Calcolo Numerico

Matteo Mazzetti

23/09/2025 - 25/11/2025

1 Sistema floating point

1.1 23/09/2025

1.1.1 Errori nel calcolo

Nella risoluzione di un problema numerico al calcolatore si possono avere diversi tipi di errore:

- Errore di misura;
- Errore di troncamento;
- Errore algoritmico: dovuto alla propagazione degli errori nelle singole operazioni;
- Errore inerente: dovuto alla rappresentazione dei numeri nel calcolatore;

La misura dell'accuratezza è valutata con gli errori assoluti, relativi e percentuali. $E_a = |x_{approx} - x_{esatto}|$
$E_r = \frac{E_a}{|x_{esatto}|}$
$E_p = (E_r \cdot 100)\%$
Fissato un numero intero $\beta > 1$, rappresentiamo nella base β un numero reale qualunque α.

- Rappresentazione di tipo misto: $\alpha = \pm(\alpha_0\alpha_1...\alpha_p.\alpha_{p+1}...)_\beta$
- Rappresentazione scientifica normalizzata: $\alpha = \pm(\alpha_0\alpha_1...)\beta^p$

Notare che una notazione è detta normalizzata se la prima cifra decimale (d_1) è diversa da zero e le altre cifre decimali $(d_i \forall i)$ non sono tutte uguali a $\beta - 1$.

1.1.2 Floating point

Il sistema floating point $\mathbb{F}$ è una quadrupla $\mathbb{F}(\beta, t, L, U) = \{0\} \cup \{x \in \mathbb{R} = sign(x)\beta^p \sum_{i=1}^{t} d_i\beta^{-i}\}$ dove β è la base, t è il numero di cifre della mantissa, $L < 0$ e $U > 0$ costituiscono un range all'interno del quale può essere rappresentato l'esponente.
$x \in \mathbb{F}$ con $x = \pm(0.d_1d_2...d_t)\beta^p$ dove $p \in (L, U)$. E' evidente che $\mathbb{F}$ non sia né continuo né infinito. Anzi l'ampiezza degli intervalli fra un numero e ilsuccessivo dipende da p, i limiti massimi dipendono da U. Mentre centrato in 0 vi è un intorno vuoto che dipende da L.
Esistono inoltre degli standard (IEEE) per rappresentare i numeri sul calcolatore e sono:

- Singola precisione(32 bit): $\mathbb{F}(2, 24, -128, 127)$.
- Doppia precisione(32 bit): $\mathbb{F}(2, 53, -1024, 1023)$.

e in aggiunta nelle varie codifiche ci sono delle eccezioni con determinate sequenze di bit per NaN (not a number), $+\infty$, $-\infty$.
Come "traduco" $x \in \mathbb{R}$ in float? Se $x \in \mathbb{F}$ allora $fl(x) = x$, altrimenti se

- se la mantissa ha più di t cifre ottengo $fl(x)$ per troncamento: $x = \pm 0.d_1d_2...d_td_{t+1}... \rightarrow fl(x) = \pm 0.d_1d_2...d_t$;
- se $p \notin (L, U)$ non posso rappresentare x ma ho Overflow se $p > U$, Underflow se $p < -U$ e $x = 0$ se $p < L$;

Notiamo che anche se si usa il troncamento si parla di errore di arrotondamento ed è tale che $E_a = |fl(x) - x| < \beta^{p-t}$ e $E_r < \frac{1}{2}\beta^{1-t} = eps$. eps è detto precisione di macchina, dipende dal sistema floating point fissato ed è il più piccolo numero rappresentabile tale che $fl(eps + 1) > 1$.

1.2 25/09/2025

1.2.1 Aritmetica floating point

Dato che $\mathbb{F} \subset \mathbb{R}$ le operazioni aritmetiche son definite anche per gli operandi in $\mathbb{F}$ ma non è detto che il loro risultato sia in $\mathbb{F}$.

$$\odot : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$$
$$x \odot y = fl(x \cdot y)$$

Ogni operazione provoca, in generale, un errore di arrotondamento: $|\frac{(x \odot y)-(x \cdot y)}{x \cdot y}| < eps$. Nell'addizione e nella sottrazione lo spostamento della mantissa può causare la perdita di cifre.
Il prodotto di due numeri con t cifre di mantissa ha al più $2t$ cifre, quindi il risultato non sta in $\mathbb{F}(\beta, t, ...)$.
nella divisione il quoziente di un numero con t cifre di mantissa può contenere più di t cifre.

2 Zeri di funzione

2.1 29/09/2025

2.1.1 Risoluzione di equazioni non lineari

In questa sezione calcoleremo con metodi numerici la soluzione di un'equazione non lineare $F(x) = 0$.

1. Esiste (ed è unica) la soluzione del problema "continuo"?
2. Analizziamo più algoritmi che approssimano una soluzione in base alla loro accuratezza e al loro costo computazionale.
3. Analizziamo l'errore.

TEO (Teorema degli zeri): Sia $f : I \subset \mathbb{R} \to \mathbb{R}$ continua in $[a, b]$ tale che $f(a)f(b) < 0$. Allora esiste, ma non perforza è unico, $c \in (a, b)$ tale che $f(c) = 0$.

Usiamo un metodo iterativo per trovare lo zero di una funzione. Se x_0 è assegnato, costruiamo una successione $x_0, x_1, x_2...x_k$ con la seguente proprietà. $lim_{k\to\infty} x_k = x^*$ dove x^* è soluzione.
Il metodo è detto iterativo perché $x_{k+1} = F(x_k)$ e quindi si può implementare con un ciclo for o while.
I criteri di arresto dell'algoritmo sono i seguenti:

- $Ea(x) < toll$
- $|F(x_k)| < toll$
- $k < maxIt$

Si dice inoltre che la successione $(x_k)_k$ generata da un metodo iterativo che converge a x^* è di ordine $p > 1$ se: $\frac{|x_{k+1}-x^*|}{|x_k-x^*|^p} = C \forall k$ dove $C \in \mathbb{R}$ tale che se $p = 1, 0 < C \leq 1$ e se $p > 1, C > 0$. Parliamo di convergenza lineare se $p = 1$, quadratica se $p = 2$ e superlineare se $1 < p < 2$. La cnvergenza di un metodo iterativo con successione $(x_k)_k$ dipenda dalla scelta iniziale di x_0. Inoltre si parla di convergenza globale se il risultato non dipende dalla scelta di x_0, locale altrimenti.

2.1.2 Metodo di bisezione

L'idea è di costruire una successione di intervalli $I_1 = [a_1, b_1], I_2 = [a_2, b_2], ..., I_k = [a_k, b_k]; I_k \subset I_{k-1} \subset ... \subset I_1$ con $f(a_k)f(b_k) < 0 \ \forall k = 1, 2, ...$ dove $a_1 = a \ b_1 = b$.
Al passo k si calcola $c_k = \frac{a_{k-1}+b_{k-1}}{2}$ e valuto $f(c_k)$. Se $f(c_k) = 0$ allora $x^* = c_k$, altrimenti $[a_{k+1}, b_{k+1}] = [c_k, b_k] \vee [a_k, c_k]$ a seconda se $f(c_k) < 0$ o $f(c_k) > 0$.
Le dimensioni degli intervalli seguono la legge $I_{k+1} = \frac{1}{2^k} I_k$ e metodo ha convergenza lineare con $c = 1/2$.

2.2 30/09/2025

2.2.1 Metodo del punto fisso

Cercare la radice di $f(x)$ è legato alla ricerca di x tale che $x = g(x)$, cioè alla ricerca di un punto fisso di $g(x) = x - f(x)\phi(x)$ dove $\exists c \in [a,b] : f(c) = 0$ e $0 < |\phi(x)| < \infty, \forall x \in [a,b]$.
Le soluzioni di $x = g(x) = x - f(x)\phi(x)$ e di $f(x) = 0$ sono analoghe:
DIM: che $f(x) = 0 \implies g(x) = x$ è ovvio, invece $g(x) = x \implies f(x) = 0$ perché per ipotesi $\phi(x) \neq 0$.

Dato x_0, $x_{k+1} = g(x_k)$, quindi calcolo una successione $(x_k)_k = x_0, g(x_0), g(g(x_0)), ...$ Vedremo fra poco dei teoremi che ci garantiscono che questa successione per $k \to \infty$ tende a x^*.

2.3 05/10/2025

2.3.1 Teoremi per Fixed point

TEO (esistenza del punto fisso): sia $g(x)$ continua in $[a,b]$ tale che $g(x) \in [a,b]$ e sia $L \in (0,1]$ tale che $\forall x, y \in [a,b]$ vale $|g(x) - g(y)| \leq L|x - y|$(ossia g deve essere una contrazione[1] in $[a,b]$). Allora $\exists! x^*$ punto fisso di g in $[a,b]$.

EG: Verifichiamo se $h(x) = cos(x)$ è una contrazione in $[0, 2\pi]$. $h'(x) = -sin(x) \implies |h'(x)| \leq 1, \forall x \in [0, 2\pi] \implies h$ non è una contrazione.

TEO (convergenza globale dei metodi iterativi): sia $g(x)$ definita in $[a,b]$ tale che:
(1) g è continua in $[a,b]$;
(2) $g(x) \in [a,b]$;
(3) $g(x)$ contrazione in$[a,b]$
Allora $\forall x_0 \in [a,b]$ la successione per iterati $(x_k)_k$ costruita con $x_{k+1} = g(x_k)$ converge, per $k \to \infty$, all'unico punto fisso x^* di g in $[a,b]$.
Inoltre vale che $|x_k - x^*| \leq \frac{L^k}{1-L}|x_1 - x_0|$[1] quindi la covergenza è lineare.

Più L è piccolo, più la frazione è piccola, ovvero più la costante di contrazione è piccola più il metodo è rapido.
Notare che se una funzione non è una contrazione si perde l'unicità del punto fisso.
TEO (covergenza locale): sia x^* un punto fisso di g con g continua e contrazione $\forall x \in [x^* - \rho, x^* + \rho]$. Allora $\forall x_0 \in I_\rho$ la successione $(x_k)_k$ converge a x^* per $k \to \infty$. Inoltre x^* è l'unico punto fisso di g in I_ρ.

2.4 07/10/2025

2.4.1 Metodo di Newton

Il metodo di Newton ha le sue fondamenta sul metodo di punto fisso ma, per ottenere una convergenza quadratica, si sceglie $\phi(x) = \frac{1}{f'(x)}$.
Quindi gli iterati sono costruiti seguendo il modello $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$.
La convergenza è quadatica perché $|x_k - x^*| \leq C \cdot |x_{k-1} - x^*|^2$.

TEO (convergenza locale): sia x^* uno zero di $f(x) \in C^3([a,b])$ con la derivata prima e seconda diverse da zero. Allora $\forall x_0 \in I_\rho$ la successione converge a x^*.

[1] g è una contrazione in $[a,b]$ se $|g'(x)| \leq L < 1, \forall x \in [a,b]$

3 Minimi di funzione

3.1 14/10/2025

3.1.1 Ottimizzazione non vincolata

Il problema che vogliamo risolvere è $arg(min_x(f(x))$ dove $f : \mathbb{R}^n \to \mathbb{R}^m$ ed è importante osservare che il problema è legato a quello del massimo: $argmax_x(f(x)) = arg(-min_x(-f(x))$

DEF (minimo locale): x^* è un punto di minimo locale di f se $\exists \epsilon > 0 : f(x^*) \leq f(x), \forall x \in \|x - x^*\| < \epsilon$ dove $\|x\| = \sqrt{\langle x, x \rangle}$ e $I_\epsilon(x^*) = \{\|x - x^*\| < \epsilon\}$, ovvero $I_\epsilon(x^*)$ sono i punti appartenenti alla sfera centrata in x^* di raggio ϵ.

DEF (minimo globale): x^* è di minimo globale per f se $\forall x \in \mathbb{R}^n, f(x^*) \leq f(x)$.

TEO (Weierstrass): sia $S \subset \mathbb{R}^n$ un insieme compatto e non vuotoe sia $f : S \to \mathbb{R}$ continua su S. Allora esiste x punto di minimo globale di f.

DEF (differenziabilità): sia $f : \mathbb{R}^n \to \mathbb{R}$, questa si dice differenziabile in $x_0 \in \mathbb{R}^n$ se esiste un'applicazione lineare $L : \mathbb{R}^n \to \mathbb{R} : lim_{h \to 0} \frac{f(x_0+h) - f(x_0) - L(h)}{\|h\|} = 0$.

PROP (proprietà funzioni differenziabili):
(1) se f è fifferenziabile in $x_0 \implies \exists \partial_i f(x_0), \forall i = 1...n$.
(2) se f è fifferenziabile in $x_0 \implies f$ è continua in x_0.

3.1.2 Condizioni di ottimo

TEO (condizione necessaria del prim'ordine): se x^* è di minimo locale e f è differenziabile in x^* allora $\nabla f(x^*) = \underline{0}$ e x^* viene detto punto stazionario.

DEF (matrici positive, negative, indefinite): sia $A \in M_n(\mathbb{R})$ simmetrica:
(1) A è definita positiva se $\forall x \in \mathbb{R}^n - \{\underline{0}\} : x^T A x > 0$;
(2) A è definita negativa se $\forall x \in \mathbb{R}^n - \{\underline{0}\} : x^T A x < 0$;
(3) A è indefinita se $\exists x, y \in \mathbb{R}^n - \{\underline{0}\} : x^T A x > 0 \wedge y^T A y < 0$;

PROP:
(1) A è definita positiva ed è simmetrica $\implies$ A ha tutti gli autovalori positivi.
(2) A è definita negativa ed è simmetrica $\implies$ A ha tutti gli autovalori negativi.
(3) A è indefinita ed è simmetrica $\implies$ A ha sia autovalori positivi che negativi.

TEO (condizioni necessarie del secondo ordine): se x^* è di minimo locale per f e f è due volte differenziabile con continuità in x^* allora Hf è semidefinita positiva.

TEO (condizioni sufficienti del secondo ordine): sia f due volte differenziabile con continuità in un intorno aperto di x^* e $\nabla f(x^*) = 0$ allora:
(1) se $Hf(x^*)$ è (semi)definita positiva $\implies$ x^* è di minimo locale.
(1) se $Hf(x^*)$ è (semi)definita negativa $\implies$ x^* è di massimo locale.
(1) se $Hf(x^*)$ non è definita $\implies$ non si può classificare x^*.

DEF (funzioni convesse): f si dice convessa in $\mathbb{R}^n$ se $f(\alpha x + (1-\alpha)y \leq \alpha f(x) + (1-\alpha)f(y), \forall \alpha \in [0,1], \forall x, y \in \mathbb{R}^n$.

TEO (ottimalità nel caso convesso): se f è convessa allora $\forall x^*$ minimo locale x^* è di minimo globale. Inoltre se f è strettamente convessa $\exists! x^*$ punto di minimo globale.

3.2 20/10/2025

3.2.1 Metodi di discesa

Nel caso non convesso, la determinazione di punti stazionari non fornisce una soluzione globale e, in generale, non è neanche possibile garantire che sia raggiunto un minimo locale.
I metodi che utilizzeremo sono metodi iterativi che, partendo da un iterato iniziale $x_0 \in \mathbb{R}^n$ geerano una successione definita dalla legge $x_{k+1} = x_k + \alpha_k p_k$ dove p_k è una direzione di ricerca e α_k è uno scalare detto lunghezza del passo. I vettori p_k e α_k sono scelti in modo da garantire la decrescita di f ad ogni iterazione.
DEF: il vettore p_k è una direzione di discesa di f in x se $\exists \alpha' > 0 : f(x + \alpha p) < f(x), \forall \alpha \in (0, \alpha']$.

PROP: sia $f : \mathbb{R}^n \to \mathbb{R}$ continuamente differenziabile nell'intorno di un punto $x \in \mathbb{R}^n$ e sia $p \in \mathbb{R}^n - \{\underline{0}\}$. Allora se $p^T \nabla f(x) < 0$ la direzione p è di discesa per f in x.

Siano $x, y \in \mathbb{R}^n - \{\underline{0}\}$, ricordiamo che l'angolo fra di essi si può trovare grazie alla definizione di coseno: $cos(\theta) = \frac{\langle x,y \rangle}{\|x\|\|y\|}$ e posso dire che θ è ottuso se $cos(\theta) < 0$ e acuto se $cos(\theta) > 0$, mentre se $\langle x, y \rangle = 0$ allora i due vettori sono ortogonali. Queste considerazioni diventano interessanti se consideriamo $x = p$ e $y = \nabla f$.
La retta $x = x_k + \alpha p_k$ deve formare un angolo ottuso con la direzione del gradiente. Di solito si sceglie $p_k = -\nabla f(x_k)$ (antigradiente) così è ovvio che l'angolo sia ottuso.
La scelta del passo (ricerca in linea) può essere fatta in divesi modi tra cui:

- prendere un α piccolo, EG: 10^-2;
- utilizzare un algoritmo di backtracking;

Osserviamo, innanzitutto, che la scelta di $\alpha_k : f(x_{k+1}) < f(x_k)$ non garantisce la convergenza del metodo. E' la scelta del passo che deve garantiere sia la convergenza che la rapidità dell'algoritmo.
Seguono le condizioni di Wolfe per la ricerca in linea inesatta:

- Armijo: $f(x_k + \alpha_k p_k) \leq c_1 \alpha \langle \nabla f(x_k), p_k \rangle$;
- curvatura: $\langle \nabla f(x_k + \alpha_k p_k), p_k \rangle \geq c_2 \langle \nabla f(x_k), p_k \rangle$;

```
function backtracking(alpha, rho, c1):
    j=0
    while(f(xk+alpha*pk)>f(xk)+c1*alpha*df(xk)*pk):
        alpha=rho*alpha
        j++
    return alpha
```

3.2.2 Metodo del gradiente

Scegliamo $p_k = -\nabla f(x_k), \forall k$ come direzione di ricerca: l'antigradiente è sempre direzione di discesa di f in x_k se è diverso dal vettore nullo $\underline{0}$.
Usiamo poi una ricerca in linea inesatta.
Per il teorema di Zontendijk è assicurata la convergenza ad un minimo locale.

```
function GD(f,df,x0,xToll,fToll,maxIt):
    cont=0
    p=-df
    while(cont<maxIt && dfNorm>fToll && xNorm>xToll):
        alpha=backtracking(1,0.5,1e-4)
        xNew=x0+alpha*p
        xNorm=norm(xNew-x0)
        dfNorm=norm(df(xNew))
        x0=xNew
        cont++
    return x0
```

3.2.3 Metodo di discesa di Newton

Questo metodo usa come direzione di ricerca la direzione di Newtow $p_k = -Hf^{-1}(x_k)\nabla f(x_k)$, quindi $x_{k+1} = x_k - Hf^{-1}(x_k)$ dove $\alpha = 1$. Però inquesto caso p_k è una direzione di disscesa solo se l'Hessiana è definita positiva.
Questo metodo se converge ha un ordine di convergenza pari a 2, ma necessita di un iterato iniziale già vicino al minimo. Inoltre il costo computazionale dell'algoritmo di Newton è grandissimo in quanto, ad ogni iterazione, devo calcolare l'inversa di una matrice.

3.2.4 Gradiente stocastico

L'idea è che se la funzione da minimizzare $F(x)$ è somma di altre funzioni $f_i(x)$ allora anche il gradiente, siccome è lineare, lo è:
$F(x) = \sum_{i=1}^{n} f_i(x) \implies \nabla F(x) = \sum_{i=1}^{n} \nabla f_i(x)$. Con il metodo SGD approssimo quindi $\nabla F(x)$ utilizzando ad ogni iterazione solo alcuni $\nabla f_i(x)$: $\nabla F(x) \approx \sum_{i \in S_k} \nabla f_i(x)$ con $S_k = \{i_1, ..., i_k\}$. Fissato k (dimensione del mini-batch) scelgo in maniera casuale gli indici nell'intervallo $[1, n]$ senza ripetizioni. Eseguo questo procedimento(epoca) per un numero di volte impostato in precedenza.

4 Analisi dati

4.1 28/10/2025

4.1.1 Predizione

Assegnate n coppie di dati (x_i, y_i), un modello predittivo è una funzione $f(x)$ che dipende da due o più parametri $\theta_1, ..., \theta_k$ che rappresenta i dati. In particolare x è una variabile indipendente, y è dipendente e $f(x) = y$ mette in relazione le due.
Assegnato $\tilde{x} \in [min(x_i), max(x_i)]$ il modello predittivo deve predirre un valore $\tilde{y}$ coerente con i dati forniti. Come faccio però a determinare i parametri θ?
Stabiliamo una misura (funzione di costo) di vicinanza fra i dati (x_i, y_i) e le predizioni: $\mathcal{L}(\theta) = \sum_{i=1}^{n} h(f(x_i; \theta), y_i)$.
$\mathcal{L}$ sta per loss-function e il nostro obiettivo è cercare di minimizzarla, ossia calcolare $min_{\theta \in \mathbb{R}^n} \mathcal{L}(\theta)$.
Questa funzione, i cui parametri sono stimati come minimo della funzione di perdita $\mathcal{L}$, si chiama funzione di regressione. Se $\mathcal{L}$ è un polinomio parliamo di regressione polinomiale: f è un polinomio e $h(f(x_i; \theta), y_i) = (f(x_i; \theta) - y_i)^2$ è definita da $\mathbb{R} \to \mathbb{R}$, quindi $\mathcal{L}(\theta) = \sum_{i=1}^{n} (f(x_i; \theta) - y_i)^2$ dove, come al solito, θ è il vettore di coefficenti del polinomio.
Se la regressione è lineare $f(x_i; \theta) = \theta_0 + \theta_1 x$ ment nel casodi un polinomio di grado k la regressione è $f(x_i; \theta) = \theta_0 + \theta_1 x + \theta_2 x^2 ... + \theta_k x^k$. Quindi $\mathcal{L}$ di una regressione polinomiale è $\mathcal{L}(\theta) = \sum_{i=1}^{n} (\theta_0 + \theta_1 x + \theta_2 x^2 ... + \theta_k x^k) - y_i)^2$ e se $\theta = (\theta_1, ..., \theta_k)^T, y = (y_1, ..., y_n)^T$ e $\phi \in M_{n,k+1}(\mathbb{R})$ tale che

$$\phi = \begin{pmatrix} x_{1,0} & x_{1,1} & \cdots & x_{1,k} \\ x_{2,0} & x_{2,1} & \cdots & x_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,0} & x_{n,1} & \cdots & x_{n,k} \end{pmatrix}$$

allora $\mathcal{L}(\theta) = \|\phi\theta - y\|^2$ e quindi i parametri theta li trovo risolvendo attraverso tecniche di algebra lineare il sistema ai minimi quadrati $min_{\theta \in \mathbb{R}^k} \|\phi\theta - y\|^2$.

5 Algebra lineare

5.1 03/11/2025

5.1.1 Norme vettoriali

DEF (norma): un'applicazione lineare $\mathbb{R}^n \to \mathbb{R}_+$ è chiamata norma se verifica le seguenti condizioni:
(1) $\forall x \in \mathbb{R}^n, \|x\| \geq 0$
(2) $\|x\| = 0 \iff x = \underline{0}$
(3) $\|\alpha x\| = |\alpha| \cdot \|x\|, \forall x \in \mathbb{R}^n, \forall \alpha \in \mathbb{R}$
(4) $\forall x, y \in \mathbb{R}^n$ vale la disuguaglianza triangolare $\|x + y\| \leq \|x\| + \|y\|$

Definiamo la norma a $p \in [1, \infty)$ come $\|x\|_p = (\sum_{i=1}^{n} (|x_i|^p)^{\frac{1}{p}}$.

- con $p = 2$ si ottiene la norma euclidea;
- con $p = \infty : \|x\|_\infty = max_{i=1,...,n}|x_i|$, detta norma del massimo;
- con $p = 1 : \|x\|_1 = |x_1| + ... + |x_n|$, detta norma di Manhattan;

Per ogni coppia di norme $\|x\|$ e $\|x\|', \exists m, M \in \mathbb{R}, \forall x \in \mathbb{R}^n : m\|x\|' \leq \|x\| \leq M\|x\|'$ o, in altre parole, in $\mathbb{R}^n$, per n fissato le norme sono tra loro equivalenti.

5.1.2 Norme matriciali

Vediamo ora come sono definite le norme per le matrici:
DEF: $\|\cdot\| : M_{m,n}(\mathbb{R}) \to \mathbb{R}$ tale che:
(1) $\|A\| \geq 0$
(2) $\|A\| = 0 \iff A = \underline{0}$
(3) $\|\lambda A\| = |\lambda| \cdot \|A\|, \forall \lambda \in \mathbb{R}$
(4) $\|A + B\| \leq \|A\| + \|B\|$

Esistono, anche per le matrici, più tipologie di norme:

- di Frobenious: $\|A\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}^2}$, ossia la somma dei quadrati degli elementi sotto radice;
- a 2: $\|A\|_2 = \sqrt{\rho(A^T A)}$ dove $\rho = \lambda_{max}$ di $A^T A$;
- a 1: $\|A\|_1 = max\{C_1, .., C_n\}$ dove $C_i = \sum_{j=1}^{m} |a_{ij}|$, ossia il massimo tra la somma delle colonne in valore assoluto;
- a ∞: $\|A\|_\infty = max\{R_1, .., R_m\}$ dove $C_i = \sum_{j=1}^{n} |a_{ij}|$, ossia il massimo tra la somma delle righe in valore assoluto;

5.1.3 Sistemi lineari

Per i sistemi lineari $Ax = b$ con $A \in M_m, n(\mathbb{R})$ considereremo solo il caso di matrici quadrate. A noi interessano:

1. esistenza e unicità delle soluzioni;
2. algoritmi per calcolarle;
3. errore inerente;

Il sistema $Ax = b$ ammette una sola soluzione $\forall b \in \mathbb{R}^n \iff A$ è non singolare ($detA \neq 0$). Se $detA \neq 0 \implies \exists A^{-1}$ quindi $Ax = b \iff A^{-1}Ax = A^{-1}b \iff Ix = A^{-1}b \iff x = A^{-1}b$.
Il calcolo dell'inversa di una matrice però è molto costoso computazionalmente $O(n^3)$. Per calcolare la soluzione possiamo usare metodi diretti oppure metodi iterativi che sono meno costosi e quindi utilizzati per matrici grandi.
L'idea dietro ai metodi diretti è quella di sostituire alla matrice iniziale A, una matrice equivalente più semplice (triangolare). Partendo da una matrice triangolare infatti il costo di risoluzione del sistema è ridotto a $O(n^2)$.
Il metodo di eliminazione di Gauss e quello LU (LR) sono esempi di metodi diretti.

5.1.4 Metodo LU

L'idea è che la matrice completa A sia uguale ad LU: partedo da

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{pmatrix}$$

e usando il moltiplicatore $e_{ij} = -\frac{a_{ij}}{a_{ii}}$ sulla prima colonna, quindi con $i = 2...n$ e $j = 1$ ottengo la matrice

$$E_1 = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ e_{21} & 1 & 0 & \cdots & 0 \\ e_{31} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ e_{n1} & 0 & 0 & \cdots & 1 \end{pmatrix}$$

che annulla la prima colonna sotto il pivot e

$$E_1 A = A_{(1)}$$

dove A^1 ha zeri sotto a_{11}.
Ripeto, cambiando colonna ogni volta per n-1 passi per ottenere che $E_{n-1}E_{n-2}...E_1A = U$ dove $U = A_{n-1}$ e $A = LU$ con $L = \prod_{i=1}^{n-1} E_i^{-1}$. Il costo computazionale di questo metodo è $O(n^3/3)$ che è un po' meglio del calcolo diretto dell'inversa.
Questo metodo non funziona però se A ha almeno un elemento nullo sulla diagonale perché non posso calcolare E perché dovrei fare una divisione per zero. Modifichiamo l'algorimo come segue:
ad ogni passo k, prima di calcolare E_k, scambio le righe della matrice A in modo che l'elemento $a_{kk} = max_{i=1...n}\{|a_{ik}|\}$. Lo scambio di righe i, j nella matrice si può fare introducendo una matrice di permutazione $P = I$ con righe i e j scambiate e facendo diventare A la matrice PA.
Questo metodo è detto LU con Pivoting e per risolvere un sistema lineare $Ax = b$ devo risolvere $LUx = Pb$ risolvendo da prima $Ux = y$ e poi $Ly = Pb$.

5.2 10/11/2025

5.2.1 Numero di condizione

Il numero di condizione $K(A) = \|A\| \cdot \|A^{-1}\|$ è un valore che ci permette di capire se il problema è mal condizionato o meno. Dato che abbiamo studiato che le norme sono equivalenti il numero di condizione ha lo stesso significato al variare dalla norma scelta.

- se $K(A) = 10^p$ con $p = 0, 1, 2, 3 \implies$ il problema è ben condizionato;
- se $K(A) = 10^n \implies$ il problema è mal condizionato;

Se il problema è mal condizionato non ha senso cambiare l'algoritmo, bisogna variare un poco i dati di partenza: al posto di usare A uso $\tilde{A}$ con valori molto vicini a quelli della matrice originale. Sia x^* tale che $Ax^* = b$, allora si studia la soluzione del sistema pertubato $(A + \Delta A)\tilde{x} = b + \Delta b$. Dopo vari calcoli otteniamo che $\frac{\|\Delta x\|}{\|x+\Delta x\|} \leq \|A\|\|A^{-1}\|\frac{\|\Delta A\|}{\|\Delta A+A\|}$.
E' importante ricordare che $K(A) \geq 1, \forall A$.

5.2.2 Fattorizzazione SVD

DEF: i vettori $v_1, ..., v_m$ si dicono ortonormai se $v_i^T v_j = 0, \forall i \neq j$ (ortogonalità) e $\|v_h\| = 1 \forall h$ (norma unitaria).

DEF: una matrice A è ortogonale se le sue colonne sono vettori ortonormali. In tal caso A ha le seguenti proprietà:
(1) $A^{-1} = A^T$;
(2) $\forall x \in \mathbb{R}^n, \|x\| = \|Ax\|$, quindi A è un isometria, cioè manitene le distanze;

TEO: se gli autovettori di A sono linearmente indipendenti, allora $A = PDP^{-1}$ dove D è diagonale e contiene gli autovalori e P gli autovettori di A.

TEO (Singular Value Decomposition): sia $A \in M_{m,n}(\mathbb{R})$ di rango $k \leq n \leq m$, allora esistono:
(1) una matrice ortogonale $U, m \times m$;
(2) una matrice ortogonale $V, n \times n$;
(3) una matrice diagonale $\Sigma, m \times n$ che sulla diagonale ha tutti i valori singolari di A;
tali che $A = U\Sigma V^T$ dove σ_i sono i valori singolari di A ordinati in ordine decrescente ($\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_n$)

mentre le colonne di $U = (u_1, ..., u_m)$ e di $V = (v_1, ..., v_n)$ sono rispettivamente i vettori singolari sinistri e destri di A.

Fra i valori singolari di A e gli autovalori di A^TA vale la seguente relazione: $A^TA = (U\Sigma V^T)^T(U\Sigma V^T) = V\Sigma^T U^T U\Sigma V^T = V\Sigma^2 V^T$ che sono rispettivamente PDP^T e quindi Σ^2 che contiene σ_i^2 contiene anche gli autovalori di $A^TA : \lambda_i = \sigma_i^2$.
Nota 1: $\sigma_1 = \|A\|$ per definizione.
Nota 2: gli autovalori $\mu_i(A)$ sono l'inverso degli autovalori $\lambda_i(A)$ e ciò vale anche per i valori singolari.
Da tutto ciò otteniamo anche che $K(A) = \|A\|\|A^{-1}\| = \frac{\sigma_{max}}{\sigma_{min}}$.

5.2.3 Minimi quadrati

Vogliamo risolvere $Ax = b$ con $A \in M_{m,n}(\mathbb{R}), b \in \mathbb{R}^m, x \in \mathbb{R}^n$. Il sistema in generale non ha soluzioni, cerchiamo dunque di rendere più piccolo possibile il vettore residuo $r = Ax - b$, quindi cerco $min\|r\|$ che è analogo a cercare $min\|r\|^2 = min\|Ax - b\|^2$.

1. Quando c'è soluzione (unica) al problema? C'è sempre soluzione perché $\|Ax - b\|^2$ è una funzione convessa, quindi ha sempre un minimo.
   - se $rg(A) = n$(massimo): in questo caso la soluzione è unica;
   - se $rg(A) = k < n$: ci sono infinite soluzioni;
2. Come calcolo la soluzione?
   - metodo del gradiente;
   - se $rg(A) = n$ risolvo il sistema lineare delle equazioni normali $A^TAx = A^Tb$, derivate dalla condizione del primo ordine $\nabla f = \underline{0} \implies \nabla(\|Ax - b\|^2) = 0 \implies A^TAx - A^Tb = 0$;
   - se $rg(A) = k < n$ uso la decomposizione SVD;

5.3 11/11/2025

5.3.1 SVD per risolvere LSQ

Per iniziare decomponiamo $A = U\Sigma V^T$ come abbiamo visto prima.
Sostituiamo A con $U\Sigma V^T$ in $\|Ax - b\|_2^2 = \|U\Sigma V^T x - b\|_2^2$.
Dato che U è ortogonale ed è quindi un isometria $\|x\| = \|Ux\|$ e quindi anche U^T lo è, moltiplico quindi tutto ciò che c'è dentro la norma per $\|U^T\|$ e ottengo $\|U\Sigma V^T x - b\|_2^2 = \|U^T(U\Sigma V^T x - b)\|_2^2$.
Distribuendo U^T ottengo $\|\Sigma V^T x - U^T b\|_2^2$.
Siano $y = V^T x \in \mathbb{R}^n, g = U^T b \in \mathbb{R}^m$ rispettivamente nuova incognita e nuovo termine noto: $\|\Sigma V^T x - U^T b\|_2^2 = \|\Sigma y - g\|_2^2$.
Poiché Σ è diagonale (e dopo la parte diagonale ha delle righe nulle) allora $\|\Sigma y - g\|_2^2 = \sum_{i=1}^r(\sigma_i y_i - g_i)^2 + \sum_{i=r+1}^m(-g_i)^2$ dove $r = rg(A)$.
Per ridurre le due sommatorie ad una sola ricordo che $\sigma_{r+1} = ... = \sigma_m = 0$ quindi ottengo $\sum_{i=1}^m(\sigma_i y_i - g_i)^2$ che convessa quindi ha un minimo in $\underline{0}$. I valori di y_i nel minimo li ottengo ponendo $\sigma_i y_i - g_i = 0 \implies y_i = \frac{g_i}{\sigma_i}$.
Ma $y = V^T x = V^{-1}x$ perché V è ortogonale, quindi $x = Vy$ è il vettore che minimizza il sistema dove $y = (y_1, ..., y_n), y_i = \frac{g_i}{\sigma_i} = \frac{u_i^T b_i}{\sigma_i}$.

5.3.2 Approssimazione con SVD

Sia $A = U\Sigma V^T, rg(A) = k$ e quindi $A = \sum_{i=1}^k \sigma_i u_i v_i^T = \sum_{i=1}^k \sigma_i A_i$. Per ottenere un'approssimazione di A con rango $p < k$ posso troncare la somma all'indice $i = p : A \approx A_p = \sum_{i=1}^p \sigma_i A_i$ dove l'errore $\|A - A_p\|$ risulta essere σ_{p+1} perché i valori singolari sono scritti in ordine decrescente.
TEO: l'approssimazione fatta con SVD di ordine p di una matrice A con rango k è la miglior approssimazione di ordine p di A.

6 Imaging

6.1 24/11/2025

6.1.1 Le immagini in digitale

Ogni immagine digitale di $M \times N$ pixel non è altro che una matrice di dimensioni analoghe ed i suoi coefficienfi rappresentano quali sfumature di colore vengono mostrate in ogni pixel. Per esempio, un'immagine in scala di grigi è una matrice in cui ogni elemento ha un valore intero variabile tra 0-255 dove 0 corrisponde al nero mentre 255 è associato al colore bianco.

6.1.2 Convoluzione

In generale la convoluzione è una operazione matematica che, date due funzioni, ne produce una terza che è calcolata come: $(f * g)(t) = \int f(\tau)g(t-\tau)d\tau$. $*$ è l'operatore di convoluzione.
Nell'imaging f, g sono funzioni discrete bidimensionali, corrispondenti a delle matrici, e la convoluzione in 2D assume la formula: $(f * g)(x, y) = \sum_{i=0}^{M} \sum_{j=0}^{N} f(x, y)g(x-i, y-j)$ con $x, y = 0, ..., N$.
Altrimenti possiamo definire la convoluzione introducendo una matrice K detta nucleo (o kernel) di convoluzione. In tal caso dobbiamo:

1. considerare $K, n \times n$, con n dispari, il cui elemento centrale viene fatto coincidere coi un pixel di f a cui verrà apllicata la convoluzione;
2. moltiplicare i valori nella matrice con i valori dei pixel corrispondenti e sommare tutti i risultati;
3. il valore ottenuto è il valore del pixel centrale nel risultato;
4. ripetere queste operazioni per tutti i pixel della matrice di partenza.

E' evidente la necessità di estendere le immagini di $(n-1)/2$ pixel per applicarci una convoluzione con kernel K $n \times n$. Possiamo farlo mettendoci dei bordi neri, ripetendo parte dell'immagine, specchiano l'immagine usando come asse di simmetria i bordi originali, ecc..
I diversi modi di estendere una matrice prendono il nome di condizioni al bordo.
osserviamo inoltre che al variare dei kernel di convoluzione l'effetto finale sarà diverso.

6.1.3 Sistemi di formazione dell'immagine

Supponiamo di voler fotografare una singola fonte luminosa puntiforme, l'immagine che otteniamo è più simile a una macchia centrata nel punto in cui avremmo dovuto trovare la nostra fonte luminosa puntiforme. Questo fenomeno di "allargamento" delle fonti luminose è deterministico e legato all'apparecchiatura utilizzata per catturare l'immagine.
Matematicamente possiamo modellizzare il fenomeno definendo una funzione $\mathcal{A}$ detta Point Spread Function (PSF) che, in pratica, permette di allargare la dimensione di un'immagine. Come possiamo immaginare $\mathcal{A}$ è una convoluzione.
Inoltre, nei sistemi di formazione di immagini è stato osservato che la forma della "macchia" non dipende dalla posizione della fonte luminosa, di conseguenza questi sistemi sono detti spazio-invarianti. La PSF risulta quindi essere la stessa per tutte le possibili posizioni della fonte luminosa.
Se x è l'immagine "perfetta" e $\mathcal{A}$ è la PSF, vale che $\mathcal{A}(x) = [K * x|\mathcal{P}]$ dove K è un kernel di convoluzione e $\mathcal{P}$ è un pattern di estensione.
Sia ora e un rumore gaussiano che aggiungeremo alla nostra immagine: $y^\delta = \mathcal{A}(x) + e = [K * x|\mathcal{P}] + e$ o anche $y^\delta = Ax + e$ dove $Ax = K * x$. Definiamo un problema in cui y^δ è l'immagine sfuocata e $\mathcal{A}$ è la PSF legata al nostro sistema di formazione delle immagini e noi vogliammo ricavare l'immagine x originaria. Questo problema si può affrontare come un LSQ: $\|Ax - y^\delta\|_2^2$ che può essere risolto con GD utilizzando la funzione di backtracking per la scelta del passo ottimale.

6.2 25/11/2025

6.2.1 CGLS

Per risolvere $min\frac{1}{2}\|Ax-b\|_2^2$ se A è grande e con tanti elementi nulli, senza utilizzare le equazioni normali perché A è mal condizionata, utilizziamo CGLS (Conjugate Gradient for Least Squares) ricordando che:

- L'algoritmo fa uso della matrice A solo con operazioni del tipo Ax oppure $A^T z$.

- Il numero di operazioni di tipo Ax e $A^T z$ è il più piccolo possibile.

```
x[0]=(0,0,...,0)                              //iterato iniziale
r[0]=A.t(y-A@x[0])                            //residuo iniziale
p[0]=r[0]                                     //Direzione di discesa
for k=0...maxIt:
    q[k]=A@p[k]
    alpha[k]=(norm(r[k])/norm(q[k]))^2        //Calcolo alpha
    x[k+1]=x[k]+alpha[k]p[k]                  //Nuovo iterato
    r[k+1]=r[k]-alpha[k]@A.t@q[k]             //Aggiorno il residuo
    beta[k]=(norm(r[k+1])/norm(r[k]))^2
    p[k+1]=r[k]+beta[k]p[k]                   //Nuova direzione di discesa
```

E' utile notare che se $A \in M_{m,n}(\mathbb{R})$, allora l'algoritmo converge al più alla n-esima iterazione.

6.2.2 Regolarizzazione Tikhonov

L'algoritmo CGLS può essere adattato per risolvere problemi LSQ con regolarizzazione della forma $min\frac{1}{2}\|Ax-y\|_2^2 + \frac{\lambda}{2}\|Lx\|_2^2$ dove $L \in M_{d,n}(\mathbb{R})$ è la matrice di Tikonov (spesso si sceglie $L = I$) e $\lambda > 0$ è il parametro di regolarizzazione.
Le equazioni normali di questo problema sono: $(A^T A + \lambda L^T L)x = A^T y$. La domanda che ci poinamo adesso è come segliere λ per l'imaging.
Il λ_{best} è quello che minimizza: $\lambda_{best} = min_\lambda \|x_\lambda - x^*\|_2^2$. In genere si fa del Trial&Test con vari $\lambda > 0$, ma si potrebbe anche usare (e di solito accade) il principio di massima discrepanza. λ_{DP} deve soddisfare la seguente relazione: $\|Ax_{TIK} - y^\delta\|_2 = \tau_{DP}\|\delta\|_2$. Quindi la norma del residuo deve essere uguale alla discrepanza sui dati , $\tau_{DP}\|\delta\|_2$.
Il parametro τ_{DP} ciene in genere scelto come valore poco maggiore di 1: 1.01 oppure 1.001.
Nota: DP richiede la conoscenza della norma del rumore $\|\delta\|_2$ ed è raro conoscerla.

6.2.3 Total Variation

Una funzione di regolarizzazione alternativa a Tikhonov è la funzione di variazione totale: $TV_\beta(x) = \|\nabla(x)\|_1 = \sum_{i=1}^m \sum_{j=1}^m \sqrt{(x_{i+1,j} - x_{ij})^2 + (x_{i,j+1} - x_{i,j})^2 + \beta^2}$ dove $\beta > 0$ è un parametro inserito per rendere $TV(x)$ differenziabile in $(0,0)$ ed in genere è dell'ordine di 10^{-3}.
Il problema di regolarizzazione diventa $min_x \|Ax - y^\delta\|_2^2 + \lambda TV_\beta(x)$ che viene risolto applicando l'algoritmo GD con backtracking.

6.2.4 Metriche di valutazione

In un problema, per valutare la qualità della ricostruzione ottenuta si utilizzano delle metriche, ma per farlo bisogna essere a conoscenza dell'immagine perfetta x^*. Utilizziamo:

- l'errore relativo: $E_r = \frac{\|x-x^*\|_2}{\|x^*\|_2}$;
- l'errore quadratico medio: $MSE = \frac{\|x-x^*\|_2^2}{M \cdot N}$ dove M, N sono le dimensini dell'immagine;
- il rapporto segnale rumore Peak to Noise Signal Ratio: $PSNR = 10log_{10}\frac{(max_{i,j}|x_{ij}|)^2}{MSE}$;
- lo Structural Similarity Index (SSIM) che è l'indice più fedele alla qualità visiva dell'immagine;