CALCOLO NUMERICO

Guida all'orale

Author
Marco Omicini (TüT)
Università degli studi del Molise
Luglio 2023

Contents

1 1-cfu

Theorem 1.1. *This is a theorem.*

Proposition 1.2. This is a proposition.

Principle 1.3. This is a principle.

1.1 pictures

1.2 citation

this is a citation[**?**].

1.3 problemi ben posti (hadamard)

un problema matematico si può esprimere nella forma

$$f(x, d) = 0 \tag{1}$$

dove x e d possono essere qualsiasi cosa, come matrici, vettori, numeri reali o anche funzioni. in questa sede, stiamo considerando un problema matematico astratto, non legato a situazioni specifiche.
il problema $f(x, d) = 0$ è **ben posto** se e solo se esiste una sola soluzione x, e tale soluzione dipende in modo continuo dai dati.
se almeno una delle tre condizioni non è verificata, il problema sarà **mal posto**, e non avrebbe senso affrontarlo da un punto di vista numerico e richiederebbe una sua riformulazione.

1.4 problemi ben condizionati

un problema si dice **ben condizionato** se piccole perturbazioni sui dati producono piccole perturbazioni sui risultati. consideriamo il problema perturbato $f(x + \delta x, d + \delta d)$. allora deve valere quanto segue:

$$\exists k_0 = k(d) \text{ tale che } \forall \delta d : d + \delta d \in d, \|\delta x\| \leqslant k_0 \|\delta d\|. \tag{2}$$

dove $\|\cdot\|$ rappresenta la norma euclidea o una qualsiasi altra norma matematica appropriata. in altre parole, se la perturbazione sui dati è proporzionale alla perturbazione sui risultati, allora il problema è ben condizionato.
al contrario, un problema **mal condizionato** è caratterizzato da un'elevata sensibilità alle perturbazioni sui dati, il che lo rende più difficile da risolvere correttamente.

nota

k_0 è una costante detta indice di condizionamento
d è l'insieme dei dati ammissibili.
$\forall \delta d : \delta d + d \in d$ ci assicura che il dato perturbato $\delta d + d$ sia ancora un dato ammissibile per il problema

1.5 indice di condizionamento

definiamo, per il problema matematico (1), l'indice di condizionamento, ossia il valore che ci indica quanto sono sensibili i risultati rispetto a una perturbazione sui dati.

$$k_{rel}(d) = \sup \left\{ \frac{\|\delta x\|/\|x\|}{\|\delta d\|/\|d\|}, \delta d \neq 0, d + \delta d \in d \right\}. \tag{3}$$

nel caso in cui $d = 0$ o $x = 0$ ha più senso introdurre *l'indice di condizionamento assoluto*:

$$k_{abs}(d) = \sup \left\{ \frac{\|\delta x\|}{\|\delta d\|}, \delta d \neq 0, d + \delta d \in d \right\}. \tag{4}$$

come possiamo immaginare ormai, un k_0 grande indicherà che il problema è molto sensibile a perturbazioni sui dati, di conseguenza più k_0 sarà "grande" più il problema sarà mal condizionato

la proprietà di condizionamento del problema prescinde il metodo numerico (aka l'algoritmo) che andremo poi ad utilizzare per risolverlo.
difatti come già preannunciato prima, un problema mal posto, ossia un problema tale per cui esistono più soluzioni (o non ne esiste nessuna)
oppure un problema tale per cui i risultati non dipendono in maniera continua dai dati (mal condizionato) non merita di essere proprio affrontato, perché ha dei disagi profondi che l'algoritmo non sarà in grado di risolvere.
un problema di questo tipo, quindi, necessiterà una riformulazione tale per cui venga rispettata la proprietà di ben posizione.

nota

potrebbe creare un po di confusione, almeno per me è così, il fatto che la definizione di mal condizionamento è un pò ambigua.
infatti il termine "grande" lascia un po a desiderare visto che "grande" è una grandezza non definita.
per chiarire diremo che al crescere di k_0 crescerà il mal condizionamento del problema.
se $k_0 = \infty$ allora siamo sicuri che il problema è mal condizionato (e quindi anche mal posto visto che non vale piu la dipendenza continua dei risultati dai dati)

1.6 stabilità di un algoritmo

supponendo che il probelma sia ben posto allora un metodo numerico consisterà, in generale, nel costruire una serie di probelmi approssimanti

$$f_n(x_n, d_n) = 0, \quad n \geqslant 1 \tag{5}$$

con la speranza che per $n \to \infty$ si verifichi che $x_n \to x$ ossia che la soluzione approssimante tenda alla soluzione effettiva.
per essere più precisi vorremmo che si verifichi quanto segue:

$$f_n(x_n, d_n) - f(x, d) \to 0 \text{ per } n \to \infty \tag{6}$$

ossia che la successione di problemi approssimanti converga al problema effettivo.

definizione: un metodo numerico si dirà **stabile** se

1. l'errore si mantiene limitato e non cresce all'aumentare del numero delle operazioni
2. ogni sotto problema $f_n(x_n, d_n)$ è ben posto

un algoritmo che non rispetta anche una sola di queste proprietà si dirà **instabile**

1.7 rappresentazione floating point

se vi chiedessi di eseguire questa motliplicazione, quanto tempo ci mettereste?

$$0,00000000034 \times 1343400000000 \tag{7}$$

sarebbe una spina nel fianco da fare come operazione vero?
un modo per semplificarci di molto la vita è togliere quella virgola, e segnarci in qualche modo la sua posizone.
possiamo quindi rappresentare il numero $0,00000000034$ come $34*10^{-12}$ e il numero 1343400000000 come $13434 * 10^8$
ora l'operazione che prima ci sembrava complicata da fare è decisamente piu semplice. infatti basta fare $34 * 13434 = 456756$ e aggiungere i due esponenti $-12 + 8 = -4$ ottenendo in definitiva

$$456757 * 10^{-4} = 4,56756$$

questa viene detta *notazione scientifica* e porta con se un concetto importante che abbiamo appena visto:

Proposition 1.4. la posizione della virgola la decide l'esponente. questo ci permette di far fluttuare la virgola dove ci pare *(floating point)*

più formalmente diremo:

un numero reale x può essere rappresentato come

$$m \times b^{esp} \qquad m \in \diagdown,\ esp \in \ltimes, \tag{8}$$

m viene detta **mantissa**, b viene detta **base** mentre esp viene detto **esponente** o **caratteristica**
quindi fissata una base, ogni numero reale x può essere rappresentato tramite la coppia (m, esp)
il numero 0 per convenzione viene rappresentato tramite la coppia $(0, 0)$
ciò che abbiamo visto poco fà può essere applicato anche nei calcolatori, difatti basta solo ragionare in termini di 0 e 1

ottimo! facciamo un esempio e rappresentiamo il numero 12,3.
la prima cosa da fare è convertirlo in binario. per fare ciò dobbiamo spezzare il numero, quello che c'è prima della virgola e quello che c'è dopo. risolviamo quindi il 12 e ricaviamo

la sua rappresentazione binaria. per farlo usiamo la nota tecnica della divisione (dividere sistematicamente il numero fino a reggiungere lo zero, memorizzando il resto per ogni divisione):

$$\begin{aligned} 12/2 &= 6 \quad (0) \\ 6/2 &= 3 \quad (0) \\ 3/2 &= 1 \quad (1) \\ 1/2 &= 0 \quad (1) \end{aligned} \tag{9}$$

ora prendiamo il numero *dall'alto verso il basso* ottenendo così che $12_{10} = 1100_2$
occupiamoci adesso dell'altro pezzo, ossia $0,3$.
per convertire questo numero in binario possiamo usare la tecnica della motliplicazione (moltiplicare sistematicamente il numero memorizzando ogni volta se abbiamo raggiunto o meno un intero):

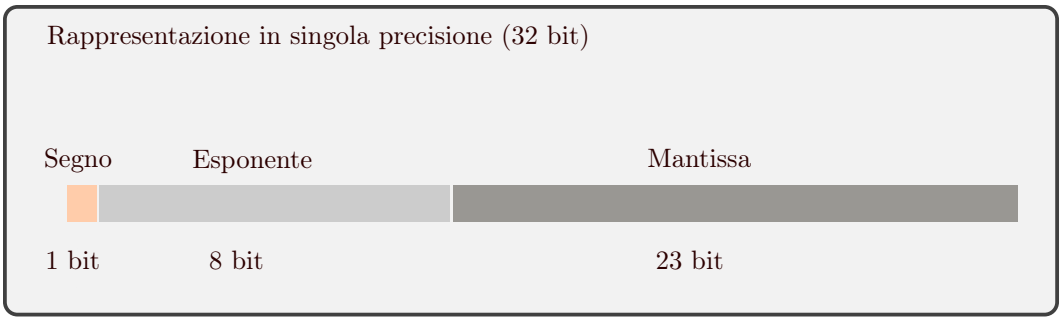
$$\left.\begin{aligned} 0,3 \times 2 &= 0,6 \quad (0) \\ 0,6 \times 2 &= 1,2 \quad (1) \\ 0,2 \times 2 &= 0,4 \quad (0) \\ 0,4 \times 2 &= 0,8 \quad (0) \\ 0,8 \times 2 &= 1,6 \quad (1) \end{aligned}\right\} \text{pattern} \qquad \begin{aligned} 0,6 \times 2 &= \cdots \end{aligned} \tag{10}$$

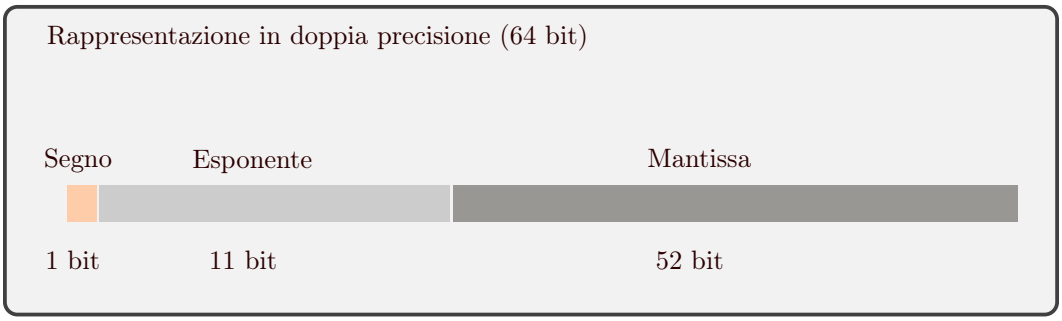
notiamo immediatamente il pattern ricorsivo, quindi non ha senso continuare a fare operazioni.
benissimo, abbiamo i due pezzetti che ci interessano $12,3 = 1100,0100110011001\ldots$ ora usiamo la notazione scientifica per indicare la posizione della virgola. allora sappiamo che dovrà fare tre salti a sinistra per sparire, quindi l'esponente è pari a 4.
possiamo quindi rappresentare questo numero nella forma $m * b^{esp}$ (normalizzazione)

$$m = 11000100110011001100\ldots, \qquad esp = 100, \qquad b = 2$$

bello vero?
facciamo adesso un passo avanti e contestualizziamo questo ragionamento.
la rappresentazione binaria ci servirà in scenari reali, e quindi la giocata che abbiamo appena fatto (concedere alla mantissa un numero infinito di cifre) non è stata molto onesta.
prendiamo dunque il nostro pennello e ritocchiamo quanto abbiamo appena fatto.

prima di tutto dobbiamo trovare un modo per regalare un bit alla mantissa così da poterne indicare facilmente il segno. quindi d'ora in poi il bit piu singificativo (quello più a sinistra) della mantissa indicherà il segno (0 = positivo, 1 = negativo)
per quanto rigurada l'esponente possiamo fare un gioco diverso. invece di regalare un bit aggiuntivo anche a lui, possiamo usare una tecnica chiamata *bias dell'esponente*.
questa tecnica consiste nel sommare all'esponente 127 e rappresentare quel valore. la conversione avverrà allo stesso modo, ossia sottraendo 127 all'esponente
per quanto riguarda lo spazio da dedicare a mantissa ed esponente possiamo seguire i seguenti due approcci.





consideriamo che, nell'esempio del numero $2,3$ se decidessimo di dare alla mantissa 5 bit otterremmo un'appossimazione pari a $2,25$
questo ci fa capire che più cifre diamo meglio è!
basta infatti salire a 8 e otteniamo un'approssimazione di $2,29$ circa.
ovviamente, non potremmo mai ottenere $2,3$, perche per farlo dovremmo rappresentare tutte le infinite cifre della mantissa, ed è proprio qui che si nascondono i disagi della rappresentazione numerica.

1.8 insieme dei numeri macchina

Bene prociediamo a definire bene questo insime di numeri.
Sia $f(b,s,l,u)$ l'insieme dei numeri macchina, dove b è la nostra base, s rappresenta la grandezza consentita per la mantissa, mentre l e u sono rispettivamente il range inferiore e superiore dell'esponente tale per cui $l \leqslant esp \leqslant u$

se un numero dovesse essere troppo piccolo allora sappiamo che l'esponente, negativo, è più piccolo di quanto il nostro caclolatore può offrire, ossia è *minore* di l di conseguenza o viene lanciato un errore oppure si approssima quel valore a zero.
questa situazione viene detta ***underflow***

se invece un numero dovesse avere, nella sua rappresentazione $x = m \times b^{esp}$ un esponente *maggiore* di u allora avremmo a che fare con la situazione opposta, ossa il numero è troppo grande per essere memorizzato, e di conseguenza si fa un'approssimazione dello stesso.
questo scenario viene detto ***overflow***

Proposition 1.5. l'insieme dei numeri macchina non consiste di punti equispaziati.

questo perché ci sono più numeri binari possibili con un bit significativo rispetto a quelli che ci sono con due bit significativi e man mano che ci allontaniamo dallo 0 il numero di bit usabili diminuisce, facendo diminuire anche il numero di numeri rappresentabili.
difatti l'insieme dei numeri macchina è piu denso vicino lo zero e meno denso man mano che ci si allontana

1.9 operazioni di arrotondamento

se un numero x rappresentato nella forma $m \times b^{esp}$ con $l \leqslant esp \leqslant u$ dovesse avere una mantissa con un numero di cifre superiorio a quello consentito, quindi $m > s$ allora si dovrà effettuare un arrotondamento per quel numero x
per fare questo si aggiunge $\frac{1}{2}b^{-s}$ e si tronca all's-esima cifra.
intuitivamente significa che si aggiunge 0.5 in ultima posizione e poi si tronca, questo non è altro che il noto arrotondamento per eccesso o per difetto.

1.10 errore di arrotondamento

per quanto riguarda l'errore che l'arrotondamento porta con se, il suo calcolo è molto semplice e faremo distinzione tra due tipi di errore: l'errore relativo e quello assoluto
dato $x = m \times b^{esp}$ una soluzione e $\bar{x} = \bar{m} \times b^{esp}$ una sua approssimazione, possiamo quindi definire

errore assoluto

questo errore è la differenza tra la soluzione effettiva e quella approssimata e rappresenta l'intervallo entro il quale il valore vero della grandezza si trova.

$$|x - \bar{x}| \tag{11}$$

errore relativo

l'errore assoluto ci da un'indicazione dell'intervallo entro in quale ci aspettiamo ragionevolmente che si trovi il valore vero della grandezza osservata però non descrive un quadro completo della situazione.
infatti mentre un errore di un centimetro su una distanza di un chilometro indicherebbe una misura insolitamente precisa, un errore di un centimetro su una distanza di quattro centimetri indicherebbe una valutazione piuttosto rozza.
e' perciò importante definire l'errore relativo

$$\frac{|x - \bar{x}|}{|x|} \tag{12}$$

1.11 epsilon macchina

la costante $\epsilon = \frac{1}{2}b^{1-s}$ rappresenta il più piccolo numero tale per cui vale $1 + \epsilon > 1$ ossia il più piccolo valore che un calcolatore può elaborare.

```c
int main() {
    epsilon = 1;
    while ((1 + epsilon) > 1) {
        epsilon = epsilon / 2;
    }

    printf("epsilon macchina: %e\n",
           epsilon);
    return 0;
}
```

1.12 errori di incolonnamento

prendiamo in esempio la seguente sottrazione $0.1234 * 10^3 \ominus 0.7231 * 10^0$ in un calcolatore che opera con mantissa $m = 4$
ciò che causerà l'errore è il fatto che dobbiao mettere in colonna i due numeri per effettuare la sottrazione, ma avendo una mantissa pari a 4 invece di fare questa sottrazione

$$\begin{array}{l} 0.1234000 * 10^3 \\ \underline{0.0007231 * 10^3} \\ 0.1226769 * 10^3 \end{array} \tag{13}$$

avremo una cosa del genere

$$\begin{array}{l} 0.1234 * 10^3 \\ \underline{0.0007 * 10^3} \\ 0.1227 * 10^3 \end{array} \tag{14}$$

l'errore nel risultato non è stato causato da una rappresentazione approssimata dei due addendi, ma dalla necessità di incolonnarli per poter procedere all'operazione

1.13 errori di cancellazione

Dati i seguenti due numeri reali:

$$p_1 = 0.12345789 \quad \text{e} \quad p_2 = 0.12345666$$

la cui differenza vale

$$d = p_1 - p_2 = 0.00000123 = 1.23 \times 10^{-6}$$

supponiamo che l'architettura del calcolatore ci permetta di memorizzare solo le prime 6 cifre dopo la virgola, quindi i due numeri, per essere rappresentati all'interno del calcolatore, verrebbero troncati (supponiamo per semplicità mediante chopping) appena dopo la sesta cifra:

$$p_1^* = 0.123457 \quad \text{e} \quad p_2^* = 0.123456$$

effettuando la sottrazione e riscrivendo il risultato nella notazione floating point, si ha invece:

$$d^* = p_1^* - p_2^* = 0.000001 = 1 \times 10^{-6}$$

1.14 cancellazione catastrofica

La cancellazione catastrofica (o perdita di cifre significative) è un problema comune nel calcolo numerico e si verifica quando un'operazione aritmetica produce risultati che sono meno precisi di quanto ci si aspetterebbe a causa di limitazioni nella rappresentazione dei numeri in un sistema di calcolo a virgola mobile.
Più formalemente possiamo descriverla in termini di errori sui dati e errori sui risultati:

Dati s e b due numeri.
Siano $fl(s)$ e $fl(b)$ le loro rappresentazioni, quindi due loro approssimazioni, usando la **??** calcoliamo

Errore relativo sui dati: $\delta_s = \left|\frac{s-fl(s)}{s}\right|$ e $\delta_b = \left|\frac{b-fl(b)}{b}\right|$ con $\delta_s, \delta_b \leqslant \varepsilon$

Errore relativo sui risultati: $\delta_{ris} = \left|\frac{(s-fl(s))-(b-fl(b))}{b-s}\right|$

Sarà possibile quindi definire il condizioamento del problema seguendo la **??**

$$\delta_{ris} = \frac{|fl(b) - fl(s) - (b-s)|}{|b-s|} = \frac{|\ \overbrace{(fl(b)-b)-(fl(s)-s)}^{\text{Ottenuta tramite un aggiustamento dei termini}}\ |}{|b-s|} \leqslant \frac{\overbrace{|b|\cdot\delta_b + |s|\cdot\delta_s}^{\substack{\text{visto che } fl(b)-b=\delta_b\cdot|b| \\ \text{e } fl(s)-s=\delta_s\cdot|b| \\ \text{possiamo sostituire. Rendiamo tutto minore uguale perchè facciamo diventare la sottrazione un'addizione}}}}{|b-s|} \leqslant \frac{|b|+|s|}{|b-s|}\cdot \overbrace{\varepsilon}^{\substack{\text{Maggioriamo } \delta_b \text{ e } \delta_s \\ \text{con } \varepsilon \text{ e raccogliamo}}}$$

Avremo quindi che il condizionamento del problema (l'amplificazione dell'errore) sarà dato da

$$\frac{|b|+|s|}{|b-s|} \tag{15}$$

2 Risoluzione numerica di sistemi di equazioni lineari

Visto che ci siamo accorti che operare con il calcolatore può rivelarsi pericoloso in questa sezione ci occuperemo di trovare i migliori approcci per risolvere in maniera numerica sistemi di equazioni lineari, ossia sistemi della forma

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n \end{cases} \quad \text{con A} \in \mathbb{R}^{n\times n}, \quad b \in \mathbb{R}^n$$

Un sistema lineare potrà quindi essere rappresentato nella forma $Ax = b$ ossia nella sua forma matriciale.
Per risolvere quindi questi sistemi possiamo procedere in due modi, usando:

- **Metodi diretti**: Forniscono la soluzione "esatta" (a meno di errori di round-off) in un numero finito di passi
- **Metodi iterativi**: La soluzione è ottenuta come limite di una successione di operazioni. Quindi i metodi, in un numero finito di passi, forniscono una soluzione approssimata

2.1 Ben posizione del problema

Il problema $Ax = b$ si dirà ben posto se la matrice dei coeficenti A è **non singolare**, ossia se il determinante di A è diverso da 0

2.2 Condizionamento del problema

Ricordandoci la **??** possiamo parlare di condizionamento e indice di condizionamento solo in riferimento a delle perturbazioni.
Sia quindi $(A + \delta A)(x + \delta x) = (b + \delta b)$ il nostro problema perturbato.
Il numero di condizionamento (ossia lo scalare che ci indica la "sensibilità" della soluzione rispetto a una perturbazione sui dati) sarà dato da:

Condizionamento del problema con perturbazioni su A e b

$$\frac{\|\delta x\|}{\|x\|} \leqslant \frac{K(A)}{1 - K(A)\frac{\|\delta A\|}{\|A\|}} \cdot \frac{\|\delta b\|}{\|b\|} \tag{16}$$

Possiamo anche considerare il problema perturbato solo sul vettore dei termini noti b

Condizionamento del problema con perturbazioni solo su b

$$\frac{\|\delta x\|}{\|x\|} \leqslant K(A)\frac{\|\delta b\|}{\|b\|} \tag{17}$$

Proof. Prendiamo il problema condizionato $A(x+\delta_x) = b+\delta_b$ allora deve valere $Ax+A\delta_x = b + \delta_b$ e consideriamo solo la sezione $A\delta_x = \delta_b$. Moltiplichiamo tutto per A^{-1} ottenendo

$\delta_x = A^{-1}\delta_b$. Applichiamo la funzione norma su ambe due i membri $\|\delta_x\| = \left\|A^{-1}\delta_b\right\|$ e usiamo la disuguaglianza triangolare delle norme ottenendo $\|\delta_x\| \leqslant \left\|A^{-1}\right\| \cdot \|\delta_b\|$.
Ora dividiamo entrambi i membri per $\|x\|$ e otteniamo

$$\frac{\|\delta_x\|}{\|x\|} \leqslant \frac{\left\|A^{-1}\right\| \cdot \|\delta_b\|}{\|x\|}$$

ricordandoci che $\|x\| = \|b\|/\|A\|$ possiamo scrivere

$$\frac{\|\delta_x\|}{\|x\|} \leqslant \frac{\left\|A^{-1}\right\| \cdot \|\delta_b\|}{\|b\|/\|A\|}$$

e quindi avremo

$$\frac{\|\delta_x\|}{\|x\|} \leqslant \left\|A^{-1}\right\| \cdot \|A\| \frac{\|\delta_b\|}{\|b\|}$$

ossia

$$\frac{\|\delta_x\|}{\|x\|} \leqslant K(d) \frac{\|\delta_b\|}{\|b\|}$$

con $K(d) = \left\|A^{-1}\right\| \cdot \|A\|$ detto indice di condizionamento del problema □

Theorem 2.1 (L'indice di condizionamento è sempre maggiore o uguale a 1)**.**

$$\forall A : det(A) \neq 0 \textit{ (non singolare) e } \forall \textit{ norma naturale } \|\cdot\|$$

$$K(A) = \|A\| \cdot \left\|A^{-1}\right\| \geqslant \left\|A \cdot A^{-1}\right\| = \|I\| = 1$$

Theorem 2.2. *Per ogni matrice non singolare* $\mathrm{A} \in \mathrm{R}^{n\times n}$ *e per ogni norma di matrice compatibile con una di vettore, la quantità* $\frac{1}{K(A)}$ *rappresenta la distanza relativa di A dall'insieme di tutte le matrici non singolari*

$$\frac{1}{K(A)} = \min\left\{\frac{\|A-B\|}{\|A\|} : B \in^{n\times n} \quad con \ |B| = 0\right\}$$

Il teorema 2.2 di dice che, all'aumentare di $K(A)$ (il numeretto che ci indica quanto sensibile è il problema a eventuali perturbazioni) diminuisce la distanza tra la matrice A a tutte quelle matrici che hanno determinante parti a 0 ossia matrici singolari. Questo ci interessa perche sappiamo che le matrici singolari sono mal poste. Quindi più aumenta l'indice di condizionamento più la matrice si avvicina ad essere una matrice mal posta.