9	0.0322247	0.0004866	0.0009747	0.0000074
17	0.0158690	0.0001220	0.0002440	0.0000005
33	0.0078735	0.0000305	0.0000610	0.0000000
65	0.0039215	0.0000076	0.0000153	0.0000000

valore esatto: 0.69314718055995

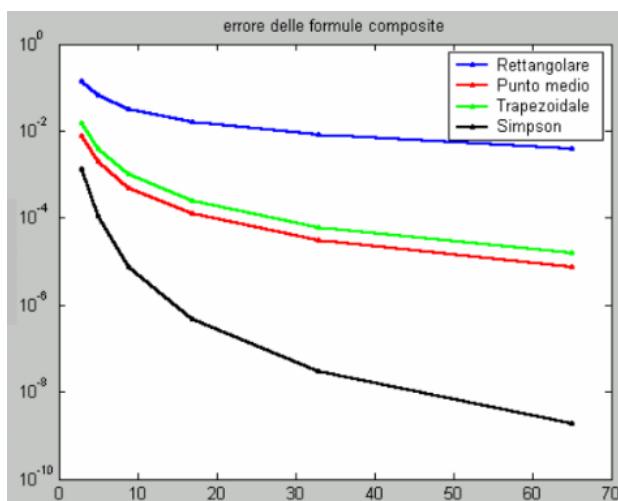
Possiamo notare come al raddoppiare dei nodi (tra l'altro, ricorda che $p=n-1$) l'errore diminuisce di un certo fattore..

Notiamo che al raddoppiare del numero di intervalli, h si dimezza, e l'errore diminuisce di un certo fattore: nel caso della rettangolare l'errore diminuisce di **1/2**, nel caso del punto medio e della trapezoidale l'errore diminuisce di **1/4**, mentre nel caso della Simpson l'errore diminuisce di **1/16**.

Infatti otteniamo l'**ordine di convergenza** facendo il **rapporto** fra l'errore al passo n e l'errore al passo $2n-1$:

RAPPORTO ERRORE(n)/ERRORE($2n-1$)				
n	Rett	PMedio	Trap	Simpson
3	2.1119806	3.8566575	3.9173604	12.1480653
5	2.0598079	3.9605103	3.9773767	14.5287510
9	2.0306652	3.9898464	3.9941940	15.5636784
17	2.0154907	3.9974429	3.9985385	15.8850124
33	2.0077804	3.9993595	3.9996340	15.9708438
	2	4	4	16

Curve dell'errore:



Abbiamo osservato come al raddoppiare del numero di nodi l'ampiezza dell'intervallo si dimezza. In particolare **osserveremo che**:

se $E_h = \alpha \cdot h$ allora si ha

$$\frac{E_h}{E_{h/2}} = \frac{\alpha h}{\alpha h/2} = 2$$

$$E_n \equiv E_h$$

errore di una formula composita su n nodi

$$E_{2n} \equiv E_{h/2}$$

errore di una formula composita su $2n$ nodi

se $E_h = \alpha \cdot h^2$ allora si ha

$$\frac{E_h}{E_{h/2}} = \frac{\alpha h^2}{\alpha (h/2)^2} = 4$$

$$\frac{E_n}{E_{2n}} = \frac{E_h}{E_{h/2}}$$

se $E_h = \alpha \cdot h^4$ allora si ha

$$\frac{E_h}{E_{h/2}} = \frac{\alpha h^4}{\alpha (h/2)^4} = 16$$

Questa è un'**osservazione sperimentale**, tuttavia esistono dei **teoremi** che descrivono l'errore di quadratura di una formula composita su n nodi in modo **generale**.

Rettangolare

$$|f'(x)| \leq K$$

$$|If - Q_{r,n} f| \leq \frac{K(b-a)}{2} h \approx \frac{K(b-a)^2}{2n}$$

Punto medio

$$|f''(x)| \leq K$$

$$|If - Q_{pm,n} f| \leq \frac{K(b-a)}{24} h^2 \approx \frac{K(b-a)^3}{24n^2}$$

Trapezoidale

$$|f''(x)| \leq K$$

$$|If - Q_{t,n} f| \leq \frac{K(b-a)}{12} h^2 \approx \frac{K(b-a)^3}{12n^2}$$

Simpson

$$|f^{IV}(x)| \leq K$$

$$|If - Q_{s,n} f| \leq \frac{K(b-a)}{180} h^4 \approx \frac{K(b-a)^5}{180n^4}$$

Questi teoremi hanno delle **conseguenze**:

- **rettangolare**: l'errore è **nullo** se f è costante;
- **punto medio**: l'errore è **nullo** se f è lineare;
- **trapezoidale**: l'errore è **nullo** se f è lineare;
- **simpson**: l'errore è **nullo** se f è un polinomio di grado al più 3.

ALTRE FORMULE DI QUADRATURA INTERPOLATORIA

Tali formule sono basate sull'idea di integrare una funzione che **interpola la funzione integranda** su n nodi dell'intervallo di integrazione.

$$\int_a^b f(x) dx \approx \int_a^b s(x) dx$$

Possiamo distinguere: la **spline cubica interpolante** (nodi equispaziati), la **cubica di Hermite interpolante** (nodi equispaziati), e il **polinomio di grado $n-1$ interpolante su nodi di Chebychev**.

SPLINE CUBICA INTERPOLANTE

Possiamo **calcolare l'integrale di una spline cubica interpolante** come:

$$\int_a^b s(x) dx = \sum_{i=1}^{n-1} \int_{x_i}^{x_{i+1}} s(x_i) dx$$

Ricordando come è fatto $s_i(x)$ (vedi *spline cubica interpolante*) potremo riscrivere tale integrale come:

$$\int_{x_i}^{x_{i+1}} (a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i) dx$$

Per le proprietà degli integrali, possiamo risolvere tale integrale “a pezzi”.

Si tratterà di trovare integrali di polinomi molto semplici del tipo $f = a_i(x-7)^3$, la quale primitiva sarà $F = a_i(x-7)^4 / 4$ (ricorda: *integrale del tipo $x^{\text{esp}} = x^{\text{esp}+1}/\text{esp}+1$*)

Fatta quest'osservazione, l'integrale di cui sopra verrà risolto come:

$$\frac{1}{4} a_i (x_{i+1} - x_i)^4 + \frac{1}{3} b_i (x_{i+1} - x_i)^3 + \frac{1}{2} c_i (x_{i+1} - x_i)^2 + d_i (x_{i+1} - x_i)$$

Notare come $x_{i+1} - x_i$ non sia altro che h , e quindi possiamo riscrivere il tutto come:

$$\frac{h^4}{4} a_i + \frac{h^3}{3} b_i + \frac{h^2}{2} c_i + h d_i$$

(IMPLEMENTAZIONE FUNZIONE SUL WORD DI MATLAB)

CUBICA DI HERMITE INTERPOLANTE

La **cubica di Hermite** è costruita in modo diverso rispetto alla spline, ma si tratta sempre di un'interpolante. Ciò significa solo che i vincoli utilizzati sono diversi sull'interpolante, ma vale lo stesso ragionamento (e le stesse formule) viste fino ad ora per la spline.

$$\int_a^b f(x) dx \approx \int_a^b s(x) dx = \sum_{i=1}^{n-1} \int_{x_i}^{x_{i+1}} s(x_i) dx$$

(anche l'implementazione è la stessa, cambia solo che invece di usare *spline* per il calcolo dell'interpolante, useremo *pchip*).

FORMULE DI QUADRATURA (DI TIPO) MONTECARLO

Si tratta di una famiglia di formule che fanno riferimento al casinò di Montecarlo in quanto “tentano la fortuna”.

Tuttavia esiste un teorema (**diseguaglianza di Chebychev**) che quantifica quanto sia improbabile essere “sfortunati” ripetendo un certo esperimento. Infatti afferma che se faccio molti esperimenti dello stesso tipo, il **valore misurato** si avvicina in media al **valore atteso** con una probabilità che si può misurare.

(*si pensi al lanciare una moneta. Se la lancio un milione di volte, mi aspetterò che circa la metà delle volte sia testa e l'altra metà croce*).

Si trattano di algoritmi **stocastici**, infatti sono algoritmi di quadratura basati sulla generazioni di **ascisse casuali** (distribuite uniformemente) nell'intervallo di integrazione.

L'**idea**, per il calcolo dell'integrale, è quella di trovare l'**altezza intermedia** della funzione e moltiplicarla per la **base** (area del rettangolo). L'area così calcolata avrà delle parti in eccesso e delle parti in difetto che andranno più o meno a compensarsi.

Ciò è noto anche come *Teorema della media integrale*.

Tale teorema afferma che, più o meno, l'integrale è la **f media della base**:

$$If = \int_a^b f(x) dx = (b - a) \cdot f_{medio}^{[a,b]}$$

Quindi dovremo **stimare** f_{medio} (in a, b) mediante la **media aritmetica** di n valori $f(x_i)$, con x_i **ascisse casuale** (distribuite uniformemente) nell'intervallo di integrazione.

Ovvero dovrò valutare la f in tanti punti a caso e ne farò la media:

$$Q_{MC}f = (b - a) \frac{1}{n} \sum_{i=1}^n f(x_i^{rand})$$

nb: x_i^{rand} sono numeri pseudocasuali uniformemente distribuiti in $[a, b]$.

Il valore atteso della funzione, calcolato in tanti punti, **converge** alle media integrale.

PROPRIETÀ FORMULE QMC

Se la funzione integranda è continua, all'aumentare del numero di punti mi avvicinerò sempre di più al valore atteso, ovvero le formule QMC sono **sempre convergenti**, tuttavia l'errore diventa piccolo *lentamente*, ovvero tali formule sono a **bassa velocità di convergenza**.

Possiamo calcolare l'**andamento dell'errore** come:

$$|If - Qf| = \frac{\alpha(b-a)}{\sqrt{n}}$$

Si tratta di una funzione che è proporzionale a $n^{-1/2}$ (cioè \sqrt{n}), cioè quando n aumenta l'errore tende a 0 (*lentamente*).

Infatti per ridurre l'errore di un fattore 10 è necessario moltiplicare n per un fattore 100)

(es. $E(400) = 0.02$ \rightarrow $E(40.000) = 0.002$. Per guadagnare una sola cifra significativa abbiamo dovuto centuplicare il numero di punti).

LEZIONE 21 –Statistica descrittiva

INTRODUZIONE

La statistica è una disciplina che ha come fine lo studio quantitativo e qualitativo di un particolare fenomeno collettivo in condizioni di incertezza o non determinismo.

Ne distinguiamo **due rami**:

- **descrittiva**: si ha una popolazione e le unità statistiche. Si fissano uno o più caratteri (modalità,...). Avremo inoltre indici, tabelle e grafici che raccontano la descrizione dei dati per un dato carattere su tutta la popolazione.
- **inferenziale**: si ha un campione il quale viene estratto da una popolazione, e sul quale farò l'indagine statistica. Anche in questo caso fissero uno o più caratteri ed avremo indici, tabelle e grafici. Dal campione, avrà lo scopo di ottenere informazioni generali che valgano per la popolazione.

In statistica per **popolazione** si intende l'insieme degli elementi che sono oggetto di studio, ovvero l'insieme delle unità (dette **unità statistiche**) sulle quali viene effettuata la rilevazione.

Gli elementi che descrivono una unità statistica prendono il nome di **carattere**.

I possibili *valori* prendono il nome di **modalità**.

Si parla di **statistica descrittiva** se i dati rimangono circoscritti alla sola popolazione, mentre si parla di **statica inferenziale** se questi dati vengono portati fuori dalla popolazione di partenza, ovvero se si passa dal *particolare* al *generale*.

Nel secondo caso la popolazione di partenza prenderà il nome di **campione**, in quanto farà parte della popolazione (più vasta).

STATISTICA DESCRIPTIVA

La statistica descrittiva analizza quantitativamente la proprietà di un insieme di **dati**, e l'insieme dei dati è detto **popolazione** o **campione** (anche se è preferibile usarlo solo per la S. inferenziale). Dal campione ne deduciamo le proprietà della **popolazione**.

Per la generazione di **dati casuali in MATLAB** si usa il comando **randn(m,n)**; il quale creerà un insieme di dati la cui distribuzione **non è uniforme**, infatti ci aspetteremo che i dati siano maggiormente distribuiti al centro.

Si parla di *distribuzione normale* (o *Gaussiana*).

es:

```
>> dati = 5+randn(1,40);
```

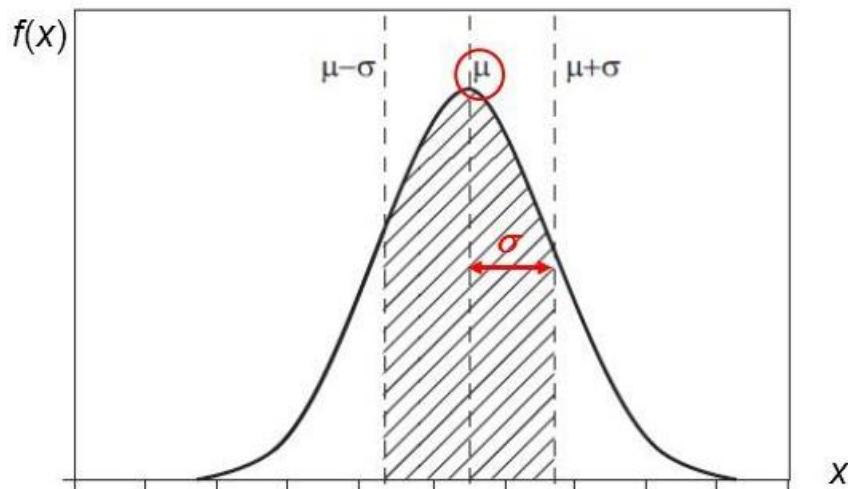
La formula della distribuzione Gaussiana è:

$$f(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Tale formula ci dice qual è la probabilità che, scelto un valore secondo tale distribuzione Gaussiana, questo valore sia proprio x . Rappresenta la **densità di probabilità**. La probabilità di x è $f(x)$.

Si tratta di una funzione simmetrica rispetto al punto μ , e sarà più o meno schiacciata in base al valore di σ .

Se i dati sono molto variabili la campana sarà più schiacciata mentre se sono poco variabili la campana sarà più allungata.



Mi immagino che la maggior parte dei valori sia vicino al punto μ . Inoltre valori che si discostano molto da μ sono ritenuti molto improbabili.

esempio statura delle persone

La maggior parte delle persone ha una statura intorno ai 170 cm ($\mu = 170$), la quale varia di più o meno 10 cm ($\sigma = 10$). Quindi la maggior parte delle persone avrà un'altezza fra 160cm e 180cm. Più ci allontaniamo da questi valori, meno persone incontreremo.

In MATLAB di default $\sigma = 1$, $\mu = 0$. Se volessivo assegnare valori diversi dovremo seguire una sintassi del tipo: $dati = \sigma + \mu * randn(m,n)$.

I dati, se conservati all'interno di un contenitore (es. array) prendono il nome di **dati grezzi**. D_i ove D_i rappresenta l'i-simo dato. Il numero di dati prende il nome di **numerosità n**. Il carattere viene rappresentato con **x**, mentre le sue modalità vengono identificate con x₁, x₂ ... x_m (ove m è il numero di modalità).

Distinguiamo due casi:

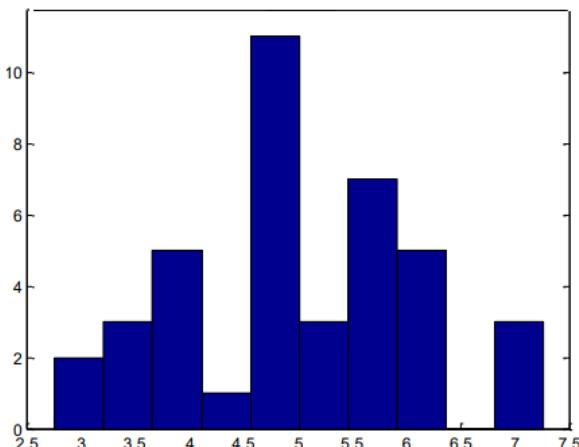
- **variabile statistica**: le modalità sono numeri, su cui possiamo fare delle operazioni numeriche (es. statura);
- **mutabile statistica**: le modalità NON sono numeri, e dobbiamo raccogliere informazioni con strumenti diversi, come la frequenza (es. colore occhi).

Classi di modalità (bins)

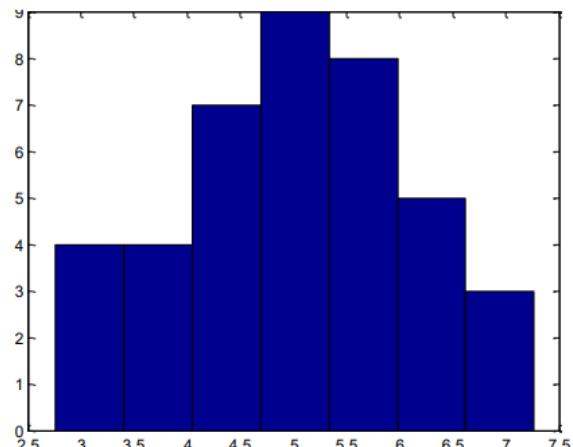
Una classe di modalità (bin) è un sottointervallo nel quale sono contenuti una partizione dei dati. Il campo di variazione viene quindi suddiviso in m sottointervallo di uguale ampiezza.

Si conta il numero di dati del campione in ogni bin, e si visualizza la barra verticale per ogni bin, che ne indica il numero di dati.

Più classi creiamo, più basse saranno le tendenze. Viceversa, meno classi creiamo, più alte saranno le tendenze.



10 bins



7 bins

frequenza relativa:

$$\text{frequenze relative} = \frac{\text{frequenze assolute}}{\text{numero di dati}}$$

La somma delle frequenze relative è sempre 1.