Calcolo Numerico
Appunti

Giovanni Palma e Alex Bastianini

Contents

Chapter 1

I numeri finiti

1.1 Base-N and Binary

I numeri hanno diversi modi per essere rappresentati, tra cui quello più comune che è la **rappresentazione decimale**, dove ogni numero è formato da una lista di caratteri compresi tra $[0, 9]$ dove ad ogni cifra è associata una potenza del 10

Example 1.1.1

Esempio: $147.3 = 1 \cdot 10^2 + 4 \cdot 10^1 + 7 \cdot 10^0 + 3 \cdot 10^{-1}$

In generale, comunque, per convertire un numero da una base n a base 10:

Theorem 1.1.1

Dato un numero in base n $(a_k a_{k-1} \dots a_1 a_0)_n$, la sua conversione in base 10 è data dalla formula:

$$\sum_{i=0}^{k} a_i \cdot n^i$$

dove a_i sono le cifre in base n e n è la base.

Per forza di cose i calcolatori operano con i numeri *in base 2*, le cui cifre vengono chiamate**bit**, esempio 101101 (base 2). Per convertire un numero dalla base 10 alle base bisogna dividerlo in potenze di due

Example 1.1.2

$$37\,(\text{base } 10) = 32 + 4 + 1 = 1 \cdot 2^5 + 0 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 100101\,(\text{base } 2)$$

In generale occorre dividerlo per due finchè non si riduce ad uno ad esempio:

$$(35)_{10} = (100011)_2$$

$$\begin{aligned} 35 : 2 &= 17 \quad \text{resto } 1 \\ 17 : 2 &= 8 \quad \text{resto } 1 \\ 8 : 2 &= 4 \quad \text{resto } 0 \\ 4 : 2 &= 2 \quad \text{resto } 0 \\ 2 : 2 &= 1 \quad \text{resto } 0 \\ 1 : 2 &= 0 \quad \text{resto } 1 \end{aligned}$$

1.1.1 Addizione e moltiplicazione in binario

1.1.2 Conversione fra basi

Chapter 2

Floating point

2.1 Cosa sono i floating point?

Dato che in informatica si lavora con calcolatori con memoria finita, non è possibile implementare il concetto di infinito (a differenza della matematica). Quindi non si è in grado di rappresentare tutto l'insieme dei reali, pertanto occorre approssimarlo. Il metodo che oggi è comune per il calcolo scientifico è noto come sistema **floating point** che permette di rappresentare un ampio intervallo della retta reale con una distribuzione uniforme degli errori.

L'idea di base e' quella di scrivere tutti i numeri come prodotto fra un indicatore s per il segno, un valore decimale $f < 1$ e la base elevata da un esponente intero β^e:

$$\forall n \in \mathbb{R}.n = (s)f\beta^e$$

Example 2.1.1

E' lo stesso procedimento che si usa in base 10 in notazione scientifica:

$$0,0000003467 = 0,3467 * 10^{-6}$$

$$-987380000000 = -0,98738 * 10^{12}$$

2.1.1 Definizione formale

Definition 2.1.1: Numeri macchina

Si definisce **inseime dei numeri numeri macchina** (floating point) con t cifre significative, con base β, con $p \in [L, U]$ e con le cifre d_i dove $0 \leqslant d_i \leqslant \beta - 1$, $i = 1, 2, \dots$ e $d_1 \neq 0$

$$\mathbb{F}(\beta, t, L, U) = \{0\} \cup \{x \in \mathbb{R} = \text{sign}(x)\beta^p \sum_{i=1}^{t} d_i\beta^{-i}\}$$

Usualmente U è positivo e L negativo

Questo insieme è, quindi, un'approssimazione all'intervallo preciso di numeri $[min, max] \subseteq \mathbb{R}$ che dipendono da U e L. Se voglio rappresentare un numero maggiore/minore di questo intervallo avrò un errore chiamato **overflow/underflow**.

2.2 Gap

Con questa notazione non è impossibile scrivere ogni numero presente nella retta dei numeri Reali $\mathbb{R}$ (se ammettiamo che t possa essere infinito); ma, quando lavoriamo con calcolatori che hanno memoria finita, tra due numeri

vi sara' sempre una certa distanza (t finito), quindi è pertanto vero che $\mathbb{F} \subseteq \mathbb{R}$. Questo concetto introduce la definizione di Gap:

Definition 2.2.1: Gap

Chiamiamo **gap** la distanza da un numero al suo sucessivo

È inoltre vero che il gap rimane costante quando l'esponente non cambia, mentre se nell'insieme $\mathbb{F}$ il sucessivo di un numero ha un espente diverso il suo gap sarà diverso di quello del suo precedente con l'esponente uguale. Facciamo un esempio per rendere il tutto più chiaro:

Example 2.2.1

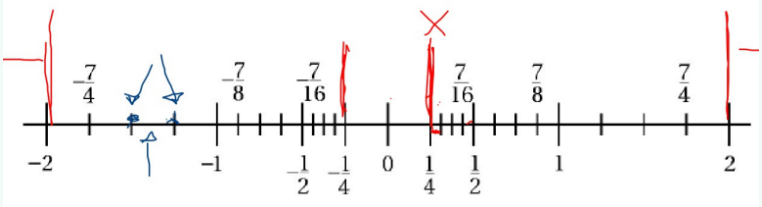
Sia $\beta = 2, t = 3, L = -1, U = 2$ allora

$$\mathbb{F}(2,3,-1,2) = \{0\} \cup \{0.100 \times 2^p, 0.101 \times 2^p, 0.110 \times 2^p, 0.111 \times 2^p, p = -1, 0, 1, 2\}$$

Questo insieme rappresenta 33 nuemri compreso lo 0:

$.100(-1)$	$.101(-1)$	$.110(-1)$	$.111(-1)$
$\frac{1}{4}$	$\frac{5}{16}$	$\frac{6}{16}$	$\frac{7}{16}$
$.100(0)$	$.101(0)$	$.110(0)$	$.111(0)$
$\frac{1}{2}$	$\frac{5}{8}$	$\frac{6}{8}$	$\frac{7}{8}$
$.100(1)$	$.101(1)$	$.110(1)$	$.111(1)$
1	$\frac{5}{4}$	$\frac{6}{4}$	$\frac{7}{4}$
$.100(2)$	$.101(2)$	$.110(2)$	$.111(2)$
2	$\frac{5}{2}$	$\frac{6}{2}$	$\frac{7}{2}$

La rappresentazione di $\mathbb{F}$ nella retta dei numeri reali è questa



Risulta quindi evidente l'aumento dell'ampiezza degli intervalli definiti dal valore p, in ognuno dai quali vengono localizzati le t cifre della mantissa; ne risulta di conseguenza una diradazione delle suddvisioni verso gli estremi sinistro e destro della retta reale.
Il numero t, quindi, fissa la precisione di rappresentazione di ogni numero; infatti la retta viene suddivisa in intervalli $[\beta^p, \beta^{p+1}]$ di ampiezza crescente e in questi intervalli vengono rappresentati lo stesso numero $(\beta - 1)\beta^{t-1}$ di valori, generando ,così, una suddivisione molto densa per valori vincini allo 0 e più rada per valori grandi in valore assoluto. Di conseguenza si ha che **maggiore è il numero t di cifre della mantissa, minore sarà l'ampiezza dei intervalli e quindi migliore l'approssimazione introdotta**.

2.3 floating point nei calcolatori

La rappresentazione più comune del sistema nei calcolatori è definita dallo Standand IEEE 745 il cui insieme è $\mathbb{F}(2, 52, 1023, 1024)$ ed è descritta dalla formula

$$n = (-1)^s \cdot 2^{e-1023} \cdot (1 + f)$$

Dove:

- **s**: è un bit che determina il segno del numero. Se $s = 0$ il numero è positivo, se $s = 1$ il numero è negativo.

- **e**: è l'esponente del numero due, ed è un intero di 11 bit tale che $e \in [0, 2047]$.
- **f**: il valore detto **mantissa** è una serie di 52 bit che rappresenta la parte frazionaria del numero, nel nostro caso $f \in [0, 1]$.
- **1023**: è detto **bias** ed è usato per codificare esponenti negativi e positivi.

In memoria si traduce in un array contentente questi bit:

Rappresentazione binaria dei numeri finiti

Example 2.3.1

Per trovare la codifica del numero in floating point 1 10000000010 1000000000000000000000000000000000000000000000000000 in base 10 occorre:

1. Trovare l'esponente decimale, che è $1 \times 2^{10} + 1 \times 10^1 - 1023 = 3$.
2. Trovare la parte frazionaria, che è $1 \times \frac{1}{2^1} = 0,5$.

Quindi $n = (-1)^1 \times 2^3 \times (1 + 0,5) = -12,0$.

2.3.1 troncamento

Sia l'insieme $\mathbb{F}(\beta, t, L, U) = \{\{\emptyset\} \cup n = \pm\beta^p \cdot d_0 d_1 \dots d_t\}$ con $-L \leqslant p \leqslant U$ l'insieme dei numeri floating point caratterizzati dai valori β, t, L, U
Inoltre sia $x \in \mathbb{R}$ dove $x = \pm\beta^p \cdot d_0 d_1 \dots d_t, d_{t+1}, \dots$ con $L \leqslant p \leqslant U$, nel caso in cui $x \notin \mathbb{F}$ rappresento x con un elemento di $\mathbb{F}$ che indico con $fl(x) = \pm\beta^p \cdot d_0 d_1 \dots d_t$. Questa operzione si chiama **troncamento**

2.3.2 Errore di arrotondamento

Essendo un'approssimazione, il sistema floating point non puo' rappresentare perfettamente tutti i valori reali, in quanto i numeri infiniti (come il π) vengono delineati con una serie finita di numeri finiti. La differenza tra la sua approssimazione e il suo valore reale è detta **errore di arrotondamento**. Sia $x \in \mathbb{R}$ un numero reale e sia $fl(x)$ (dove $fl()$ è la funzione arrotondamento) il numero x arrotondato, allora l'errore di arrotondamento puo' essere espresso come:

- L'errore **assoluto**, che e' piu' grande piu' e' grande l'ordine di $x \in \mathbb{R}$

$$|fl(x) - x| < \beta^{p-t}$$

- E quello **relativo**, che dipende solo da t

$$\frac{|fl(x) - x|}{|x|} < \frac{1}{2}\beta^{1-t}$$

La quantità $esp = \frac{1}{2}\beta^{1-t}$ è detta **precisione macchina** nel sistema floating point, ed è il più piccolo numero macchina positivo tale che:

$$fl(1 + esp) > 1$$

2.3.3 artimetica floating point

Il calcolatore esegue operazioni solo con numeri rappresentabili da calcolatore stesso, quindi in questo caso dai numeri scritti in forma floating ponti (per i nuemri reali), il punto è che le usuali operazioni aritmetiche non sono chiuse nell'insieme $\mathbb{F}$, quindi presi 2 numeri dall'insieme $\mathbb{F}$ e computati tramite operazioni aritmetiche, il risultato di tale computazione potrebbe appartenere nell'insieme $\mathbb{R}$. Pertanto i risultati delle varie computazioni dovranno essere sempre arrotondati ad un numero che stia in $\mathbb{F}$. Definiamo due operazioni generiche, una per i reali e l'altra per i floating-point:

- $\cdot : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$
- $\circ : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$

$$x \circ y = fl(x \cdot y)$$

L'operazione floating point nei calcolatori viene strutturata in due fase:

1. **Eseguo l'operazione esatta**: (es. $z = x + y$) dove viene fatta utilizzando più bit utilizzati di quelli utilizzati per memorizzare il risultato, non è accessibile all'utente ma viene memorizzato in un registro interno alla cpu
2. **Trocamento del risultato**: (es. $x \oplus y = fl(z)$), dove una volta che il calcolo è stato fatto in precisione estesa il risultato viene arrotondato alla precisione del risultato

Example 2.3.2

Esempio: $x = 2.15 \times 10^{12}, y = 1.25 \times 10^{-5}$

$z = x - y$

Viene calcolata come:

$x = 2.15 \times 10^{12}$

$y = 0.000000000000000125 \times 10^{12}$

$x - y = 2.149999999999999999 \times 10^{12}$

arrotondato a 2.15×10^{12}

Se un numero $x \in \mathbb{R}$ ha un numero di cifre decimali maggiore di t, allora $x \notin \mathcal{F}$ e deve essere approssimato (solitamente col troncamento). Percio' in generale le operazioni floating-point generano un errore (di **arrotondamento**):

$$\left| \frac{(x \circ y) - (x \cdot y)}{x \cdot y} \right| < \text{eps}$$

Nel caso di somma di numeri di grandezza molto diversa, le cifre piu' significative del numero piu' piccolo possono essere perse. Per una sola operazione questo non causa un errore troppo grande, ma se l'operazione viene eseguita molte volte l'errore si **amplifica** e puo' diventare catastrofico.

Chapter 3

Risoluzione di equazioni (generale)

3.1 Bisezione

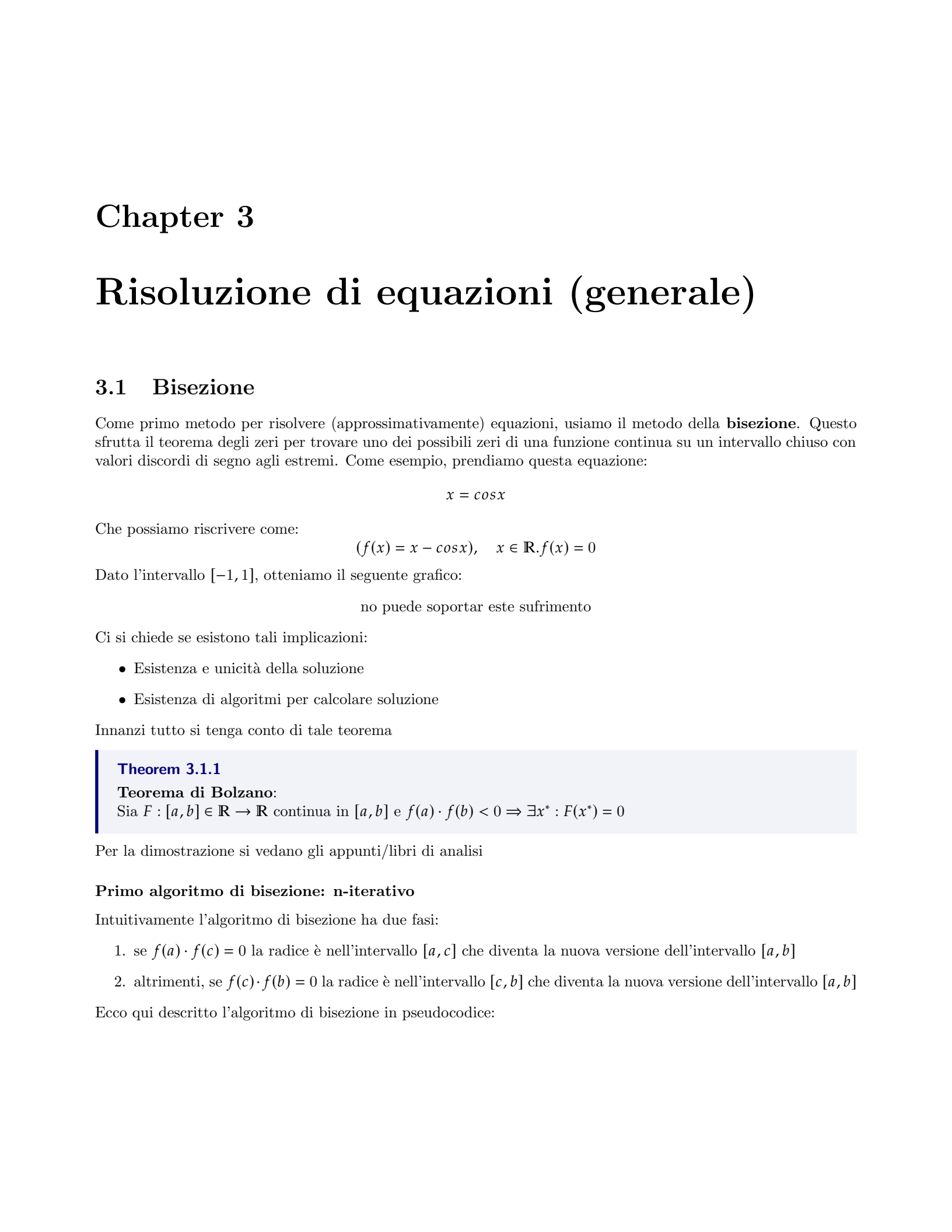
Come primo metodo per risolvere (approssimativamente) equazioni, usiamo il metodo della **bisezione**. Questo sfrutta il teorema degli zeri per trovare uno dei possibili zeri di una funzione continua su un intervallo chiuso con valori discordi di segno agli estremi. Come esempio, prendiamo questa equazione:

$$x = cosx$$

Che possiamo riscrivere come:

$$(f(x) = x - cosx), \quad x \in \mathbb{R}.f(x) = 0$$

Dato l'intervallo $[-1, 1]$, otteniamo il seguente grafico:



Ci si chiede se esistono tali implicazioni:

- Esistenza e unicità della soluzione
- Esistenza di algoritmi per calcolare soluzione

Innanzi tutto si tenga conto di tale teorema

> **Theorem 3.1.1**
>
> **Teorema di Bolzano**:
> Sia $F : [a, b] \in \mathbb{R} \to \mathbb{R}$ continua in $[a, b]$ e $f(a) \cdot f(b) < 0 \Rightarrow \exists x^* : F(x^*) = 0$

Per la dimostrazione si vedano gli appunti/libri di analisi

Primo algoritmo di bisezione: n-iterativo

Intuitivamente l'algoritmo di bisezione ha due fasi:

1. se $f(a) \cdot f(c) = 0$ la radice è nell'intervallo $[a, c]$ che diventa la nuova versione dell'intervallo $[a, b]$
2. altrimenti, se $f(c) \cdot f(b) = 0$ la radice è nell'intervallo $[c, b]$ che diventa la nuova versione dell'intervallo $[a, b]$

Ecco qui descritto l'algoritmo di bisezione in pseudocodice:

Algorithm 1: primo algoritmo di Bisezione (Function f, int a, int b, int n)

Data: Funzione continua $f(x)$, continua in a e b con $f(a) \cdot f(b) < 0$, un numero N di iterazioni
Result: Approssimazione della radice

```
1 for 1 to n do
2     c ← (a+b)/2;
3     if f(c) = 0 then
4         return c ;                                   // Trovata la radice esatta
5     if f(a) · f(c) < 0 then
6         b ← c ;                                      // La radice è nell'intervallo [a,c]
7     else
8         a ← c ;                                      // La radice è nell'intervallo [c,b]
9 return c;                                            // Approssimazione della radice
```

Questo tipo di soluzione per la bisezione ha un numero n di iterazioni per approssimare la soluzione, questa soluzione è molto semplice ma non la migliore. Vi è tuttavia una soluzione migliore

Secondo algoritmo di bisezione: tolleranza d'errore

Nella pratica è molto più utile definire una costante ϵ che vuole rappresentare l'errore di approssimazione massimo, di modo che un'approssimazione $\overline{r}$ della radice reale r grantisca che $|r - \overline{r}| \leqslant \epsilon$. In altre parole si vuole garantire che $r \in [\overline{r} - \epsilon, \overline{r} + \epsilon]$.
Nel nostro caso $\overline{r}$ è il valore di $c = (b-a)/2$ mentre impostiamo $\epsilon = (b-a)/2$. Si imposti inoltre un **errore di tolleranza** e_{tol} per iterare le operzione fino a che non si incontri. Di seguito è riportato l'algoritmo:

Algorithm 2: Algoritmo di Bisezione

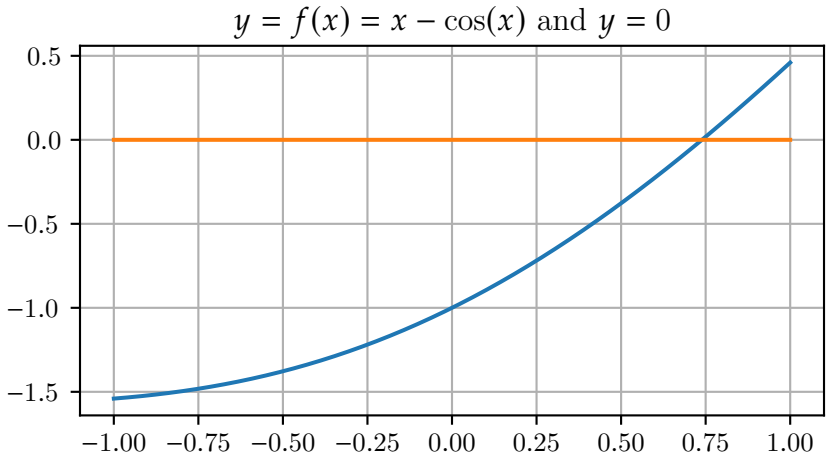
Data: Funzione continua $f(x)$, estremi a e b con $f(a) \cdot f(b) < 0$, tolleranza ϵ
Result: Approssimazione della radice x^* con $|f(x^*)| < Toccalarivoluzioneperlesighedelchea2euro\epsilon$

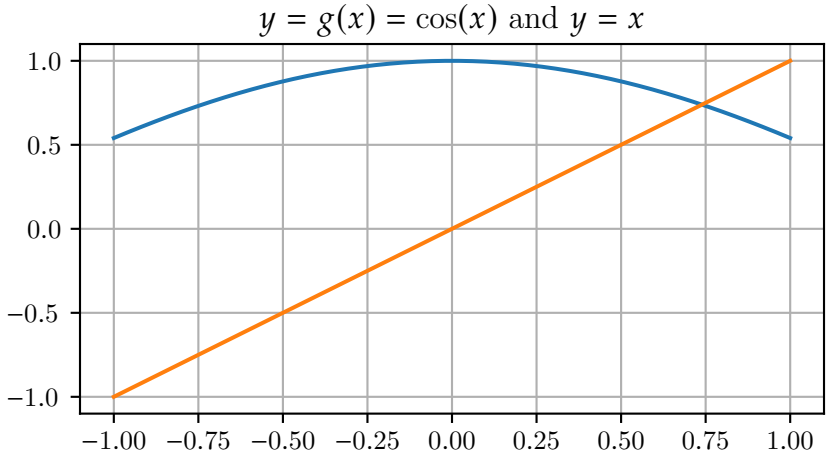
```
1 while |b − a| > ϵ do
2     c ← (a+b)/2; if f(c) = 0 then
3         return c;
          ;                                            // Trovata la radice esatta
4     if f(a) · f(c) < 0 then
5         b ← c;
          ;                                            // La radice è nell'intervallo [a,c]
6     else
7         a ← c;
          ;                                            // La radice è nell'intervallo [c,b]
8 return (a+b)/2;
  ;                                                    // Approssimazione della radice
```

3.2 Iterazione di punto fisso

Presa la funzione f di cui vogliamo trovare lo zero, possiamo definire una funzione $g(x) = x - w(x)f(x)$ dove w e' una funzione peso *positiva* e *limitata*. Quindi:

$$f(x) = 0 \iff g(x) = x$$





Quindi siamo passati dal cercare uno zero di una funzione a trovare il **punto fisso**:

Definition 3.2.1: Punto fisso

Data $g : D \to C$, si chiamano **punti fissi** tutti i punti $x \in D$:

$$g(x) = x$$

Per trovare il punto fisso, bisogna quasi sempre usare un metodo *iterativo*, ovvero di approssimazioni successive, seguendo questa procedura:

- Iniziare con una prima approssimazione x_0
- Iterare con la seguente formula: $x_{k+1} = g(x_k)$

Proposition 3.2.1

Data una funzione $g : D \to C$ continua e una successione $x_n \in D . x_{k+1} = g(x_k)$ tale che $x_n \xrightarrow{n \to +\infty} p$, allora:

$$g(p) = p$$

Ovvero p e' un **punto fisso** di g.

Dimostrazione: Per proprieta' delle successioni e per ipotesi, sappiamo che $\lim_{k \to +\infty} x_k = p$. Quindi, per continuita' di g:

$$\lim_{k \to +\infty} g(x_k) = g(p)$$

Inoltre, per costruzione della successione x_n, sappiamo che $g(x_k) = x_{k+1}$, quindi:

$$\lim_{k\to+\infty} x_{k+1} = \lim_{k\to+\infty} g(x_k)$$

$$p = g(p)$$

😄

3.2.1 Mappe contrattive

Ma come facciamo ad assicurarci che la successione converga? Ecco che entrano in gioco le **mappe contrattive**.

Definition 3.2.2: Mappa

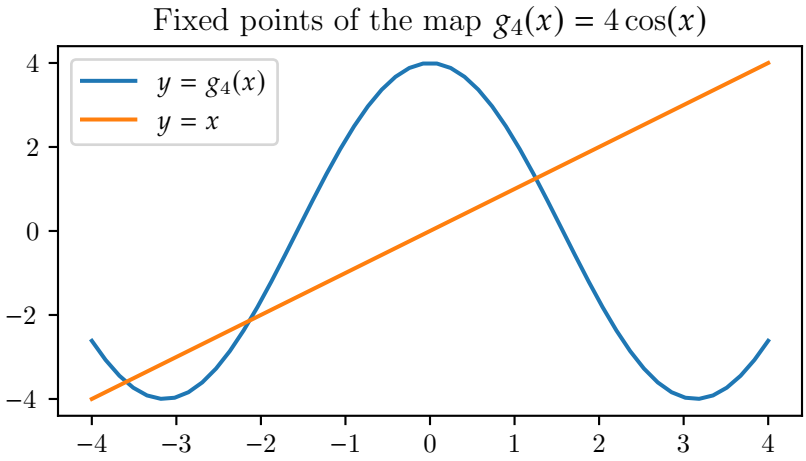
Una **mappa** e' una funzione $g : D \to D$ dove D e' un intervallo chiuso $[a, b]$.

Proposition 3.2.2

Presa una mappa continua su un intervallo $[a, b]$, allora:

$$\exists p \in [a, b].p \text{ e' un punto fisso}$$

Dimostrazione: Consideriamo la funzione $f(x) = x - g(x)$, dove g e' una mappa continua sull'intervallo $[a, b]$. Quindi $f(a) = a - g(a) \leqslant 0$ e $f(b) = b - g(b) \geqslant 0$, dato che per definizione di mappa $g(a), g(b) \in [a, b]$. Quindi per teorema degli zeri $\exists p \in [a, b].f(p) = 0$. Quindi $g(p) = p$ e' un punto fisso. 😄



Come si vede dal grafico della mappa continua $g(x)$ su $[-4, 4]$, e' possibile che una mappa abbia piu' di un punto fisso. Per assicurarci l'esistenza di una soluzione unica, dobbiamo aggiugere un'ultima condizione:

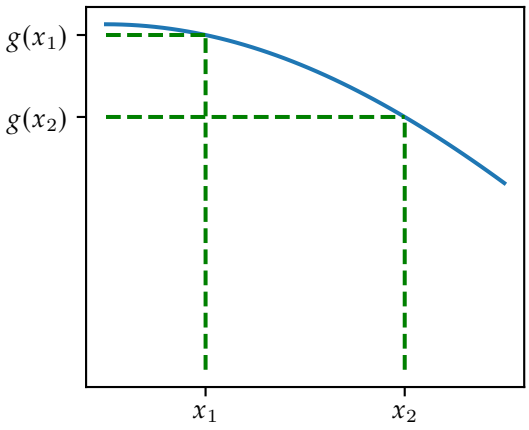
Definition 3.2.3: Mappa contrattiva

Una mappa $g : D \to D$ si dice **contrattiva** se esiste $C < 1$ tale che:

$$\forall x, y \in D.|g(x) - g(y)| \leqslant C|x - y|$$

C e' detta costante contrattiva

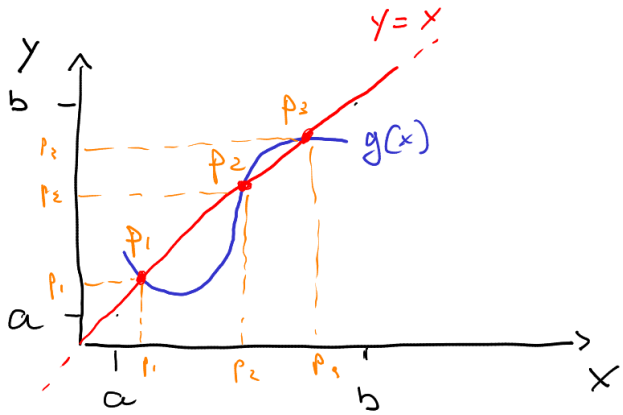
Questo significa che per ogni due punti che prendiamo dal dominio, la loro distanza sara' sempre maggiore (o uguale) alla distanza fra i due punti corrispondenti nel codominio. Graficamente (in $\mathbb{R}^2$):



Theorem 3.2.1

Data una mappa contrattiva g su un intervallo chiuso D, esiste un solo punto fisso p a cui converge la successione x_n dove $x_{k+1} = g(x_k)$, scegliendo x_0 come qualunque punto in D.

Dimostrazione (singolo punto fisso): Assumiamo che g abbia piu' di un punto fisso e dimostriamo l'assurdo. Consideriamo il seguente grafico:



Prendendo due dei punti fissi, p_1, p_2, notiamo che $\frac{|g(p_1)-g(p_2)|}{|p_1-p_2|} = \frac{|p_1-p_2|}{|p_1-p_2|} = 1$, ma dato che la costante C deve essere minore di 1, abbiamo dimostrato l'assurdo. 😄

Manca la dimostrazione della garanzia dell'esistenza del punto fisso, che vedremo piu' avanti. (3.2.2)

Come controllare se una funzione differenziabile e' una mappa contrattiva

Nella dimostrazione precedente e' apparsa una forma che somigliava molto a una derivata. Infatti, se la funzione g e' derivabile possiamo usare il seguente teorema:

Theorem 3.2.2

Se una funzione $g : [a,b] \to [a,b]$ e' differenziabile e esiste una costante $C < 1$ tale che $\forall x \in [a,b].|g'(x)| < C$, allora g e' una **mappa contrattiva**

Dimostrazione: Per il teorema di Lagrange, $\forall x < y \in [a,b].\exists c \in [x,y].g(y) - g(x) = g'(c)(y-x)$. Quindi, aggiungendo valori assoluti, $|g(y) - g(x)| = |g'(c)| \cdot |y - x| \leqslant C \cdot |y - x|$ per ipotesi. Quindi, per definizione, g e' una mappa contrattiva. 😄

3.2.2 Velocita' del miglioramento dell'approssimazione

Chiamiamo l'errore assoluto $|E_k|$ la differenza assoluta fra x_k e il valore effettivo del punto fisso p.

Proposition 3.2.3

$|E_{k+1}| \leqslant C|E_k|$, quindi

$$|E_k| \leqslant C^k|E_0|$$

Dove $E_0 = x_0 - p$.

Quindi per ogni iterazione, l'errore iniziale diminuisce almeno di un fattore C (che ricordo e' sempre < 1 quindi e' una cosa buona). Cio' implica che piu' e' piccolo C, piu' ci avviciniamo in fretta a p.

Dimostrazione: Per definizione di errore $E_{k+1} = x_{k+1} - p$, che possiamo riscrivere come $g(x_k) - g(p)$, quindi per definizione di mappa contrattiva $|E_{k+1}| = |g(x_k) - g(p)| \leqslant C|x_k - p| = C|E_k|$. Se sostituiamo $k = 0$ abbiamo il caso base:

$$|E_1| \leqslant C|x_0 - p|$$

Da cui ricorsivamente:

$$|E_k| \leqslant C|E_{k-1}| \leqslant C \cdot C|E_{k-2}| \leqslant ... \leqslant C \cdot ... \cdot C|E_0| = C^k|x_0 - p|$$

😄

Ora siamo in grado di finire la dimostrazione di prima:

Diostrazione (convergenza): Dato che $C < 1$, se $k \to +\infty$ allora $C^k \to 0$, quindi $\lim_{k\to+\infty} |E_k| = \lim_{k\to+\infty} C^k|x_0 - p| = 0$. Allora:

$$x_k \xrightarrow{k\to+\infty} p$$

😄

3.3 Metodo di Newton

Vediamo ora il metodo di Newton, che rispetto alla bisezione e' generalmente piu' veloce (ma non abbiamo la garanzia). Puo' essere dimostrato ricorrendo alle mappe contrattive o come approssimazione a linee tangenti, vediamo la prima:

Prendiamo una funzione differenziabile f e costruiamole una mappa contrattiva g. Per migliorare l'efficenza, vogliamo fare in modo che la costante C abbia valore minimo dato un intorno della radice r di f, come facciamo? Dato che C assume il valore della derivata massima della mappa, vogliamo fare in modo che la derivata di g sia minima per quell'intorno. Un modo per fare cio' e' assicurarci che $g'(r) = 0$, vediamo come:

$$g'(r) = 1 - w'(r)f(r) - w(r)f'(r) = 1 - w(r)f'(r)$$

dato che $f(r) = 0$ per definizione, quindi:

$$w(r) = \frac{1}{f'(r)}$$

Dato che non sappiamo il valore di r, poniamo $w(x) = \frac{1}{f'(x)}$ per ogni x nel dominio. Da notare che $w(x)$ non esiste quando $f'(x) = 0$. Facendo cosi', la formula di iterazione diventa:

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

che e' la formula per il metodo di Newton.

Chapter 4

Un po' di algebra lineare e la SVD

4.1 Autovalori e autovettori

Definition 4.1.1

Data una matrice A di dimensione $n \times n$ se vale:

$$Ax = \lambda x$$

con $\lambda \in \mathbb{R}$ e $x \in \mathbb{R}^n, \lambda$ si dice autovalore di A e x autovettore di A associato a λ

Data questa definizione, e richiamando i vari concetti dell'algebra lineare, abbiamo che se vale $Av_i = \lambda_i v_i, \forall i \in 1, \ldots, n$ allora vale anche:

$$[Av_1, Av_2, \ldots, Av_n] = [\lambda_1 v_1, \lambda_2 v_2, \ldots, \lambda_n v_n]$$

$$\text{quindi:} \quad A[v_1, v_2, \ldots, v_n] = [v_1, v_2, \ldots, v_n] \begin{pmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{pmatrix}$$

Questa bellissima scrittura si può tranquillamente riassumere come:

$$AV = VD$$

4.1.1 Decomposizione agli autovalori

Definition 4.1.2

sia $A \in M^{n \times n}$, allora una Una decomposizione agli autovalori di A è una fattorizzazione (ovvero un prodtto) di questo tipo:

$$A = VDV^{-1}$$

Dove $V = [v_1, v_2, \ldots, v_n]$ è la matrice con colonne gli autovettori v_1 di A e D è una matrice diagonale che ha sulla diagonale gli autovalori di A associati alle colonne di V

Se A ha una decomposizione agli autovalori allora si dice che A è **diagonalizzabile**; in caso contrario A si dice **non diagonalizzabile** (o defettiva).

Se A ha n autivalori distinti, allora A è diagonalizzabile

4.2 Vettori ortogonali e ortonormali

Definition 4.2.1

Siano $v_1, v_2, \ldots, v_m$ vettori di R^n, si dicono **ortognonali** se:

$$v_i^T v_j = 0, \quad \text{se } i \neq j$$

Si può scrivere anche:

$$\langle v_i, v_j \rangle = 0, \quad \text{se } i \neq j$$

Ovvero il prodotto scalare

Questa relazione di vettori si collega molto bene con i vettori ortonormali:

Definition 4.2.2

Siano $v_1, v_2, \ldots, v_m$ vettori di R^n, si dicono **ortonormali** se sono ortogonali w di lunghezza unitaria, ovvero $v_i^T v_j = \|v_i\| = \delta_{ij}$ dove

$$\delta_{ij} = \begin{cases} 1 & \text{se } i = j \\ 0 & \text{se } i \neq j \end{cases}$$

4.2.1 Matrici ortogonali

Definition 4.2.3

Una matrice W di dimensione $n \times n$ si dice ortogonale se le sue colonne sono vettori ortonormali

Un esempio semplice:

Example 4.2.1

$$W = \begin{pmatrix} \frac{1}{\sqrt{3}} & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & 1 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & 0 & 0 \end{pmatrix}$$

le matrici ortogonali hanno diverse proprietà:

1. $W^T = W^{-1}$
2. $\forall x \in R^n, \|x\|_2 = \|Wx\|_2$

4.3 SVD

Note:

Fonti:

- Visualizzazione geometrica
- Visualizzazione vettori ortogonali e dimostrazione minimo quadrato
- Dimostrazione SVD

Prima di iniziare ad introdurre metodi pratici per l'approssimazione dei dati, e' necessario esplorare uno dei teoremi finali dell'algebra lineare: la **SVD** (Singular Value Decomposition). Questo teorema sara' fondamentale per descrivere vari modelli di ottimizzazione e approssimazione per motivi che vedremo dopo.

4.3.1 Introduzione

Un modo per arrivare alla SVD e' porci la seguente domanda: "Data una matrice $A \in M(\mathbb{R})_{n\times m}$, esistono m vettori ortonormali in $\mathbb{R}^m$ la cui immagine e' sempre un insieme di m vettori ortogonali (se $m > n$ allora $m - n$ vettori dell'immagine saranno nulli) in $\mathbb{R}^n$?" Proviamo a scrivere la domanda in modo piu' matematico:

$$\exists V \in M(\mathbb{R})_m, U \in M(\mathbb{R})_n, \Sigma \in M(\mathbb{R})_{n\times m}.V, U \text{ ortonormali, } \Sigma \text{ diagonale (rettangolare)}.\forall A \in M(\mathbb{R})_{n\times m} :$$

$$AV = U\Sigma$$

dove V ha come colonne i vettori ortonormali del dominio, U ha come colonne vettori ortonormali (se $n > m$ le ultime $n - m$ colonne non importano e possono essere nulle) che moltiplicate per i corrispondenti scalari sulla diagonale della matrice Σ formano l'immagine ortogonale di V. Dato che V e' ortonormale, posso riscrivere l'equazione come scomposizione di A:

$$A = U\Sigma V^T$$

Questa e' la scomposizione di cui la SVD ci assicura l'esistenza per qualsiasi matrice A, formalmente:

Theorem 4.3.1 Singular Value Decomposition

Per qualsiasi $A \in M(\mathbb{R})_{n\times m}$, esistono:

- U ortonormale $n \times n$
- V ortonormale $m \times m$
- Σ rettangolare diagonale $n \times m$

Tali che:

$$A = U\Sigma V^T, \quad \Sigma = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{pmatrix}$$

Prima di poter dimostrare l'esistenza della SVD per ogni matrice, ci serve un lemma:

Lenma 4.3.1 Lemma 1

Per una qualsiasi matrice $A \in M(\mathbb{R})_{n\times m}$, si ha che:

- $A^TA \in M(\mathbb{R})_n$ semidefinita positiva
- $AA^T \in (\mathbb{R})_m$ semidefinita positiva

: Dimostra (forse) 😄

Dimostrazione SVD: . Sia $A \in M(\mathbb{R})_{n\times m}$. Per Lemma 1, sappiamo che A^TA e' una matrice semidefinita positiva, quindi, per il teorema spettrale, puo' essere scomposta come:

$$A^TA = V\Lambda V^T$$

dove $V \in M(\mathbb{R})_m$ e' una matrice ortonormale e $\Lambda \in M(\mathbb{R})_m$ e' la matrice diagonale di autovettori ordinati $\lambda_1 \geqslant ... \geqslant \lambda_m$ tali che $Av_1 = \lambda_1 v_1$. Definiamo ora n vettori u_i nel seguente modo, dove $\sigma_i = \sqrt{\lambda_i}$:

$$u_i = \begin{cases} \frac{Av_i}{\sigma_i} & \lambda_i \neq 0 \\ \underline{0} & \lambda_i = 0 \end{cases}$$

Dato che A^TA ha lo stesso rango di A (dim. nella fonte), se A ha rango massimo allora $\forall i.\sigma_i > 0$ (perche' si puo' dimostrare che A^TA diventa **definita** positiva); se $r = rank(A) < min(n, m)$, allora gli ultimi $min(n, m) - r$ valori

singolari saranno nulli. (Da riguardare perche non sono sicurissimo) Si puo' dimostrare che u_i sono autovettori di AA^T:

$$\begin{aligned} AA^T u_i &= AA^T \frac{Av_i}{\sigma_i} \\ &= A\frac{\sigma_i^2 v_i}{\sigma_i} \\ &= \sigma^2 \frac{Av_i}{\sigma_i} \\ &= \sigma^2 u_i \end{aligned}$$

E che sono unitari:

$$\begin{aligned} u_i^T u_i &= \frac{v_i^T A^T}{\sigma_i}\frac{Av_i}{\sigma_i} \\ &= \frac{v_i^T \sigma^2 v_i}{\sigma^2} \\ &= v_i^T v_i = 1 \end{aligned}$$

Essendo autovettori unitari di una matrice semidefinita positiva (AA^T), la matrice $U \in M(\mathbb{R})_n$ che ha come colonne u_i e' ortonormale. Usando le matrici possiamo scrivere:

$$Av_i = \sigma_i u_i \implies AV = U\Sigma$$

dove $\Sigma \in M(\mathbb{R})_{n\times m}$ ha sulla diagonale $min(n,m)$ valori singolari $(\sigma_1, ..., \sigma_{min(n,m)})$ e il resto e' 0.

$$A = U\Sigma V^T$$

😊

4.3.2 Valori Singolari

- Gli elementi $\sigma_1 \geqslant \sigma_2 \geqslant \dots \sigma_n \geqslant 0$ sono i cosidetti **valori singolari di** A
- Mentre il rango k è uguale al numero di valori singolari positivi, cioè $k = rango(A) \iff (\sigma_1 \geqslant \sigma_2 \geqslant \dots \sigma_k > 0 \wedge \sigma_{k+1} = \dots = \sigma_n = 0)$
- Le colonne si $U = (u_1, u_2, \dots, u_m)$ e di $V = (v_1, v_2, \dots, v_n)$ sono, rispettivamente, i **vettori singolari sinistri e destri di** A associati ai valori singolari σ_i e formano una base ortonormale di R^m e R^n , rispettivamente, poichè vale la relazione:
$$Au_i = \sigma_i v_i$$

Questo teorema porta a un'implicazione piuttosto interessante, inffatti si ha:

$$A^T A = (U\Sigma V^T)^T (U\Sigma V^T) = V\Sigma^T U^T U\Sigma V^T$$

dato che U è ortogonale si "annulla" con la sua trasposta, quindi continua con

$$V\Sigma^T\Sigma V^T = V\begin{pmatrix} \sigma_1^2 & 0 & \cdots & 0 \\ 0 & \sigma_2^2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_n^2 \end{pmatrix} V^T$$

Inoltre:

$$\sigma_i = \sqrt{\lambda_i}, i = \dots, n$$

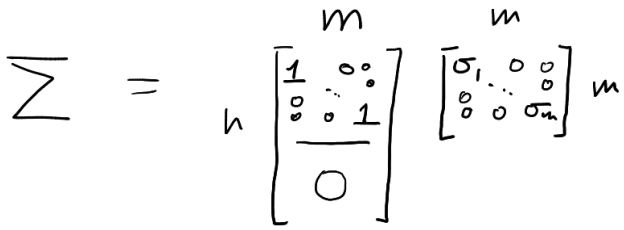
Dove λ_i non è che l'autovalore di $A^T A$, si ha quindi:

- $\sigma_1 = \sqrt{\lambda_{max}} = \sqrt{\rho} = \|A\|$
- $\sigma_k = \sqrt{\lambda_{min}} \Rightarrow \|A^{-1}\| = \frac{1}{\sigma_k}$

4.3.3 Interpretazione geometrica

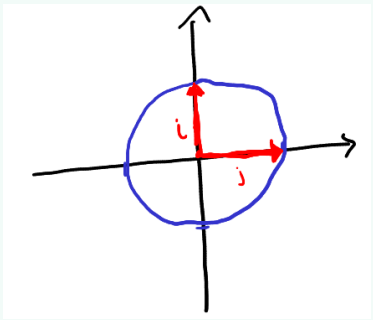
Questa scomposizione puo' essere interpretata in modo geometrico. Immaginiamo di moltiplicare un vettore x per la scomposizione di A:

- La matrice V^T, essendo ortonormale, applica una rotazione a x senza variarne il modulo. Essendo trasposta, che per le matrici ortonormali corrisponde con l'inversa, al posto di spostare i vettori base ai vettori di V fa il contrario.
- La matrice Σ puo' essere scomposta in due parti:
  - La matrice $m \times m$ diagonale, che ha come valori diagonali gli m valori singolari. Dato che e' diagonale, applica solo un'allungamento/restrizione di x ruotato.
  - Una matrice $n \times m$ rettangolare diagonale con solo valore 1. Serve per portare il vettore nel nuovo spazio vettoriale.



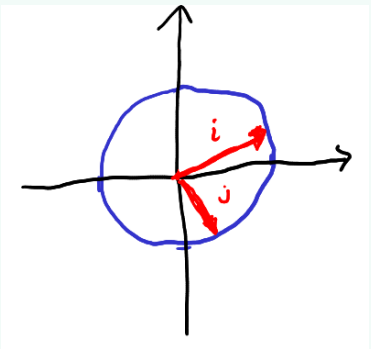
- Infine la matrice U ortonormale ruota il vettore nello spazio del codominio $\mathbb{R}^n$, spostando la base sui vettori di U.

Example 4.3.1 (Mappa da sfere a elissoidi)
Consideriamo una cironferenza unitaria in $\mathbb{R}^2$:

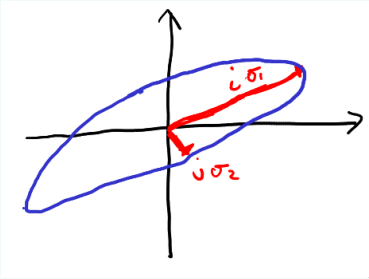


Applichiamo la trasformazione lineare definita dalla matrice $A \in M(\mathbb{R})_{3\times2}$. Come abbiamo visto, qualunque matrice puo' essere scomposta come due matrici di rotazione e una di "allungamento". Vediamo i vari passaggi:

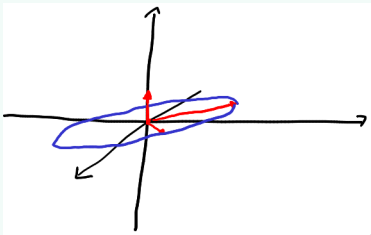
- Applicando la prima rotazione, la nostra figura non cambia dato che e' una circonferenza:



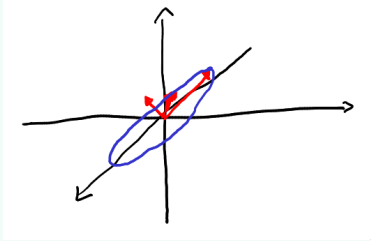
- Moltiplicando per la matrice diagonale, la circonferenza viene allungata lungo due assi principali (i vettori della base canonica che sono stati roteati da V^T). Abbiamo quindi un'ellisse:



- Spostiamo l'ellisse in $\mathbb{R}^3$:



- Manca solo l'ultima rotazione applicata da U, questa volta in 3 dimensioni:



Per questo motivo, possiamo dire che le matrici $n \times m$ trasformano sfere in $\mathbb{R}^m$ in elissoidi a n dimensioni.

Quindi i **valori singolari** $\sigma_1, ..., \sigma_{min(n,m)}$ sono i coefficenti scalari che moltiplicano i vettori lungo $min(n, m)$ assi ortogonali, e solo $rk(A) = k$ di questi saranno non nulli, quindi le restanti assi si azzerano.

4.3.4 Applicazioni

La SVD e' uno strumento utilizzato moltissimo per problemi di ottimizzazione e di aprossimazione, in quanto e' applicabile universalmente e fornisce informazioni molto importanti per l'analisi dei dati. La lista ordinata in modo discendente rispetto a quanto la matrice "allunga" i vettori lungo una certa direzione (vettori singolari destri e sinistri ???) e' utilizzata per la PCA (Principal Component Analisis ???) e per l'aprossimazione di matrici. Inoltre, la SVD e' utilizzata per risolvere il problema dei minimi (o massimi) quadrati, come vedremo prossimamente.

4.4 Numero di condizione

Definition 4.4.1: Numero di condizione

Si dice **numero di condizione di** A tale valore:

$$K_2(A) = \frac{\sigma_1}{\sigma_k}$$

Dove σ_k è il valore singolare minimo di A e k è il rango della matrice, quindi $k \leqslant \min(m, n)$
Un'altra definizione di questo valore è:

$$K_2(A) = \|A\| \cdot \|A^{-1}\|$$

Un numero di condizionamento basso (vicino a 1) indica che la matrice è ben condizionata. Ciò significa che piccole variazioni negli input producono piccole variazioni negli output, viceversa un numero di condizionamento alto indica che la matrice è mal condizionata. Ciò significa che piccole variazioni negli input possono causare grandi variazioni negli output

Chapter 5

Sistemi lineari

5.1 Algoritmi per il calcolo di sistemi lineari

$$\begin{cases} a_{21}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_2 \\ a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ \vdots \\ a_{n1}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_n \end{cases}$$

Con m equazioni ed n incognite. E sia $m = n$, corrispondente ad un sistema quadrato
Sia $Ax = b$ con

$$A = \begin{pmatrix} a_{11} & \dots & a_{1n} \\ a_{21} & \dots & a_{2n} \\ \vdots & \vdots & \vdots \\ a_{n1} & \dots & a_{nn} \end{pmatrix}$$

e

$$x \in \mathbb{R}^n = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, b \in \mathbb{R}^n = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}$$

Allora su ha che il sistema $Ax = b$ ha una ed una sola soluzione sse A non è singolare ($detA \neq 0$)

$$Ax = b \iff A^{-1}Ax = A^{-1}b \iff x = A^{-1}b$$

Dove il Calcolo di A^{-1} complessità computazionale $O(n^3)$
A questo proposito ci vengono in aiuto diversi metodi per il calcolo:

- **Metodi Diretti**: la soluzione viene calcolata in un numero finito di passi modificando la matrice del problema in modo da rendere piú agevole il calcolo della soluzione.

  Es.

  - Matrici triangolari: Metodi di Sostituzione
  - Fattorizzazione LU
  - fattorizzazione di Cholesky,

- **Metodi Iterativi**: Calcolo di una soluzione come limite di una successione di approssimazioni x_k, senza modificare la struttura della matrice A. Adatti per sistemi di grandi dimensioni con matrici sparse (pochi elementi non nulli)

5.1.1 Metodi diretti

Iniziamo con la definizione di matrici triangolari:

Definition 5.1.1: Matrice triangolare inferiore

Una matrice $M \in \mathbb{R}^{n \times n}$ è detta **triangolare inferiore** se:

$$L = \begin{pmatrix} l_{11} & 0 & 0 & \dots & 0 \\ l_{21} & l_{22} & 0 & \dots & 0 \\ l_{31} & l_{32} & l_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{nn} \end{pmatrix}$$

Dove l_{ij} per $i < j$, ossia tutti gli elementi sopra la diagonale (con $j > i$) sono nulli

Da cui discende la definizione di **sistema triangolare inferiore**:

Definition 5.1.2: sistema triangolare inferiore

Un **sistema triangolare inferiore** è un sistema lineare di equazioni in cui la matrice dei coefficienti è una matrice triangolare inferiore, e si presenta nella seguente forma:

$$\begin{pmatrix} a_{1,1} & 0 & 0 & \dots & 0 \\ a_{2,1} & l_{2,2} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \dots & a_{n-1,n-1} & 0 \\ a_{n,1} & a_{n,2} & a_{n,3} & \dots & a_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}$$

e in un sistema lineare:

$$\begin{cases} a_{1,1}x_1 = b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 = b_2 \\ a_{3,1}x_1 + a_{3,2}x_2 + a_{3,3}x_3 = b_3 \\ \vdots \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,n}x_n = b_n \end{cases}$$

Adesso c'è contrapposto con il **sistema triangolare superiore**

Definition 5.1.3: matrice triangolare superiore

Una matrice $M \in \mathbb{R}^{n \times n}$ è detta triangolare superiore se:

$$L = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1,n} \\ 0 & a_{2,2} & a_{2,3} & \cdots & a_{2,n} \\ 0 & 0 & a_{3,3} & \cdots & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{n,n} \end{pmatrix}$$

Dove $u_{ij} = 0$ per $i > j$, ossia tutti gli elementi sotto la diagonale (con $i > j$) sono nulli.

Da cui discende la definizione di **sistema triangolare superiore**

Definition 5.1.4: sistema triangolare superiore

Un **sistema triangolare superiore** è un sistema lineare di equazioni in cui la matrice dei coefficienti è una matrice triangolare superiore, e si presenta nella seguente forma:

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,n-1} & a_{1,n} \\ 0 & a_{2,2} & a_{2,3} & \cdots & a_{2,n-1} & a_{2,n} \\ 0 & 0 & a_{3,3} & \cdots & a_{3,n-1} & a_{3,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & a_{n-1,n-1} & a_{n-1,n} \\ 0 & 0 & 0 & \cdots & 0 & a_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}$$

e in un sistema lineare:

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n = b_1 \\ a_{2,2}x_2 + a_{2,3}x_3 + \cdots + a_{2,n}x_n = b_2 \\ a_{3,3}x_3 + \cdots + a_{3,n}x_n = b_3 \\ \quad \vdots \\ a_{n,n}x_n = b_n \end{cases}$$

algortimo delle sostituzioni

Quando si ha a che fare con matrici triangolari la soluzione può essere facilmente calcolata attraverso **l'algoritmo delle sostituzioni**, che si differenzia nel caso di matrici triangolari inferiori o superiori

Theorem 5.1.1

- Nel caso di matrici triangolari superiori l'algoritmo prende il nome di **sostituzioni all'indietro** ed è descritto dal sistema qui sotto:

$$\begin{cases} x_n = \frac{b_n}{a_{nn}} \\ x_i = \frac{b_i - \sum_{k=i+1}^{n} a_{ik}x_k}{a_{ii}} \quad i = n-1, n-2, \dots 1 \end{cases}$$

- Nel caso di matrici triangolari inferiori l'algoritmo prende il nome di **sostituzioni in avanti** ed è descritto dal sistema qui sotto:

$$\begin{cases} x_1 = \frac{b_1}{a_{11}} \\ x_i = \frac{b_i - \sum_{k=1}^{i-1} a_{ik}x_k}{a_{ii}} \quad i = 2, 3, \dots, n \end{cases}$$

In entrambi i casi il numero di operazioni richiesto è $\simeq \frac{n(n+1)}{n} = O(n^2)$

Eliminazione di Gauss

Si eliminano le incognite in modo sistematico per trasformare i sistema lineare in uno equivalente con matrice a struttura triangolare superiore, più precisamente si introduce una successione di matrici tale che:

$$A^{(k)} = (a_{ij}^{(k)}), \quad k = 1, \dots, n$$

Con $A^{(1)} = A$ e $A^{(n)} = U$ e dove la matrice $A^{(k+1)}$ ha tutti gli elementi $\{a_{ij}^{(k+1)}, j+1 \leqslant i \leqslant n, 1 \leqslant j \leqslant k\}$ nulli. Ovvero essa differisce da $A^{(k)}$ per avere gli elementi sottodiagonali della colonna k-esima nulli. Risparmio ulteriori

spiegazioni di come funziona e passo direttamente alla formula:

$$m_{ik} = \frac{q_{ik}^{(k)}}{q_{kk}^{(k)}}$$
$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ik} a_{kj}^{(k)}$$

Inoltre è possibile che bisogni scambiare le righe, ad esempio quando $det(A_k) = 0$ e pertanto il $a_{kk} = 0$, Perciò occorre scambiarli con altre righe. Si noti che lo scambio di righe i, j di una matrice A equivale a moltiplicarla per una matrice P^{ij} tale che:

$$(P^{(ik)})_{km} = \begin{cases} \delta_{km} & \text{se } k \neq i \text{ e } k \neq j, \\ 0 & \text{se } k = m \text{ e } (k = i \text{ o } k = j), \\ 1 & \text{se } k = i \text{ e } m = j, \\ 1 & \text{se } k = j \text{ e } m = i. \end{cases}$$

La quale verrà chiamata **matrice di permutazione**

algoritmo di gauss rivisto come mettodo di fattorizzazione LU

Si può dimostrare che l'Algoritmo di Gauss realizza la seguente fattorizzazione della matrice A in due matrici triangolari:

$$A = LU = \begin{pmatrix} 1 & 0 & 0 & 0 \\ * & 1 & 0 & 0 \\ * & * & 1 & 0 \\ * & * & * & 1 \end{pmatrix} \cdot \begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & 0 & * \end{pmatrix}$$

Passiamo dunque, direttamente al teorema esplicativo

Theorem 5.1.2

Sia $A \in \mathbb{R}n \times n$ una matrice, è possibile fattorizzarla nel seguente modo:

$$A = LU$$

Dove:

- $U = A^{(n)}$ è una matrice triangolare superiore (ovvero la matrice posto sotto l'eliminazione di gauss)
- L è la matrice triangolare inferiore dei moltiplicatori, ovvero:

$$L = \begin{cases} L_{ij} = m_{ij} & i < J \\ L_{ij} = 1 & i = j \\ L_{ij} = 0 & j > i \end{cases}$$

Mentre se si necessita di uno scambio di righe durante l'eliminazione di gauss la fattorizzazione sarà:

$$LU = PA$$

Dove P è la matrice di permutazione vista prima

Dimostrazione: Sia $M_k = I - m_k e_k^T$, dove:

- $(e_k)_j = \delta_{kj}$ quindi $e_k^T = [0, 0, \ldots, 1, \ldots, 0]$ Dove 1 è alla posizione
- $m_k = \begin{pmatrix} 0 \\ \ldots \\ m_{k+1,k} \\ m_{k+2,k} \\ \ldots \\ m_{nk} \end{pmatrix}$

Che equivale in termini di componenti è uguale a:

$$(M_k)_{ip} = \delta_{ip} - (m_k e_k^T)_{ip} = \delta_{ip} - m_{ik}\delta_{kp}$$

Dalle formule dell'algoritmo di gauss si ha:

$$a_{ij}^{(k+1)} = a_{ij}^{(k)} - m_{ij}^{(k)} a_{ij}^{(k)} = a_{ij}^{(k)} - m_{ik}\delta_{kk} a^{(k)} a_{kj}^{(k)} = \sum_p (\delta_{ip} - m_{ik}\delta_{kp}) a_{pj}^{(k)}$$

e quindi è facile verificare che:

$$A^{(k+1)} = M_k A^{(k)}$$

Si prenda osservi, intatti, che moltiplicare M_k a sinistra di A ha come effetto la sottrazione da ogni riga $R_i, (i = k+1, \dots, n)$ della riga m_{ik}. Allora:

$$U = A^{(n)} = \underbrace{M_{n-1} M_{n-2} \dots, M_1}_{M} A$$

Ovvero:

$$A = M^{-1} U$$

E dato che il prodotto di matrici triangoli e l'inversa di una matrice triangolare sono ancora matrici triangolare, si ha che M^{-1} è triangolare, quindi definiamo $M^{-1} = L$.
Adesso dimostriamo la parte del teorema con la matrice di permutazione, ricordando quindi la formula relativa alla matrice di permutazione si ha:

$$A^{(k+1)} = M_k P_k A^{(k)}$$

Per U quindi si avrà la seguente espressione:

$$U = M_{n-1} P_{n-1} \dots M_1 P_1 A$$

posto

$$P = P_{n-1} \dots P_1 \text{ e } L = P(M_{n-1} P_{n-1} \dots M_1 P_1)^{-1}$$

Si ha allora che L è triangolare inferiore e si ottiene:

$$LU = P(M_{n-1} P_{n-1 \dots M_1 P_1})^{-1} (M_{n-1} P_{n-1} \dots M_1 P_1) A = PA$$

😄

Example 5.1.1

Supponiamo di voler fattorizzare la matrice:

$$A = \begin{pmatrix} 2 & 3 \\ 4 & 7 \end{pmatrix}$$

Poi inizializzo L come una matrice identità e$U = A$

$$L = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad A = \begin{pmatrix} 2 & 3 \\ 4 & 7 \end{pmatrix}$$

Poi eseguo l'algoritmo di Gauss in U avendo come pivot 2, in questo caso occorre solo modificare solo la seconda in quanto la matrice ha solo 2 righe. Il fattore moltiplicativo nella riga 2 e colonna 1 è $l_{2,1} = \frac{4}{2} = 2$, si può procedere così con l'eliminazione Gaussiana ottenendo:

$$U = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

Si inserisca, inoltre, $l_{2,1}$ nella matrice L:

$$L = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

La fattorizzazione, così, è finita e si ha $A = L \times U$:

$$\begin{pmatrix} 2 & 3 \\ 4 & 7 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}$$

A che serve tutto questo? Perché tutto sto ambaradam? Restando calmi e senza farci prendere dalla disperazione dobbiamo prendere atto che il nostro bellissimo potere fattorizzante ci permette di ridurci a risolvere sistemi lineari triangolari per poi risolverli con il potere dell'algoritmo delle sostituzioni (evviva). Si noti il seguente corollario
EOEOE

Corollary 5.1.1 sistemi triangolari

Nel caso in cui non si ricorra ad uno scambio di righe si ha

$$Ax = b \iff LUx = b \iff \begin{cases} Ly = b \\ Ux = y \end{cases}$$

Nel caso in cui si ricorre ad uno scambio di righe si ha:

$$Ax = b \iff PAx = Pb \iff LUx = Pb \iff \begin{cases} Ly = Pb \\ Ux = y \end{cases}$$

Note:

Se A e' invertibile, la fattorizzazione LU esiste sse tutti i minimi principali (leading principal minor??) sono non nulli. Nel caso in cui A e' singolare ($det(A) = 0$), se i primi $k = rk(A)$ minimi principali sono non nulli allora siamo sicuri che la fattorizzazione LU esiste, ma potrebbe esistere anche se ce ne sono di nulli.

Algorithm 3: Fattorizzazione LU

Input: Intero n, Matrice $A[1..n, 1..n]$
Output: Array $B[1, 2]$ contenente le matrici L e U

```
1  Let B be a new Array[2];
2  Let L be a new Matrix[1..n, 1..n];
3  Let U be a new Matrix[1..n, 1..n];
4  for i = 1 to n do
5      for j = 1 to n do
6          if i == j then
7              L[i, j] ← 1;
8          else
9              L[i, j] ← 0;
10 Let p be a new Real;                                   // dichiaro il pivot
   // colonne per pivot
11 for i = 1 to n do
12     p ← A[i, i];                                       // aggiorno il pivot
       // righe
13     for j = i + 1 to n do
14         L[j, i] ← A[j,i] / p;                          // fattore moltiplicativo in L
15         for k = i to n do
16             A[j, k] ← A[j, k] − L[j, i] × A[i, k];
17 B[1] ← L;
18 B[2] ← A;
19 return B;
```

5.1.2 Fattorizzazione di Cholesky

Si puo' usare solo nel caso speciale in cui A sia **definita semi-positiva**:

Definition 5.1.5: Matrice semi-definita positiva

A $n \times n$ simmetrica e' **semi-definita positiva** se $\forall x \in \mathbb{R}^n$:

$$x^T A x \geqslant 0$$

Per cui vale:

Proposition 5.1.1 Autovalori di matrici semi-definite positive

Data una matrice A semi-definita positiva, ogni autovalore di A e' positivo o nullo

Grazie a queste proprieta', possiamo dire che $\exists L$:

$$A = L \cdot L^T$$

e trovarlo e' circa due volte piu' efficente rispetto a LU.

5.1.3 metodo Doolittle e Fattorizzazione di Cholesky

Allora sempre restando sulla cuazzo di fattorizzazione LU esiste un bellissimo metodo in grado di fattorizzare agilmente la matrice di partenza A attraverso formule ben preciso, ebbene date il benvenuto al **metodo Doolittle**:

Theorem 5.1.3 metodo Doolittle

Sia $A \in \mathbb{R}^{n\times n}$, il metodo di Doolittle permette di decomporre la matrice A come il prodotto di una matrice triangolare inferiore L (dove $L_{ii} = 1,\ \forall i \in \{1,\dots,n\}$) e una matrice triangolare superiore U del tipo

$$A = LU$$

attraverso le seguenti formule

1. $$U_{pj} = A_{pj} - \sum_{k=1}^{p-1} L_{pk}U_{kj} \quad j \geqslant p$$
2. $$L_{ip} = \frac{1}{U_{pp}}\left(A_{ip} - \sum_{k=1}^{p-1} L_{ik}U_{kp}\right) \quad i > p$$

Dimostrazione: Si noti innanzi tutto che per il normale prodotto riga per colonna dobbiamo trovare due matrici L e U tale che

$$A_{ij} = \sum_{k=1}^{r} L_{ik}U_{kj} \quad r = \min(i,j)$$

che equivale a dire (per ovvia dimostrazione):

$$\begin{aligned} &1) \quad A_{pj} = \textstyle\sum_{k=1}^{p} L_{pk}U_{kj} \quad j \geqslant p \\ &2) \quad A_{ip} = \textstyle\sum_{k=1}^{p} L_{ik}U_{kp} \quad i > p \end{aligned}$$

Da queste formule mi riduco a dimostrare l'algoritmo:

1. $$A_{pj} = \sum_{k=1}^{p} L_{pk}U_{kj} = \sum_{k=1}^{p-1} L_{pk}U_{kj} + L_{pp}U_{pj}$$

   Dato che gli elementi della diagonale nella matrice L sono uguale a 1 si ha:

   $$A_{pj} = \sum_{k=1}^{p-1} L_{pk}U_{kj} + L_{pp}U_{pj} = \sum_{k=1}^{p-1} L_{pk}U_{kj} + U_{pj} \implies U_{pj} = A_{pj} - \sum_{k=1}^{p-1} L_{pk}U_{kj}$$

   ricordandosi $j \geqslant p$
2. $$A_{ip} = \sum_{k=1}^{p} L_{ik}U_{kp} = \sum_{k=1}^{p-1} L_{ik}U_{kp} + L_{ip}U_{pp}$$

   banalmente si ha che:

   $$L_{ip} = \frac{1}{U_{pp}}\left(A_{ip} - \sum_{k=1}^{p-1} L_{ik}U_{kp}\right)$$

   ricordandosi $i > p$

😄

Si ha quindi una fattorizzazione che si ottiene formando le matrici sequenzialmente dalla 1 alla 2

Example 5.1.2

Consideriamo la matrice A:

$$A = \begin{bmatrix} 2 & -1 & 1 \\ 3 & 3 & 9 \\ 3 & 3 & 5 \end{bmatrix}.$$

Passo 1: Inizializzazione delle matrici L e U:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Passo 2: Calcolo degli elementi di U e L riga per riga.
Riga 1: Gli elementi della prima riga di U si calcolano direttamente come:

$$U_{11} = A_{11} = 2, \quad U_{12} = A_{12} = -1, \quad U_{13} = A_{13} = 1.$$

Aggiorniamo U:

$$U = \begin{bmatrix} 2 & -1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Gli elementi di L nella colonna 1 si calcolano come:

$$L_{21} = \frac{A_{21}}{U_{11}} = \frac{3}{2} = 1.5, \quad L_{31} = \frac{A_{31}}{U_{11}} = \frac{3}{2} = 1.5.$$

Aggiorniamo L:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1.5 & 1 & 0 \\ 1.5 & 0 & 1 \end{bmatrix}.$$

Riga 2: Gli elementi della seconda riga di U si calcolano sottraendo il contributo della prima riga (già calcolata) da A:

$$U_{22} = A_{22} - L_{21}U_{12} = 3 - (1.5)(-1) = 4.5,$$

$$U_{23} = A_{23} - L_{21}U_{13} = 9 - (1.5)(1) = 7.5.$$

Aggiorniamo U:

$$U = \begin{bmatrix} 2 & -1 & 1 \\ 0 & 4.5 & 7.5 \\ 0 & 0 & 0 \end{bmatrix}.$$

Gli elementi di L nella colonna 2 si calcolano come:

$$L_{32} = \frac{A_{32} - L_{31}U_{12}}{U_{22}} = \frac{3 - (1.5)(-1)}{4.5} = \frac{4.5}{4.5} = 1.$$

Aggiorniamo L:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1.5 & 1 & 0 \\ 1.5 & 1 & 1 \end{bmatrix}.$$

Riga 3: L'elemento U_{33} si calcola come:

$$U_{33} = A_{33} - L_{31}U_{13} - L_{32}U_{23} = 5 - (1.5)(1) - (1)(7.5) = 5 - 1.5 - 7.5 = -4.$$

Aggiorniamo U:

$$U = \begin{bmatrix} 2 & -1 & 1 \\ 0 & 4.5 & 7.5 \\ 0 & 0 & -4 \end{bmatrix}$$

Risultato finale La matrice A è stata scomposta come:

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1.5 & 1 & 0 \\ 1.5 & 1 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 2 & -1 & 1 \\ 0 & 4.5 & 7.5 \\ 0 & 0 & -4 \end{bmatrix}.$$

Verifica:
Moltiplicando L e U, si ottiene A:

$$A = LU = \begin{bmatrix} 1 & 0 & 0 \\ 1.5 & 1 & 0 \\ 1.5 & 1 & 1 \end{bmatrix} \begin{bmatrix} 2 & -1 & 1 \\ 0 & 4.5 & 7.5 \\ 0 & 0 & -4 \end{bmatrix} = \begin{bmatrix} 2 & -1 & 1 \\ 3 & 3 & 9 \\ 3 & 3 & 5 \end{bmatrix}$$

Adesso possiamo passare alla fattorizzazione di Cholesky, non prima di aver presentato un'importante definizione. Infatti si può usare solo nel caso speciale in cui A sia **definita semi-positiva**:

Definition 5.1.6: Matrice semi-definita positiva

A $n \times n$ simmetrica e' **semi-definita positiva** se $\forall x \in \mathbb{R}^n$:

$$x^T A x \geqslant 0$$

Per cui vale:

Proposition 5.1.2 Autovalori di matrici semi-definite positive

Data una matrice A semi-definita positiva, ogni autovalore di A e' positivo o nullo

Note:

Grazie a questa proposizione si la matrice A da fattorizzare può essere scomposta senza ricorrere alla pivotazione (scambio di righe e/o colonne)

tramite questa osservazione ci possiamo ricondurre al seguente teorema

Theorem 5.1.4 Metodo di Cholesky

Sia $A \in \mathbb{R}^{n \times n}$ una matrice *simmetrica e definita semi-positiva* allora esiste una matrice H triangolare inferiore tale che:

$$A = HH^T$$

Che si può calcolare attraverso tale algoritmo

$$1) \quad H_{pp} = \sqrt{A_{pp} - \sum_{k=1}^{p} H_{pk}^2}$$

$$2) \quad H_{ip} = \left(A_{ip} - \sum_{k=1}^{p-1} H_{ik} H_{pk} \right) \frac{1}{H_{pp}}, \quad i \geqslant p+1$$

Con una complessità computazionale di $O\left(\frac{n^3}{6}\right)$

Dimostrazione: Sia $A \in \mathbb{R}^{n \times n}$ una matrice *simmetrica e definita semi-positiva*, devo dimostrare che $\exists H : A = HH^T$, ovvero che $\exists H : a_{ij} = \sum_{k=1}^{i} h_{ik} h_{kj}^T$ dove $a \in K$ e $h \in H$
Procedimento nel costruire tale matrice:

- elementi nella diagonale:

$$a_{pp} = \sum_{k=1}^{p} h_{pk} h_{kp}^T = \sum_{k=1}^{p-1} h_{pk} h_{kp}^T + h_{pp} h_{pp}^T$$

  Per definizione di matrice trasposta si ha che $h_{pp} = h_{pp}^T$, il che implica

$$h_{pp} = \sqrt{a_{pp} - \sum_{k=1}^{p-1} h_{pk}^2}$$

- elementi sotto la diagonale:

$$a_{ip} = \sum_{k=1}^{p} h_{ik} h_{kp}^T = \sum_{k=1}^{p-1} h_{ik} h_{kp}^T + h_{ip} h_{pp}^T$$

  quindi

$$h_{pp} = \left(a_{ip} - \sum_{k=1}^{p-1} h_{ik} h_{pk} \right) \frac{1}{h_{pp}}$$

Inoltre si ha, per definizione di matrice trasposta e matrice triangolare inferiore, che:

$$a_{ip} = \sum_{k=1}^{p} h_{ik} h_{kp}^T = \sum_{k=1}^{p} h_{ik} h_{pk} = \sum_{k=1}^{p} h_{pk} h_{ik} = \sum_{k=1}^{p} h_{pk} h_{ki}^T = a_{ip}$$

Dato che per ipotesi la matrice A è simmetrica si ha che questa proposizione è corretta 😄

Come per la fattorizzazione LU ci troviamo a discutere del sistema seguente:

Corollary 5.1.2

La suluzione x si ottiene attraverso la soluzione di due sistemi triangolari:

$$\begin{cases} Hy = b \\ H^T x = y \end{cases}$$

5.1.4 metodi iterativi

I metodi iterativi si basano sull'idea di calcolare la soluzione del sistema

$$Ax = b$$

Come limite di una sucessione di vettori:

$$x = \lim_{k \to \infty} x^k$$

e dove ci si fermerà al minimo n per cui si abbia

$$\|x^n - x\| < \epsilon$$

Una strategia generale per costruire una sucessiano $\{x^k\}$ è basata sulla tecnica dello "splitting" $A = P - N$ dove P ed N sono due matrici e P è non singolare. Precisamente si ottiene

$$Px = Nx + b$$

A questo punto, possiamo riscrivere il sistema in forma iterativa. L'idea è di utilizzare una stima iterativa $x^{(k)}$ al passo k per sostituire x nel termine dipendente Nx, ottenendo una formula iterativa:

$$Px^{(k+1)} = b + Nx^{(k)}.$$

Scomponendo ulteriormente:

$$x^{(k+1)} = P^{-1}(b + Nx^{(k)}).$$

Questa è la **formula iterativa** per calcolare una nuova stima $x^{(k+1)}$ a partire dalla stima precedente $x^{(k)}$, adesso formalizziamo tutto in un teorema non prima di aver dichiarato un bel lemma

Lenma 5.1.1

Sia $B = P^{-1}N$, allora si ha che

$$\lim_{k\to\infty} B^k = 0 \iff \|B\| < 1$$

Theorem 5.1.5

Sia $A = P - N$, si ha che assegnato x^0 si ottiene x^k risolvendo tali sistemi

$$Px^k = Nx^{k-1} + b, \quad k \geqslant 1$$

e si ha che x^k è un'iterazione più vicina alla soluzione sse la matrice $\lim_{k\to\infty} (P^{-1}N)^k$

Dimostrazione: Sia

$$Px^k = Nx^{k-1} + b$$

Posto

$$e^k = x^k - x$$

usando e^k, possiamo scrivere:

$$x^{(k)} = x + e^k.$$

Per il passo successivo $x^{(k+1)}$, avremo:

$$x^{(k+1)} = x + e^{(k+1)}.$$

Sostituendo nella formula iterativa $x^{(k+1)} = P^{-1}(b - Nx^{(k)})$, otteniamo:

$$x + e^{k+1} = P^{-1}\left(b - N(x + e^k)\right) \iff x + e^{k+1} = P^{-1}\left(b - Nx + Ne^k\right)$$

Poiché $Ax = b$, possiamo sostituire $b = Ax$, e quindi otteniamo:

$$x + e^{k+1} = P^{-1}\left(Ax - Nx - Ne^k\right) \iff x + e^{k+1} = P^{-1}\left(Px - Nx - Ne^k\right)$$

😄

presentiamo qui il **metodo Jacobi**

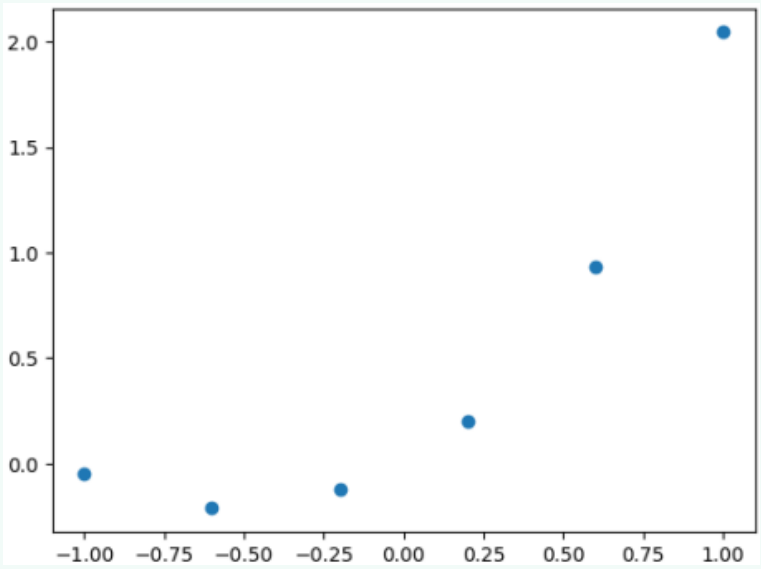
5.2 Problema di interpolazione

Definition 5.2.1

Dati $n+1$ punti distinti $(x_0), (x_1, y_1), \ldots, (x_n, y_n)$ con $x_0 < x_1 < \cdots < x_n$ chiamati **nodi**, si definisce il **problema di interpolazione** quel problema che consiste nel trovare una funzione (polinomiale) $p(x)$ chiamata **interpolante**, tale che $\forall k \in [0, n], p(x_k) = y_k$

Esempietto con alcuni nodi:

Example 5.2.1



Si può risolvere il polinomio di grado n come:

$$p(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n$$

Per identificare questo polinomio, ordunque, serve calcolare i coefficienti $c_i, i \in [1, n]$

5.2.1 Esistenza dell'unicità

Theorem 5.2.1

$\forall i \in [1, n], \forall (x_i, y_i)$ con i nodi t_i, distinti tra loro, esiste un unico polinomio di grado minore od uguale a n, che indichiamo $P_n(x)$ con e chiamiamo polinomio interpolatore dei valori y_i nei nodi t_i, tale che

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

Per la dimostrazione serve prima enunciare un lemma (che non verrà dimostrato)

Lenma 5.2.1

Sia $r_n(x)$ un polinomio a coefficienti reali di grado n, il numero massimo di zeri distinti è n, ovvero:

- Un polinomio di grado n può annullarsi in al massimo n punti distinti.
- Se un polinomio di grado n si annulla in più di n punti distinti, allora deve essere il polinomio nullo, cioè identicamente uguale a zero per ogni valore di x

Dimostrazione: Assumiamo che esistano due polinomi distinti di grado n, ovvero $p_n(x)$ e $q_n(x)$ che soddisfino entrambi le relazioni nodali precedenti. La loro differenza $p_n(x) - q_n(x)$ è ovvio verificare che è un polinomio di

grado n che si annulla in $n+1$ punti distinti. Per il fatto che il polinomio si annulla in $n+1$ punti e per il lemma enunciato si ha che $p_n(x) - q_n(x) = 0$, quindi $p_n(x) = q_n(x)$. Per assurdo dimostrato
Q.e.d 😄

5.2.2 Calcolo del polinomio di interpolazione

Metodo classico

È ovvio verificare che i coefficienti del polinomio di interpolazione di $n+1$ punti $(x_i, y_i) = (x_i, f(x_i))$ si potrebbero calcolare risolvendo il sistema lineare:

$$\begin{cases} c_0 + c_1 x_1 + \cdots + c_n x_1^n = y_1 \\ c_0 + c_1 x_2 + \cdots + c_n x_2^n = y_2 \\ \cdots \\ c_0 + c_1 x_n + \cdots + c_n x_n^n = y_n \end{cases}$$

Tuttavia la matrice dei coefficienti di questo sistema, detta **matrice di Vandermonde** può essere molto mal condizionata. Inoltre la risoluzione del sistema lineare richiede almeno $O(\frac{n^3}{3})$

Metodo del polinomio interpolatore di Lagrange

Introduciamo i polinomi base di lagrange

Definition 5.2.2

Si chiamano **polinomi base di Lagrange** di grado n particolari funzioni tali che:

$$\phi_k(x_j) = \delta_{k,j}$$

dove $k = j \implies \delta_{k,j} = 1$ e $k \neq j \implies \delta_{k,j} = 0$

La funzione $\phi_k(x)$ può essere scritta come:

$$\phi_k(x) = \prod_{j=0, j\neq k}^{n} \frac{x - x_j}{x_k - x_j}$$

Si può notare, infatti, che include tutti i punti x_j tranne x_k, quindi l'indice j percorre tutti gli indici da 0 a n e nel caso in cui la x in input sia diversa da x_k vi sarà un x_j tale che $x_j = x$, in quel caso il numeratore farà 0 e renderà nulla la productoria. Nel caso, invece, la x in input sia uguale a x_k avremo $\frac{x_k - x_j}{x_k - x_j} \forall j$ rendendo la productoria uguale ad 1.
Definito il polinomio posso scrivere il polinomio di interpolazione di Lagrange:

Definition 5.2.3

Si chiama **polinomio interpolatore di Lagrange** tale funzione:

$$p_n(x) = y_0\phi_0(x) + y_1\phi_1(x) + \cdots + y_n\phi_n(x) = \sum_{k=0}^{n} y_k\phi_k(x)$$

Infatti soddisfa le condizioni di interpolazione

$$p_n(x_i) = \sum_{k=0}^{n} y_k\phi_k(x_i) = \sum_{k=0}^{n} y_k\delta_{i,k} = y_i$$

5.2.3 Errore di un polinomio interpolatore di una funzione

Iniziamo con la definizione di un polinomio interpolatore di una funzione

Definition 5.2.4

Sia $f[0,n] \to \mathbb{R}$ una funzione continua nel suo dominio e sia $y_i = f(x_i), \forall i \in [0,n]$. Sia anche $p_n(x)$ il polinomio interpolatore nei punti $(x_0,y_0),(x_1,y_1),\dots,(x_n,y_n)$
Allora $p_n(x)$ è detto **polinomio interpolatore di** f e si denota:

$$p_n(f)$$

Attraverso il polinomio di Lagrange è possibile quantificare l'errore che si commette sostituendo ad una funzione f il suo polinomio interpolante attraverso il seguente teorema

Theorem 5.2.2

Sia I un intervallo limitato, e si considerino $n+1$ nodi di interpolazione distinti $x_i, i = 0,\dots,n$ in I. Sia f derivabile con continuità fino all'ordine $n+1$ in I. Allora

$$\forall x \in I. \exists \eta \in I : E(x) = f(x) - p_n(x) = \frac{f^{n+1(\eta)}}{(n+1)!} \prod_{i=0}^{n} (x - x_i)$$

E dove ovviamente $E(x_i) = 0$ dove $i = 0,\dots n$

l'obbiettivo di questo teorema, quindi, è fornire una formula da cui ricavare un **errore di interpolazione** $E(x)$, che è la differenza tra la funzione originale $f(x)$ e il polinomio interpolante $p_n(x)$ nei punti in cui non conosciamo il valore esatto di $f(x)$ (ovvero tutti i punti diversi dai nodi)
Da questo teorema non riusciamo a dedurre che, qualsiasi sia la distribuzione dei nodi, $max_{x\in I}|E(X)| \to 0$ per $n \to \infty$ ovvero (che il massimo valore dell'errore $E(x) = f(x) - p_n(x)$ su un intervallo I tende a zero al crescere di n). Esistono, infatti, alcune funzioni, come la funzione di Runge, in cui per determinate scelte dei nodi, per esempio equidistanti dove $max_{x\in I}|E(X)| \to \infty$ per $n \to \infty$. Alla luce di questa considerazione si ha che ad un aumento del grado n del polinomio interpolatore non corrisponde necessariamente un miglioramento nella ricostruzione di una funzione f.

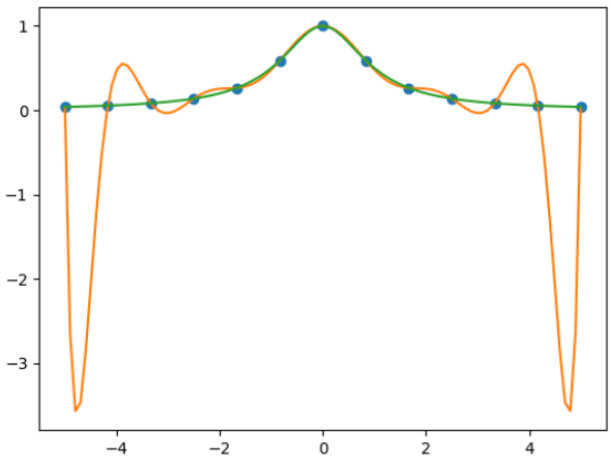


Figure 5.1: In questa immagine troviamo la funzione di Runge (in arancione) e il suo polinomio interpolante (in verde), dove la scelta dei nodi ha causato un errore di interpolazione elevato.

Il fenomeno di Runge può essere evitato utilizzando opportune distribuzioni di nodi. In particolare, su un arbitrario intervallo $[a,b]$ consideriamo i cosiddetti **nodi di Chebyshev- Gauss-Lobatto**:

$$x_i = \frac{a+b}{2} - \frac{b-a}{2}\cos\left(\frac{i\pi}{n}\right) \quad \text{con } i = 0,\dots,n$$

Notiamo che $x_i = \cos\left(\frac{i\pi}{n}\right)$ se $[a,b] = [-1,1]$. Si può dimostrare che se f è una funzione continua e derivabile con continuità in $[a,b]$ il polinomio interpolante $p_n(x)$ converge a $f(x)$ per $n \to \infty$

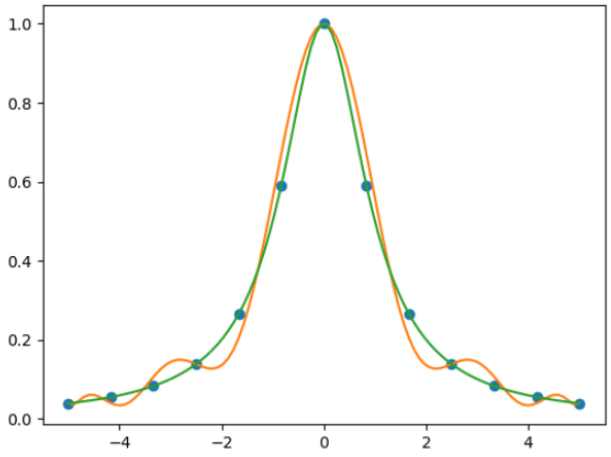


Figure 5.2: in questa immagina la scelta dei nodi ha causato un errore di interpolazione relativamente basso

5.2.4 Condizionamento dell'interpolazione polinomiale

Alle volte non si dispone della funzione $f(x_i)$ relativi ai nodi x_i, con $i = 0, \dots, n$ in un intervallo I ma solo della sua approssimazione, denotata $\tilde{f}(x_i)$. La perturbazione $f(x_i) - \tilde{f}(x_i)$ potrebbe essere dovuta ad esempio all'effetto degli errori di arrotondamento o ad errori di misura. Se chiamiamo $\tilde{p}_n x$ il polinomio interpolatore dei nodi $(x_i, \tilde{f}(x_i))$, si ha:

$$\max_{x \in I} |p_n(x) - \tilde{p}_n(x)| = \max_{x \in I} (\sum_{i=0}^{n} (f(x_i) - \tilde{f}(x_i)\phi_i(x))) = \Lambda_n(x) \max |f(x_i) - \tilde{f}(x_i)|$$

Dove $\Lambda_n(x)$ è il numero di condizione che in questo caso è:

$$\Lambda_n(x) = \max_{x \in I} |\sum_{i=0}^{n} \phi_i(x)|$$

Che viene detta **costante i Lebesgue** che nel caso dell'interpolazione polinomiale su nodi equispaziati la costante di Lebesgue assume il valore:

$$\Lambda_n(x) \simeq \frac{2^{n+1}}{n(\log(n) + \gamma)e}$$

Dove $\gamma \simeq 5.47721$ è la **costante di Eulero**. Ciò comporta che per n grande questo tipo di interpolazione potrebbe essere instabile

Chapter 6

Problemi inversi

6.1 Problemi diretti

Nel mondo reale i vari fenomeni sono descritti dalla fisica attraverso un problema diretto come

$$\text{causa} \longrightarrow \text{modello} \longrightarrow \text{effetto}$$

Per esempio, se voglio calcolare l'accelerazione di un'auto, ne individuo i vari passaggi

- La causa: ovvero la forza generata dal motore
- Modello: la leggi di newton che forniscono le formule e le leggi apposite (supponendo un sistema inerziale)
- Effetto: calcolo dell'accelerazione

Quindi acquisisco in input l'origine dell'effetto (ovvero la forza) attraverso un modello (leggi di newton) elaboro questo dato ed infine ne calcolo l'effetto finale (l'accelerazione). In si può scrivere "matematichese": $x \longrightarrow A \longrightarrow y$, che traslato in algebra lineare si ha così:

$$Ax \longrightarrow y$$

Si giunge così alla definizione di **problema diretto**

Definition 6.1.1

Si definisce **problema diretto** il processo mediante il quale si determina l'effetto dato un sistema e le cause iniziali

In altre parole: input -¿ output

6.2 Problemi inversi

D'altra parte esistono dei casi in cui si riesce solo ad osservare l'effetto di un fenomeno ed occorre, partendo da questo, risalire alla causa, da questo concetto abbiamo la definizione di **problema inverso**

Definition 6.2.1

Un **problema inverso** è un tipo di problema matematico3 in cui, dato l'effetto o l'output osservato, si cerca di determinare le cause o i parametri del sistema che lo hanno generato

Che in termini matematici significa determinare una x conoscendo A e y, ovvero output -¿ input.
Tuttavia nel mondo reale i dati y sono sempre affetti da errore (detto anche **rumore**) rappresentato dal vettore δ che non sono solo errori di rappresentazione ma anche legati a problemi di tipo fisico, forti di questa consapevolezza il problema diventa quindi:

$$Ax = y + \delta = y^{\delta}$$

6.2.1 Problemi inversi lineari con matrici mal condizionate

Vi sono delle classi di problemi la cui matrice A relativa al modello matematico del problema è mal condizionata, tali problemi si dicono **mal posti**. In questo caso il problema viene risolto come un problema di minimi quadrati (che permette di avere anche una matrice rettangolare):

$$min_x \|Ax - y^\delta\|_2^2$$

la cui soluzione è

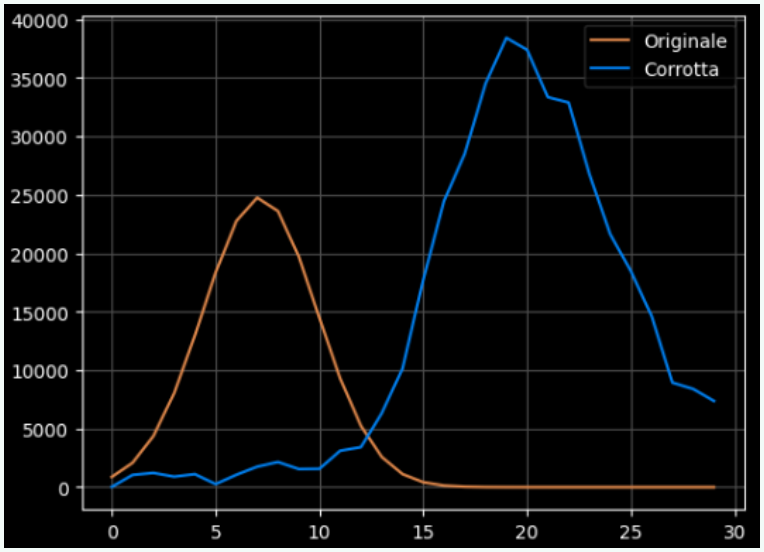
$$x^* = \sum_{i=1}^{n} \frac{u_i^T y^\delta}{\sigma_i} v_i$$

6.2.2 Perturbazione dell'errore

A causa di un mal condizionamento di A si ha che, in presenza di un errore δ, si ha che questo errore venga eccessivamente amplificato nella soluzione dei minimi quadrati

Example 6.2.1

Si consideri una soluzione "vera" (ground truth) denotata con x_{GT} a cui si applica una matrice A che rappresenta il sistema di acquisizione dati per ottenere il vettore $y : Ax = y$
Al vettore y si aggiunge un vettore δ che rappresenta il rumore e che e' preso come vettore random da una distribuzione normale con media nulla: $y^\delta = y + \delta$
Ecco il grafico di questo scempio:



Questo accade perché quando una matrice A è mal condizionata i suoi valori singolari σ_i sono molto piccoli, pertanto il numeratore della sommatoria è molto grande, per capire meglio si osservino i calcoli:

$$\min_x \|Ax - y^\delta\|_2^2 = min_x \|Ax - (y + \delta)\|_2^2$$

La cui soluzione è:

$$x^* = \sum_{i=1}^{n} \frac{u_i^T y^\delta}{\sigma_i} v_i = \sum_{i=1}^{n} \frac{u_i^T (y + \delta)}{\sigma_i} v_i = \sum_{i=1}^{n} \frac{u_i^T y}{\sigma_i} v_i + \sum_{i=1}^{n} \frac{u_i^T \delta}{\sigma_i} v_i$$

È proprio la seconda sommatoria il problema! È facile verificare che più i σ_i sono piccoli più $u_i^T \delta$ è matematicamente "amplificato"

Condizioni di Picard

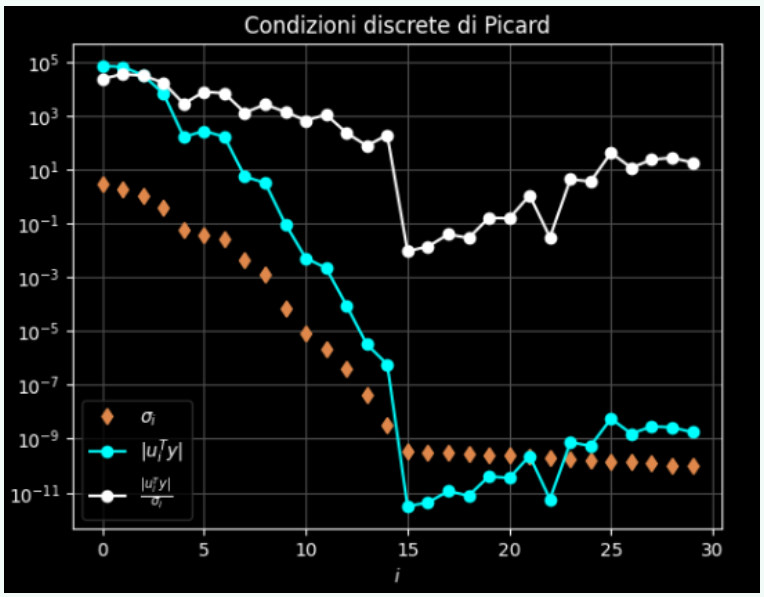
La **condizione discreta di Picard** è un criterio utilizzato per diagnosticare la "malcondizionatura" nei problemi inversi lineari e per capire la stabilità della soluzione. La condizione discreta di Picard si basa sull'analisi dei valori singolari σ_i e i cosidetti coefficienti di Fourier $u_i^T y^\delta$

Definition 6.2.2

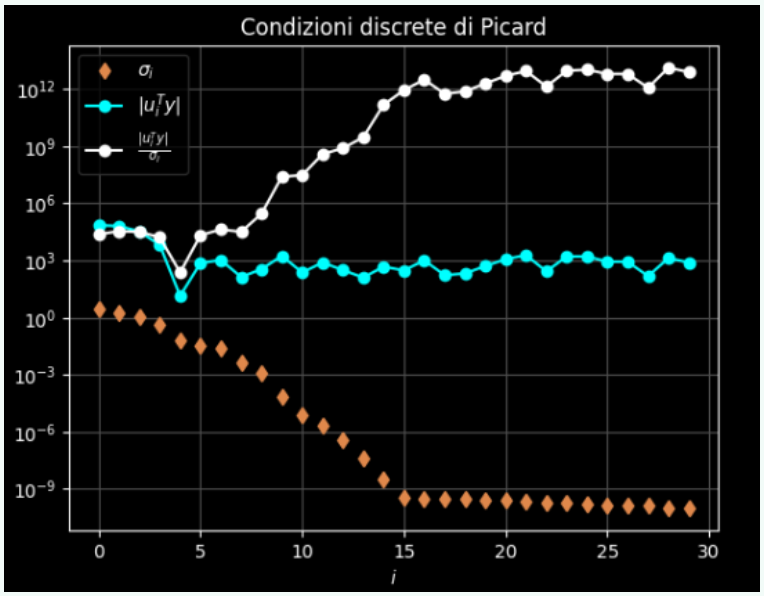
La **condizione discreta di Picard** afferma che che, affinché il problema sia ben posto, i coefficienti di Fourier $u_i^T y^\delta$devono decrescere almeno altrettanto velocemente dei valori singolari σ_i

Example 6.2.2

Si consideri questo esempio:



In questo caso la condizione di Picard è rispettata fino all' indice $i = 15$
Si consideri quest'altro esempio:



In questo caso la condizione di Picard è rispettata fino all' indice $i = 5$

6.2.3 Metodi di regolarizzazione

Per affrontare la malcondizionatura e ottenere una soluzione più stabile e accettabile, è necessario modificare il problema attraverso tecniche dette metodi di regolarizzazione. Esistono diverse tecniche di regolarizzazione, verrano riportate qui sotto

Decomposizione in Valori Singolari Troncata

Nel metodo di regolarizzazione denominato TSVD (Truncated Singular Value Decomnposition) la soluzione del problema di minimi quadrati:

$$min_x \|Ax - y^\sigma\|_2^2$$

viene calcolata come:

$$x_{TSVD} = \sum_{i=1}^{K} \frac{u_i^T y^\delta}{\sigma_i} v_i$$

Con $K < n$. Posso anche scrivere la soluzione come:

$$x_{TSVD} = \sum_{i=1}^{n} f_i \frac{u_i^T y^\delta}{\sigma_i} v_i$$

dove i coefficienti f_i sono detti **fattori di filtro**, dove in questo caso sono dati da:

$$f_i = \begin{cases} 1 & i \leqslant K \\ 0 & i > K \end{cases}$$

E dove un buon valore K è quello dopo il quale non sono più soddisfatte le condizioni di Picard

Regolarizzazione di Tikhanov

La **regolarizzazione di Tikhonov**, invece, utilizza fattori di filtro che siano numeri in $[0, 1]$ e che quindi "pesino" le componenti della somma in modo piu' graduale. In particolare,devono essere pesate di più le componenti di indice piccolo, associate ai valori singolari più grandi, e meno quelle di indice grande, associate ai valori singolari piu' piccoli
Il problema diventa quindi:

$$x_{tikh} = min_x \|Ax - y^\sigma\|_2^2 + \lambda \|Lx\|_2^2$$

Dove:

- $\lambda > 0$ è il parametro di regolarizzazione che pesa la parte di congruenza con i dati e la parte di regolarità della soluzione
- L è la matrice Identità

Si può anche scrivere in questo modo:

$$x_{tikh} = min_x \|Mx - Y\|_2^2$$

Dove:

- M è la matrice di dimensione $2m \times n$ che ha come blocchi in colonna rispettivamente la matrice A e la matrice λI
- Y è il vettore di dimensione $2m$ che ha come vettori colonna rispettivamente y^δ e il vettore nullo

Si può inoltre dimostrare che la soluzione di Tikhonov posso anche scriverla tramite i fattori di filtro come:

$$x_{tikh} = \sum_{i=1}^{n} f_i \frac{u_i^T y^\delta}{\sigma_i} v_i$$

dove:

$$f_i = \frac{\sigma_i^2}{\sigma_i^2 + \lambda}$$

Principio di maassima discrepanza

Si giunge, tuttavia, ad un problema: la scelta del parametro di regolarizzazione nel metodo di Tikhonov. Infatti questa scelta è sicuramente la parte piu' delicata della regolarizzazione, un parametro troppo piccolo non toglie il rumore dalla soluzione, mentre un parametro troppo grande produce una soluzione troppo regolare, in cui non e' piu' presente il rumore ma per esempio le oscillazioni o i picchi che ci sono nella soluzione esatta vengono troppo ridotti.
Teoricamente il valore ottimale del parametro λ_{opt} è quello che minimizza:

$$\lambda_{opt} = \min_{\lambda} \|x_\lambda - x_{GT}\|_2^2$$

dove x_λ è la soluzione calcolata in corrispondenza di un certo λ.
Non esiste un metodo sempre efficiente per scegliere il parametro di regolarizzazione. Sicuramente in pratica spesso si usa una tecnica euristica che consiste nel provare alcuni parametri e scegliere quello che produce la soluzione che ci sembra migliore (trial and test). La tecnica sicuramente più utilizzata in pratica è il principio di massima discrepanza.

Definition 6.2.3

Il **Principio di Massima Discrepanza** afferma che il parametro di regolarizzazione λ dovrebbe essere scelto in modo tale che la norma del residuo (cioè la differenza tra i dati osservati e quelli ricostruiti) sia proporzionale alla stima della norma del rumore nei dati
Formalmente:

$$\|Ax_\lambda - y^\delta\| = v_{DP}|\delta|_2^2$$

Dove:

- x_λ è la soluzione regolarizzata con parametro λ
- v è un fattore di proporzionalità (spesso scelto vicino a 1)
- $\|\delta\|$ è la norma dell'errore (spesso è impossibile conoscerla ma si può fare una stima)

Quindi il residuo deve essere uguale alla norma del rumore, quindi provo tanti λ, quello più vicino a questa soluzione è quello migliore

Chapter 7

immagini

Un problema piuttosto complesso è la conversione di una immagine da analogica a digitale che, ovviamente, deve essere gestita tramite i pixel. Astraendo matematicamente un'immagine, infatti, è possibile trarre una matrice $M \times N$ che rappresenta i vari pixel ed i suoi coefficienti, ai fatti, rappresentano quali sfumature di colore verranno mostrate ed in quale posizione. Tuttavia se un'immagine di grigi è rappresentabile da un'unica matrice che avrà come coefficienti soltanto valori interi da un minimo di 0 ad un massimo di 255 (0 rappresenta il colore nero e 255 rappresenta il colore bianco), un'immagine a colore è ottenuta tramite la sovrapposizione di tre rappresentazioni della stessa immagine: una in scala di rossi, una in scala di verdi ed una in scala di blu; sarà quindi richiesto che il colore di ogni pixel di questa immagine sia rappresentato da vettore di tre valori, ciascuno per ogni intensità del colore rosso, verde e blu

Definition 7.0.1

Sia $A \in \mathbb{R}^{M \times N}$ e sia $k \in \mathbb{N}$, diremo che: una matrice B di dimensioni $(M + 2k) \times (N + 2k)$ **estende** la matrice A se e solo se vale:

$$\forall (i, j) \in \{\dots, M\} \times \{1, \dots, N\}. A_{(i,j)} = B_{(i+k, j+k)}$$

Esempietto:

Example 7.0.1

$$A = \begin{pmatrix} 7 & 8 & 9 \\ 12 & 13 & 14 \\ 17 & 18 & 19 \end{pmatrix}$$

Si può affermare che che la seguente matrice

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{pmatrix}$$

estende A, dove si ha $k = 1$

Chiaramente, per ogni matrice ne esistono infinite che la estendono

Definition 7.0.2

Sia $K \in \mathbb{R}^{D \times D}$ con D dispari. Si definisce il **centro di matrice** della matrice K come l'elemento di coordinate $(\frac{D+1}{2}, \frac{D+1}{2})$

Example 7.0.2

Sia la matrice di dimensioni 5×5

$$A = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{pmatrix}$$

Il suo centro di matrice è $(\frac{5+1}{2}, \frac{5+1}{2}) := 13$

Definition 7.0.3

Sia $A \in \mathbb{R}^{M \times N}$ e sia $B \in \mathbb{R}^{(M+(D-1)) \times (N+(D-1))}$ con D dispari che estenda A. $\forall (i,j) \in \{1,\dots,M\} \times \{1,\dots,N\}$ si definisce $W_{(i,j)} \in \mathbb{R}^{D \times D}$ si definiscono **sotto-matrici** di B, il cui centro è $B_{(i+\frac{D-1}{2}, j+\frac{D-1}{2})}$

Example 7.0.3

Sia $A \in \mathbb{R}^{3 \times 3}$

$$\begin{pmatrix} 7 & 8 & 9 \\ 12 & 13 & 14 \\ 17 & 18 & 19 \end{pmatrix}$$

Occorre trovare una matrice $(3 + (D-1)) \times (3 + (D-1)) = (3+2) \times (3+2)$ che estenda A, ad esempio:

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 6 & 7 & 8 & 9 & 10 \\ 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 \\ 21 & 22 & 23 & 24 & 25 \end{pmatrix}$$

ne segue che, per esempio, la matrice $W_{(1,3)}$ è la sotto-matrice di B, di dimensioni 3×3, il cui centro sia $B_{(2,4)}$. Di seguito:

$$W_{(1,3)} = \begin{pmatrix} 3 & 4 & 5 \\ 8 & 9 & 10 \\ 13 & 14 & 15 \end{pmatrix}$$

Definition 7.0.4

Sia $A \in^{M \times N}$ e $K \in^{D \times D}$, con D un numero naturale dispari. Sia B una matrice di dimensioni $(M + (D-1)) \times (N + (D-1))$ che estenda A. $\forall (i,j) \in \{1,\dots,M\} \times \{1,\dots,N\}$, siano $W_{(i,j)}$ le sotto-matrici di dimensione $D \times D$ definite precedentemente.
Definiamo la matrice $C \in^{M \times N}$, come segue:

$$\forall (i,j) \in \{1,\dots,M\} \times \{1,\dots,N\} : C_{i,j} = \sum_{m=1}^{\frac{D-1}{2}} \sum_{n=1}^{\frac{D-1}{2}} K(m,n) W_{(i,j)}(m,n)$$

Tale matrice C, che verrà anche denotata con $C = [K * A|B]$, è chiamata **convoluzione** di K e A rispetto all'estensione B
All'interno del contesto di questa definizione, la matrice K prenderà il nome di **nucleo** di convoluzione o kernel di convoluzione

7.1 Estensione di una matrice

Definition 7.1.1

Sia k un numero naturlae e B una matrice di dimensioni $(M+2k)\times(N+2k)$. Definiamo la funzione:

$$\mathcal{P} :^{M\times N}\rightarrow^{(M+2k)\times(N+2k)} \quad X \rightarrow \mathcal{P}(X)$$

Tale che:

$$\forall(i,j)\in\{1,\dots,M\}\times\{1,\dots,N\} : X_{(i,j)} = \mathcal{P}(X)_{(i+k,j+k)}$$

e:

- le prime k colonne di $\mathcal{P}(X)$ siano uguali alle prime k colonne di B
- le ultime k colonne di $\mathcal{P}(X)$ siano uguali alle ultime k colonne di B
- le prime k righe di $\mathcal{P}(X)$ siano uguali alle prime k righe di B
- le ultime k righe di $\mathcal{P}(X)$ siano uguali alle ultime k righe di B

La funzione P così definita prenderà il nome di **patter di estensione di ordine k rispetto alla matrice B**

È chiaro che, per come è definita la funzione $\mathcal{P}(X)$, questa estenderà sempre la matrice X

Definition 7.1.2

Definiamo **funzione di convoluzione** la seguente:

$$[K*(\cdot)|\mathcal{P}] :^{M\times N}\rightarrow^{M\times N} \quad X \rightarrow [K*X|\mathcal{P}(X)]$$

- Dove K è un nucleo di convoluzione di dimensioni $D\times D$ fissato
- $\mathcal{P}$ è un pattern di estensione di ordine $\frac{D-1}{2}$, rispetto ad una qualche matrice $B\in^{(M+(D+1)\times N+(D-1))}$, sempre fissata

Example 7.1.1

Si prendi come esempio tale immagine

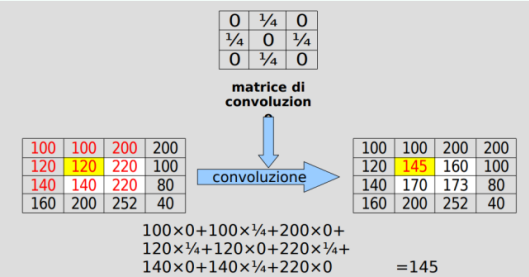
E si supponi sia una matrice A e si supponga tale matrice abbia, come nucleo di convoluzione K, tale

matrice:

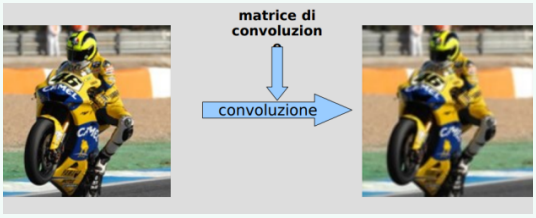
$$K = \begin{pmatrix} 0 & \frac{1}{4} & 0 \\ \frac{1}{4} & 0 & \frac{1}{4} \\ 0 & \frac{1}{4} & 0 \end{pmatrix}$$

Ora non resta che svolgere la convoluzione in sè. Ecco come:

- consideriamo una sotto-matrice W, sempre di dimensioni $D \times D$, della matrice B
- svolgiamo il prodotto scalare fra: la prima colonna di W e la prima riga di K , la seconda colonna di W e la seconda riga di K , così via fino alla D-esima colonna di W e alla D-esima riga di K
- sommiamo tutti i prodotti scalari ottenuti
- il valore così determinato, diventerà il valore del pixel al centro della sotto-matrice W considerata
- ripetere per tutte le sotto-matrici
- rimuovere le condizioni al bordo



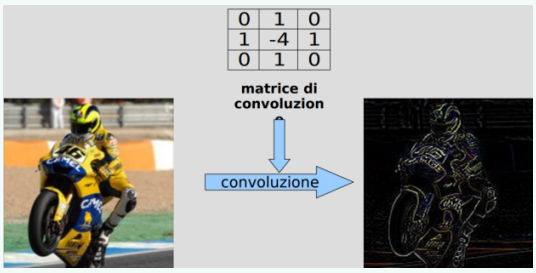
In questo caso, il kernel utilizzato sostituisce il valore di ogni pixel con la media aritmetica dei valori contenuti nei quattro pixel ad esso adiacenti
Ottenendo il seguente effetto di sfocatura:



Se si considera, invece, il seguente kernel:

$$K = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Osserviamo che, poiché la somma dei coefficienti è 0, sappiamo che se un pixel ha approssimativamente lo stesso colore dei quattro che gli sono adiacenti, allora verrà rimpiazzato da un valore prossimo a zero; in altre parole, il pixel diverrà di colore nero. Ne segue che tutti e soli i pixel che avranno un valore non nullo, saranno quelli posizionati in mezzo a due regioni di colori contrastanti. Pertanto, questo è un filtro che evidenzia i bordi dell'immagine



7.2 Point Spread Function

Nel momento in cui un macchinario sensibile alla luce si accinge a fotografare, per esempio, ad una fonte luminosa puntiforme del tipo:
Si ha che l'imagine ottenuta sarà:
L'immagine che si ottiene è, quindi, un allargamento della fonte luminosa centrata nella fonte puntiforme. Questo fenomeno è del tutto deterministico e legato all'apperacchiatura utilizzata per catturare l'immagine.
Sia X la prima immagine e Y la seconda, matematicamente si ha che:

$$A(X) = Y$$

Dove $\mathcal{A}$ è una funzione chiamata **Point Spread Function**. Ad essere più precisi riporterò qui la definizione di Point Spread Function

Definition 7.2.1

Sia x un immagine che rappresenta perfettamente il nostro oggetto originale. È definita $\mathcal{A}$ Point Spread Function, tale funzione:

$$\mathcal{A}(x) = [K * x|\mathcal{P}]$$

Dove K è un kernel di convoluzione e $\mathcal{P}$ è un pattern di estensione

7.2.1 PSF con errori non deterministici

Tuttavia gli errori non deterministici non sono solo gli unici errori che corrompono le immagini, nel corso dell'acquisizione dell'immagine fenomeni aleatori e sostanzialmente imprevedibili danneggiano ulteriormente la nitidezza dell'immagine ottenuta.
Adesso sia w un immagine i cui valori dei pixel siano stati selezionati aletoriamente secondo la distribuzione del rumore Gaussiano bianco.
Si ha, quindi, un modello matematico **discreto** che spieghi come si sia formulata l'immagine g (quella restituita dal nostro sistema di formazione di immagini), a partire da x, l'immagine che rappresenta veramente l'oggetto:

$$y^{\delta} = \mathcal{A}(x) + w = [W * x|\mathcal{P}] + w$$

dove:

- K è un qualche kernel di convoluzione
- P è un pattern di estensione
- w è il medesimo di cui sopra
- $\mathcal{A}$ è la point spread function

Alternativamente si piò riscrivere il modello nella seguente forma matriciale:

$$y^{\delta} = Ax + w$$

Dove $A \in^{M \times N}$ che contiene le informazioni del nucleo di convoluzione ed è tale che:

$$Ax = K * x$$

Adesso ovviamente si vuole ardentemente sapere l'immagine originale x zio pera

Soluzione naive

La soluzione naive è quella ottenuta risolvendo il problema di minimi quadrati:

$$min_x \|Ax - y^{\delta}\|_2^2$$

Essendo la matrice A di dimensione non molto piccola, calcolare la soluzione con la SVD è troppo costoso (ricordiamo che la complessità computazionale della fattorizzazione SVD è $O(4/3mn^2)$ per una matrice di dimensione $m * n$). Quindi si risolvono le equazioni normali:

$$A^T Ax = A^T y^\delta$$

Il sistema ha matrice simmetrica e definita positiva (supponiamo A di rango massimo) e per ragioni computazionali è conveniente utilizzare un metodo iterativo. Quindi si applichiamo il CGLS al sistema delle equazioni normali

7.2.2 Metriche di valutazione

Le **metriche di valutazione** sono strumenti matematici utilizzati per quantificare la qualità di una ricostruzione, come nel caso della ricostruzione di un'immagine. Servono per confrontare un'immagine ottenuta (ricostruita) con il "ground truth" (l'immagine originale), con l'obiettivo di avere un numero che rappresenti quanto la ricostruzione sia accurata o fedele rispetto all'originale. Ci sono diverse metriche, proprio per avere diversi punti di vista

errore relativo

Definition 7.2.2

siano x_{GT} il "ground truth" e x l'immagine ricostruita, viene definito l'errore relativo ER tale valore:

$$ER = \frac{\|x - x_{GT}\|_2^2}{\|x_{GT}\|_2^2}$$

Rapporto Segnale-Rumore di Picco

Prima di definire il Rapporto Segnale-Rumore di Picco occorre prima deifinire **l'errore quadratico medio**

Definition 7.2.3

Siano M e N le dimensioni dell'immagine, si definisce **l'Errore Quadratico Medio** il seguente valore:

$$MSE = \frac{\|x - x_{GT}\|_2^2}{MN}$$

Questo valore misura la differenza media al quadrato tra i pixel dell'immagine originale e quelli della ricostruzione. adesso si può introdurre il PSNR

Definition 7.2.4

Sia $\max_{i,j} |x_{i,j}|$ è il valore massimo dell'immagine ricostruita (ad esempio, 255 per immagini in scala di grigi a 8 bit) e MNE l'errore quadratico medio, si definisce **Rapporto Segnale-Rumore di Picco** tale valore:

$$PSNR = 10 \cdot \log_{10}\left(\frac{(\max_{i,j} |x_{i,j}|)^2}{MSE}\right)$$

Un valore di PSNR più alto indica una qualità della ricostruzione migliore, perché significa che la differenza tra l'immagine ricostruita e l'originale è bassa

Indice di Similarità Strutturale (SSIM)

Il Structural Similarity Index (SSIM) è una metrica più complessa, progettata per essere più coerente con la percezione visiva umana. Considera non solo le differenze di intensità dei pixel, ma anche la struttura, luminanza e contrasto delle immagini. In altre parole, valuta quanto due immagini sono simili in termini di caratteristiche visive piuttosto che pixel per pixel. Più è vicino a uno, migliore è la qualità dell'immagine

Roba

Definition 7.2.5

Sia A una matrice $M \times N$, definiamo la funzione $\mathcal{L} : \mathbb{R}^{M \times N} \to \mathbb{R}^{N*M}$ come segue:

$$\mathcal{L}(A)$$

è un vettore di dimensione $N * M$, le cui:

- prime N componenti sono la prima riga di A, letta da sinistra a destra;
- seconde N componenti sono la seconda riga di A, letta da sinistra a destra;
- ...
- M-esime N componenti sono la M-esima riga di A, letta da sinistra a destra.

Analogamente si definisce la conversione lessicografica per colonne.

7.2.3 Regolarizzazione di Tikhonov applicata alle immagini

Durante la soluzione di Naive si considerava il problema delle immagini come il seguente problema di minimo:

$$\min_x \|\|_2^2$$

Tuttavia dato che si ha un errore y^δ è necessario stabilizzare la funzione col metodo di regolarizzazione di Tikhonov:

$$\min_x \|Ax - y^\delta\|_2^2 + \lambda \|x^2\|_2^2$$

Dove λ è il parametro di Regolarizzazione. Applicando le condizioni del primo ordine $\nabla(f) = 0$ si ha:

$$(A^T A + \lambda I)x = A^T y^\delta$$

È possibile quindi risolvere il problema tramite il metodo CGLS

Regolarizzazione con Variazione totale

Una funzione di regolarizzazione alternativa a quella di Tikhonov e molto utilizzata nell'imaging è la funzione di Variazione Totale (TV) definita in questo modo:

$$TV(x) = \|\nabla(x)\|_1 = \sum_{i=1}^{M} \sum_{j=1}^{n} \sqrt{(x_{i+1,j} - x_{i,j})^2 + (x_{i,j+1-x_{i,j}})^2}$$

La funzione TV non è differenziabile nel punto $(0, 0)$. Per ottenre la differenziabilità, si inserice un piccolo parametro $\beta > 0$:

$$TV^\beta(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} \sqrt{(x_{i+1,j} - x_{i,j})^2 + (x_{i,j+1} - x_{i,j})^2 + \beta^2}$$

Dove i valori di solito utilizzati per β sono nell'ordine di 10^{-3}
Il problema di regolarizzazione diventa quindi:

$$\min_x \|Ax - y^\delta\|_2^2 + \lambda TV^\beta(x)$$

Il metodo di regolarizzazione con TV ha il vantaggio di essere particolarmente efficace nell'eliminare il rumore e allo stesso tempo meglio preservare i contorni, anche in caso di basso rapporto segnale/rumore. Il suo uso è motivato dalla sua abilità nel recuperare le discontinuità nell'immagine, inoltre preserva i bordi dell'immagine rimuovendo piccoli dettagli come il rumore. All'aumentare del parametro di regolarizzazione l'immagine tende a una immagine costante a tratti