SAPIENZA
UNIVERSITÀ DI ROMA

Appunti di Calcolo Numerico

Facoltà di Scienze MM.FF.NN
Corso di laurea in Informatica

Docente
Proff.ssa Biancamaria Della Vecchia

A.A. 2010/2011

Indice

Capitolo 1 - Introduzione al calcolo numerico

Analizziamo alcuni esempi di utilizzo del calcolo numerico.

1.1 - Il motore di ricerca Google

Google è il motore di ricerca più utilizzato al mondo, i servizi offerti da Google si estendono a mappe, documenti, GPS, sistemi operativi ecc e la maggior parte di questi servizi sono gratuiti.

La domanda sorge spontanea: Chi paga questi servizi?

Risposta: La pubblicità.

Quanto più il motore di ricerca è affidabile, tanto più gli utenti sono incoraggiati a usarlo e a soddisfare i propri bisogni, spendendo soldi. Dal lato opposto ci sono i produttori che investendo nella visibilità offerta dal motore di ricerca, offrono i loro prodotti sul web.

Ci possiamo rendere conto quanto sia straordinario tale meccanismo.

Il concetto fondamentale di Google è molto semplice:

Tutto ciò che per gli altri rappresenta un costo per Google è una risorsa.

Tramite il motore di ricerca, Google ottiene informazioni dagli utenti (cosa cercano, cosa comprano, gusti, mode ecc) e tramite queste informazioni può proporre la pubblicità in modo appropriato a ogni singolo utente. In altre parole Google investe conoscendo gusti, preferenze e attitudini degli utenti e propone sotto forma di pubblicità ciò che gli utenti cercano.

La quantità d'informazioni che Google deve gestire è enorme, il numero di pagine disponibile sul web è dell'ordine dei miliardi, e per fornire un buon motore di ricerca tali pagine devono essere conosciute.

Sorgono quindi un problema di ricerca delle pagine e la necessità di una straordinaria banda di rete e di enormi centri di calcolo.

Il funzionamento di un motore di ricerca è molto semplice, s'inseriscono delle parole chiave e il motore di ricerca restituisce i documenti che meglio rispondono alle chiavi inserite.

L'ordine delle pagine restituite è chiamato rank e rappresenta l'importanza di queste rispetto alla ricerca eseguita.

I vecchi motori di ricerca erano di tipo deterministico, le pagine web contenevano metadati cioè informazioni sulle stesse ed erano trovate incrociando metadati e contenuti.

```
<META NAME="keywords" CONTENT="">
<META NAME="description" CONTENT="">
```

Il sistema non funzionava, infatti, se all'interno dei metadati o nei contenuti era inserita più volte una determinata parola, il motore di ricerca attribuiva alla pagina un rank più alto rispetto a un'altra che conteneva la stessa parola meno volte.

E' così nata una vera disputa tra i produttori di beni e servizi, il cui scopo è ottenere rank più alti e i creatori dei motori di ricerca intenzionati a restituire rank veritieri o a favorire chi paga per ricevere maggiore visibilità.

La vera rivoluzione attuata da Google nella gestione delle informazioni è attribuirà al software, in particolare all'utilizzo di un modello statistico per individuare le pagine web, e ancora di più all'uso di una matrice per gestire ricerche e rank.

Le righe della matrice rappresentano le pagine web, mentre le colonne rappresentano le parole, ogni cella della matrice è riempita con 1 (uno) o con 0 (zero) a seconda se la pagina contiene o no la parola.

		1 Sapienza	2 Facoltà	3 Biblioteche	4 Calcolo	5 Numerico	6 Matlab	7 CNR	8 Bioinformatica	
1	http://w3.uniroma1.it	1	1	1		0	0	0	0	
2	http://www.na.iac.cnr.it/~bdv/	1	0	0	1	1	1	0	0	
3	http://www.cnr.it/sitocnr/home.html	0	0	0	0	0			1	1
...										

Con tale metodo sparisce il problema citato della ripetizione delle parole.

La ricerca di un utente è rappresentata come un vettore rispetto alle colonne della matrice.

Esempio:

$$\text{Calcolo numerico} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

La moltiplicazione tra la matrice e questo vettore di ricerca restituisce un vettore colonna che rappresenta il rank delle pagine.

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

La pagina a rank più alto è quella con valore maggiore.

Il problema principale di tale sistema è il numero di parole e di pagine esistenti, le prime sono dell'ordine del le migliaia, le seconde dei miliardi, che generano una matrice fortemente rettangolare poco utile in campo matematico.

Se si vuole risolvere un sistema del tipo $A\,x = b$, potrebbe essere necessario cal colare l'inversa della matrice A.

Per calcolare l'inversa di una matrice questa deve essere quadrata e a rango pieno ovvero le righe (colonne) devono essere diverse e non deve essere possibile ricav are una riga (colonna) dalle altre tramite un a combinazione lineare. Le righe (col onne) devono essere linearmente indipendenti.

Si può dimostrare che il rango di una matrice N x M è al più il minimo tra N e M, quindi i l rango non è superiore al numero di colonne, in altre parole il numero d'informazioni indipendenti contenute all'interno di tutte l e pagine web è al p iù pari al numero di parole del vocabolario.

Anche il numero di parole real mente usate è un sottoinsieme molto limitato dell'insieme delle parole e inoltre sono presi in conside razione i sinonimi (esame-appello, programma-software ecc..), avvien e dunque un'astrazione de lla singola parola e l'obiettivo è centrato sul concetto che l'utente vuole esprimere. Dalla ri chiesta si estraggono i concetti base e la medesima operazione è eseguita sulle pagine web, infine s'incrociano i concetti ottenuti.

La matrice originale viene cosi ridotta a d ecine di migliaia di el ementi contro i miliar di originali.

Per eseguire questa riduzione servono strumenti molto sofisticati ed è qui che il calcolo numerico subentra fornendo la tecnologia necessaria per operare sulle matrici.

1.2 - Il formato JPG

Il funzionamento di una macchina fotografica digitale si basa anch'essa sull'utilizzo di una matrice.

La matrice è formata da sensori di silicio (trasduttori), che catturano la luce esterna che impatta su ogni punto denominato pixel. Ogni singolo sensore converte la quantità di luce sul pixel in un segnale elettrico.

Vista la necessità di produrre immagini a colori si può pensare di avere tre sensori su ogni singolo pixel uno per la componente verde, uno per quella rossa e uno per quella blu.

Esistono diversi tipi di formati per un'immagine, uno dei più utilizzati è il JPG.

Esiste anche un formato RAW che restituisce la matrice di pixel, e per ogni pixel l'intensità di luce misurata.

Una normale macchina fotografica digitale offre all'incirca quattordici megapixel e i pixel all'interno della matrice sono disposti in maniera equidistante (in futuro si pensa di progettare macchine fotografiche con pixel disposti a caso all'interno della matrice).

Quattordici megapixel sono quattordici milioni di pixel che vanno moltiplicati per le tre componenti, per un totale di cinquantadue milioni d'informazioni da codificare.

Per codificare ogni componente di un singolo pixel si utilizza un byte (8 bit) rappresentando ogni singola componente con un numero da 0 (zero) a 28-1=255.

Lo 0 (zero) rappresenta l'assenza di colore, mentre il 255 la saturazione del colore.

Componente rossa: [0-255]

Componente verde: [0-255]

Componente blu: [0-255]

Ogni pixel sarà rappresentato con una terna (x.y.z) dove ogni elemento rappresenta una singola componente.

La combinazione (0.0.0) rappresenta il colore nero.

La combinazione (255.255.255) rappresenta il colore bianco.

La combinazione (255.0.0) rappresenta un rosso vivo.

I colori giallo, magenta e ciano rappresentano i complementari delle tre componenti e sono ottenuti ponendo a zero una delle tre.

Tale rappresentazione comporta l'utilizzo di cinquantadue Megabyte per ogni immagine, ma utilizzando la compressione JPEG si riduce il file a qualche megabyte ottenendo un notevole risparmio di spazio sulle memorie di massa e un grande risparmio di tempo durante il trasferimento dei file in rete.

La compressione consiste, di fatto, nell'eliminazione delle informazioni ridondanti, ovvero in una riduzione della matrice che permette di gestire solo le informazioni essenziali, che sono dell'ordine delle decine di migliaia, con la conseguente perdita di alcuni dettagli dell'immagine originale.

Il procedimento matematico alla base della trasformazione della matrice è la trasformata di Fourier, che può essere vista come una moltiplicazione della stessa per una nuova matrice. La matrice risultante è detta trasformazione e rappresenta la trasformazione del segnale originale in un nuovo spazio espresso con una base diversa, molto semplice, basata sulle funzioni trigonometriche seno e coseno.

1.3 - Il calcolo numerico

Ci sono tantissimi esempi di utilizzo del calcolo numerico nella chimica, fisica, nello sport, nell'ambiente e in molte applicazioni spesso anche di vitale importanza .

Un esempio è nel campo economico, dove è importante sapere se l'economia sta evolvendo verso una situazione di espansione o depressione o se si è vicini o lontani da una crisi economica.

Un altro importante esempio sono le previsioni meteo, fondamentali per l'agricoltura per stabilire i processi di coltivazione, ma anche per la navigazione aerea per tracciare rotte sicure e per molti altri campi. Le previsioni meteo utilizzano un modello matematico basato su importanti principi della fisica come la conservazione della massa, la conservazione della quantità di moto, i sistemi isolati, e la conservazione e dissipazione dell'energia.

In base a queste leggi sono descritti vari moti (fluidi, aria, oceani ecc) utilizzando equazioni all'apparenza molto semplici ma complicate da risolvere.

Il compito del calcolo numerico è trovare una soluzione approssimata, che non è esattamente quella giusta ma vicina e valutare l'errore dovuto all'approssimazione, capire cioè di quanto la soluzione trovata si discosti da quella esatta, determinando la qualità dell'approssimazione.

Problemi che non hanno soluzione analitica sono sostituiti con problemi equivalenti più semplici per i quali esiste una soluzione analitica calcolabile tramite computer, e tali che la soluzione trovata sia quanto più vicina a quella esatta.

Il calcolo numerico si divide in varie branche per la risoluzione di equazioni a derivate parziali, la risoluzione di equazioni differenziali, l'interpolazione e altre.

Capitolo 2 - Rappresentazione dei numeri nel computer

In breve:
Le procedure del calcolo numerico sono efficaci quando implementate su un calcolatore, così devono confrontarsi con il modo in cui la macchina codifica i numeri. E' importante conoscere cosa questo comporta, ad esempio per valutare correttamente l'output di un algoritmo.

2.1 – Numeri come stringhe binarie

In un computer le informazioni sono trattate in codifica binaria. Questo vincolo è imposto dalle stesse strutture del calcolatore che, nelle sue parti, manipola bit e sequenze di bit.

In quest' accezione un bit è la quantità di informazione disponibile in un apparato in grado di assumere uno solo di due possibili stati: due valori di tensione in un circuito, due quantità di elettroni in un condensatore, due valori del campo magnetico. Che sia nel bus di sistema, nei registri della scheda video o in un disco magnetico, le informazioni sono immagazzinate e manipolate come sequenze binarie.

Vediamo dunque come possiamo utilizzare una stringa di bit per rappresentare i numeri.

2.1.1 – Numeri naturali

Innanzitutto contiamo, dato il numero di bit, quanti diversi numeri possiamo rappresentare.

1bit {0, 1} ⇒ 2 posibilità

2bit {**0**0, **0**1, **1**0, **1**1} ⇒ 4 possibilità

3bit {**0**00, **0**01, **0**10, **0**11, **1**00, **1**01, **1**10, **1**11} ⇒ 8 possibilità

Ad ogni bit in più abbiamo il doppio delle possibilità precedenti, costituite dalla concatenazione del nuovo bit (segnato in grassetto) assieme ad ognuna delle configurazioni precedenti.

Dato che un bit ci da 2 possibilità ed ogni altro bit ne raddoppia il numero , per n

bit abbiamo 2^n possibilità.

Ad esempio una codifica comune per le immagini è quella di una matrice di terne di 8 bit, in cui ogni terna rappresenta i valori di intensità di un dato pixel nella sua composizione rgb.

$8\text{bit} = 2^8 = 256$ per ogni canale

$256^3 = 16.777.216$ sono i colori possibili per ogni pixel, ad un costo di 24 bit.

In un'archtettura a 32bit avremo, in singola precisione, 32 bit a disposizione per memorizzare un intero. Utilizziamo un bit per il segno e gli altri 31 per il valore, o utilizziamo la notazione a due per guagnare una cifra rappresentabile in più. In entrambi i casi il numero più grande rappresentabile sarà $2^{31} - 1 = 2.147.483.647$. Non una gran cifra se pensiamo al numero di Avogadro, al debito pubblico o al numero di stelle della Via Lattea. Inoltre questa codifica non tiene conto dei numeri reali. Vediamo allora una rappresentazione in grado di superare questi limiti.

2.1.2 – Numeri reali

Utilizziamo una codifica analoga alla notazione scientifica (ex: $1{,}56234 \cdot 10^{29}$).

$a = m \cdot B^s$

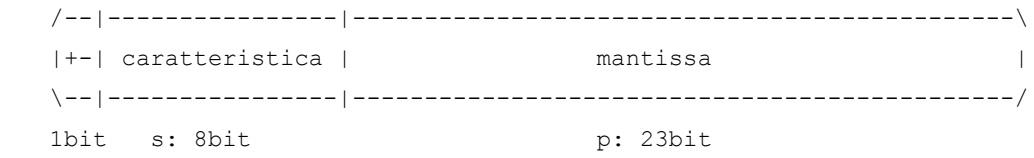
$B : base$

$s : caratteristica, \quad s \in \mathbb{N}$

$m : mantissa, \quad \frac{1}{B} < m < 1$

Questa rappresentazione prende il nome di codifica in **virgola mobile**.

Fissiamo $B = 2$, utilizziamo 8 bit per la caratteristica (s) e 23 bit per la mantissa (p). Un bit terrà conto del segno.



Il valore della mantissa è calcolato secondo la seguente formula

(1) $\sum_{i=1}^{p} x_i \cdot 2^{-i}$

Il bit più significativo dunque vale 2^{-1} (ovvero $1/2$), il seguente 2^{-2} ($1/4$) poi 2^{-3} ($1/8$) fino al meno significativo che vale 2^{-p} .

La mantissa rappresenta la quantità di informazioni a disposizione per memorizzare le cifre significative di un numero. 2^{23} è circa uguale a 10^{7} , di conseguenza sono 7 le cifre (decimali) significative che questa codifica può memorizzare.

Esempio: memorizzando in questo modo il numero 0,1259875**947**, perdiamo le ultime tre cifre e, a tutti gli effetti, il numero sarà memorizzato come 0,1259875. Nel caso del numero 543,1293**87467** la relativa mantissa sarà 0,5431293.

2.1.3 – Range dei numeri in virgola mobile

E' interessante notare quale sia il range dei numeri codificabili in virgola mobile al variare di s , la caratteristica. Per differenti valori di s calcoliamo il valore dei numeri associati alle seguenti mantisse:

1000.....000	0000.....001	1111.....111
$x_i = \begin{cases} 1, & i=1 \\ 0, & V_i > 1 \end{cases}$	$x_i = \begin{cases} 1, & i=p \\ 0, & V_i < p \end{cases}$	$x_i = 1, \forall i$

Ricordando **(1)** notiamo che il valore della prima mantissa è semplicemente 2^{-1} . Il numero relativo vale dunque $2^{-1} \cdot 2^{s} = 2^{s-1}$

La seconda mantissa vale banalmente 2^{-p} e il numero corrispondente vale 2^{s-p} .

Per la terza mantissa abbiamo bisogno di alcuni calcoli.

Tutti i bit valgono 1, dunque il valore corrisponde alla seguente sommatoria

$$\sum_{i=1}^{p} 1 \cdot 2^{-i} = 2^{-1} + 2^{-4} + 2^{-8} + \ldots + 2^{-p}$$

mettiamo in evidenza 2^{-p}

$2^{-p}\left(1+2+2^2+2^3+\ldots+2^{p-1}\right)$ *(nota: i termini della sommatoria sono scritti nell'ordine contrario)*

Il valore della sommatoria tra parentesi è noto. E' il valore del massimo intero rappresentabile con un numero p di bit, dunque 2^p-1 . Continuiamo:

$2^{-p}\left(2^p-1\right)$

$2^{-p}\cdot 2^p-2^{-p}$

$1-2^{-p}$

Il valore del relativo numero dunque è $2^s\left(1-2^{-p}\right)$

Inseriamo le formule nella seguente tabella e analizziamone i valori al variare della caratteristica.

Mantissa (p)	1000……000 $x_i=\left\{\begin{array}{ll}1, & i=1\\ 0, & V_i>1\end{array}\right\}$	0000……001 $x_i=\left\{\begin{array}{ll}1, & i=p\\ 0, & V_i<p\end{array}\right\}$	1111……111 $x_i=1, \forall i$
Caratteristica (s)	2^{s-1}	2^{s-p}	$2^s\left(1-2^{-p}\right)$
0	1/2	2^{-p}	$1-2^{-p}$
1	1	2^{1-p}	$2-2^{1-p}$
2	2	2^{2-p}	$4-2^{2-p}$
3	4	2^{3-p}	$8-2^{3-p}$

Siamo soliti utilizzare tutte le configurazioni da 0000……001 a 1111……111. Ma così facendo sperimentiamo una ridondanza delle rappresentazioni possibili, dunque uno spreco di risorse. Per verificarlo osservare il seguente schema:

```
s    0-----1-----2-----3-----4-----5-----6-----7-----8
0    |-----|
1     |----------|
2      |----------------------|
3       |-----------------------------------------|
```

Notiamo invece cosa accade utilizzando le configurazioni da 1000……000 a 1111……111

```
s    0-----1-----2-----3-----4-----5-----6-----7-----8
0       |--|
1          |-----|
2                |-----------|
3                           |-----------------------|
```

In questo modo abbiamo una buona copertura del campo dei Reali senza sovrapposizione di rappresentazione.

Possiamo così immaginare di utilizzare una codifica che interpreti i valori tra 1000…..0000 e 1111……1111 al variare di s per rappresentare i numeri Reali. Questo ci permette un'ulteriore piccola ottimizzazione: dal momento che tutte le mantisse iniziano con 1 possiamo rendere implicita questa informazione ed avere una mantissa più lunga di un bit.

Infine notiamo che i numeri compresi tra 0 e 1/2 sono rappresentati per valori negativi della caratteristica, ed il più piccolo numero possibile con questa codifica sarà $2^{-128-1}=2^{-129}=1.46936794\cdot 10^{-39}$, un numero molto piccolo ma lo 0? Secondo quanto proposto, possiamo rappresentare un range di numeri molto ampio, ma non abbiamo una codifica per lo 0. Difatti una mantissa di tutti 0 assume il valore di 2^{-1} per via del bit implicito.

La soluzione consiste nell'adozione di codici speciali per rappresentare sia lo 0 che altri valori utili come quelli di overflow e underflow ($+\infty$ e $-\infty$), not, null ed altri ancora.

Capitolo 3 - L'interpolazione

3.1 - Interpolazione polinomiale (o di Lagrange)

Date $n+1$ coppie di punti distinti ed una funzione approssimante $g(x)$ diremo che g interpola i punti dati se $g(x_i) = y_i$ per $i = 0,\dots,n$, cioè se il grafico di g passa per i punti dati. Se la funzione g è un polinomio parleremo di interpolazione polinomiale o Lagrangiana, se g è una funzione razionale parleremo di interpolazione razionale, se g è una funzione trigonometrica parleremo di interpolazione trigonometrica etc.

Vediamo alcuni esempi:

Supponiamo di avere due punti (x_0, y_0) e (x_1, y_1) e di voler determinare la retta che passa per essi; per fare ciò basterà risolvere il seguente sistema

$$\begin{cases} P_1(x_0) = ax_0 + b = y_0 \\ P_1(x_1) = ax_1 + b = y_1 \end{cases}$$

Analogamente, dati 3 punti, è possibile costruire la parabola che passa per essi risolvendo il sistema:

$$\begin{cases} P_2(x_0) = ax_0^2 + bx_0 + c_0 = y_0 \\ P_2(x_1) = ax_1^2 + bx_1 + c_1 = y_1 \\ P_2(x_2) = ax_2^2 + bx_2 + c_2 = y_2 \end{cases}$$

Più in generale, date $n+1$ coppie distinte di punti $(x_o, y_0), (x_1, y_1), \dots, (x_n, y_n)$ è possibile costruire il polinomio che passa per essi trovando la soluzione del sistema:

$$\begin{cases} P_n(x_0) = a_0 + a_1x_0 + a_2x_0^2 + a_3x_0^3 + \cdots + a_nx_0^3 = y_0 \\ P_n(x_1) = a_0 + a_1x_1 + a_2x_1^2 + a_3x_1^3 + \cdots + a_nx_1^3 = y_1 \\ \vdots \\ P_n(x_n) = a_0 + a_1x_n + a_2x_n^2 + a_3x_n^3 + \cdots + a_nx_n^3 = y_n \end{cases}$$

Si dimostra che se i nodi di interpolazione sono distinti, il determinante della matrice dei coefficienti è diverso da 0 (metodo dei coefficienti indeterminati) e il sistema ammette un'unica soluzione (quindi il polinomio interpolante di Lagrange è unico).

La matrice dei coefficienti associata a questo sistema

$$\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix}$$

è però una matrice di Van der Monde, ed è una matrice particolare in quanto fortemente mal condizionata: per valori di n molto grandi la risoluzione del sistema porta a risultati poco affidabili e quindi si preferisce utilizzare altri metodi per determinare il polinomio interpolante.

3.2 - Polinomi fondamentali di Lagrange

Una possibile soluzione al problema è quella di scrivere il polinomio mediante interpolante mediante i polinomi di Lagrange:

$$L_n(x) = \sum_{k=0}^{n} l_{n,k}(x) \cdot y_k$$

dove

$$l_{n,k}(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{x - x_i}{x_k - x_i} = \frac{(x - x_0) \cdots (x - x_{k-1})(x - x_{k+1}) \cdots (x - x_n)}{(x_k - x_0) \cdots (x_k - x_{k-1})(x_k - x_{k+1}) \cdots (x_k - x_n)}$$

Da notare che per come sono definiti questi polinomi si ha che:

$$l_{n,k}(x) = \begin{cases} 1 & se\ j = k \\ 0 & se\ j \neq k \end{cases}$$

Questa rappresentazione è soggetta al fenomeno della cancellazione numerica e quindi è instabile in quanto per valori di x_k e x_j molto vicini tra loro si ottengono divisioni per numeri prossimi allo zero.

3.3 - Rappresentazione di Newton (o metodo delle differenze divise)

Quando i punti da interpolare sono numerosi, invece del sistema converrà adottare il metodo Newton (o delle differenze divise) che non necessita del calcolo dei determinanti.

Questa rappresentazione consente di trattare anche il caso di nodi equidistanti e vicini. Per presentare questa nuova rappresentazione occorre introdurre delle nuove quantità denominate *differenze divise*.

Siano {*a,b,c, …,n*} punti o nodi assegnati,definiamo le seguenti quantità:

Differenza divisa (su due punti) di ordine 1:

$$[a, b; f] = \frac{f(b) - f(a)}{b - a}$$

Differenza divisa (su tre punti) di ordine 2:

$$[a, b, c; f] = \frac{[b, c; f] - [a, b; f]}{c - a}$$

Differenza divisa (su quattro punti) di ordine 3:

$$[a, b, c, d; f] = \frac{[b, c, d; f] - [a, b, c; f]}{d - a}$$

ed in generale quella di ordine n:

$$[a, b, \dots, n; f] = \frac{[b, c, \dots, n; f] - [a, b, \dots, n; f]}{n - a}$$

Si definisce rappresentazione di Newton del polinomio interpolante la seguente espressione:

$$L_n(P_i x) = y_0 + (x - x_0)[x_0, x_1; f] + (x - x_0)(x - x_1)[x_0, x_1, x_2; f] + \cdots$$
$$+ (x - x_0) \dots (x - x_{n-1})[x_0, \dots, x_n; f]$$

dove ciascuna delle espressioni tra parentesi quadre è la differenza divisa sui punti $x_0, \dots, x_n$.

3.3.1 - Perché è più comoda questa rappresentazione?

La definizione di differenza divisa $[a, b; f]$, per valori di b molto vicini ad a $(b \to a)$, coincide con la definizione di limite del rapporto incrementale ovvero:

$$f'(a) = \lim_{x \to a} \frac{f(b) - f(a)}{b - a}, \qquad rapporto\ incrementale$$

Se a, b, c coincidono tra loro, allora

$$[a, b, c; f] = \frac{f''(a)}{2!}$$

Se a, b, c, d coincidono tra loro, allora

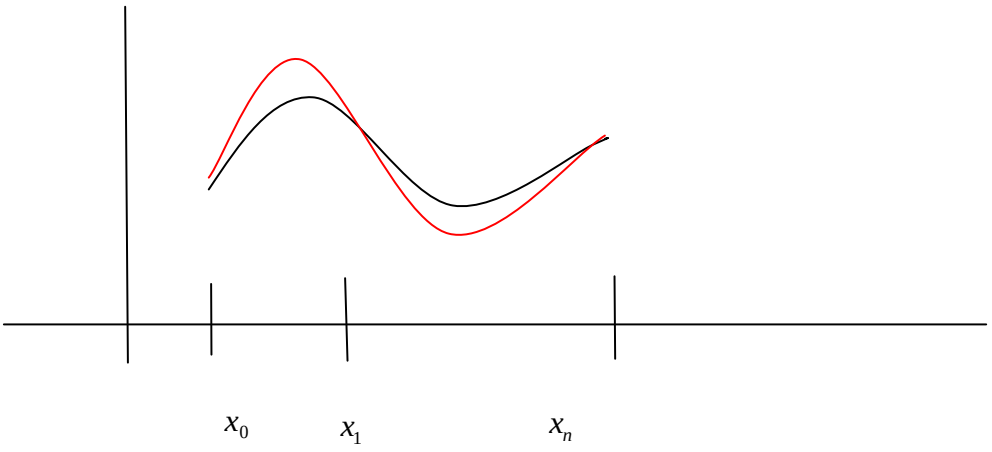
$$[a, b, c, d; f] = \frac{f'''(a)}{3!}$$

più in generale, definiamo:

$$\underbrace{[a, b, c, \dots, n; f]}_{k+1} = \frac{f^{(k)}(a)}{k!}$$

A patto sempre che la funzione sia derivabile.

3.3.2 - Errore o resto dell'interpolante di Lagrange (cosa succede nei punti diversi da x_0, x_1, x_2, x_n)



data una successione di punti, vogliamo stabilire quanto la funzione interpolante si discosta dalla funzione di partenza,

cioè:

$$|L_n(f;x) - f(x)| = errore = |R_n(f;x)|$$

che rappresenta di quanto si scosta l'interpolante dalla funzione.

L'espressione $R_n(f;x)$ viene chiamata errore o resto dell'interpolazione Lagrangiana.

Nei punti $x_0, x_1, \dots, x_n$ sappiamo che la funzione coincide con il polinomio interpolante, ma non sappiamo come va la funzione tra un punto ed un altro. La teoria può darci una maggiorazione:

$$|R_n(f;x)| \leq C \cdot (1 + \lambda_n) \cdot E_n(f)$$

Dove:

$E_n(f) =$ errore di migliore approssimazione uniforme

$$E_n(f) = \min_{P \in \wp^n} \max_{[x_0, x_n]} |f(x) - P(x)|$$

cioè scelgo fra tutti i polinomi in P_n quello che si discosta meno da f.

Inoltre la possiamo riscrivere come:

$$E_n(f) = \max |f(x) - P^*(x)|$$

Dove P^* è il polinomio di migliore approssimazione uniforme.

In più, E_n convergerà a zero ($E_n(f) \to 0$) tanto più la funzione f è "*smooth*" cioè regolare (derivabile più volte).

In particolare per funzioni "buone" $E_n(f)$ tende a zero come $1/n^2$ oppure $1/n^3$ quindi $E_n(f)$ va più velocemente a zero quante più derivate ha la funzione f (ad esempio se f ha una derivata continua, allora $E_n(f)$ è minore o uguale a C/n ,con C una costante; se f ha due derivate continue allora $E_n(f)$ è minore o uguale a C/n^2 , se f ha tre derivate continue, allora $E_n(f)$ è minore o uguale a C/n^3, e così via …).

Mentre:

$$\lambda_n = \max_{[x_0, x_n]} \sum_i \left| l_{n,k}(x) \right|, \qquad (costanti\ di\ Lebesgue)$$

dove $l_{n,k}$ sono i polinomi fondamentali di Lagrange

$[x_0, \dots, x_n]$ = nodi interpolanti (punti scelti)

Le quantità λ_n invece tendono ad infinito al crescere di n, ed in particolare si ha che

$$\lambda_n \geq C \cdot \log n$$

Per poter bilanciare il buon comportamento di $E_n(f)$ rispetto al cattivo comportamento di λ_n, affinché $R_n(f)$ tenda a 0, bisogna operare delle opportune scelte per i punti x_i del polinomio interpolante. In particolare se:

x_i = nodi di Chebischev $\Rightarrow\ \lambda_n \geq C \cdot \log n$ (scelta migliore)

x_i = nodi di polinomi ortogonali $\Rightarrow\ \lambda_n \geq C \cdot n^{\alpha}$

x_i = nodi equidistanti $\Rightarrow \lambda_n \geq C \cdot e^n$ (scelta peggiore)

Esempio scelta peggiore:

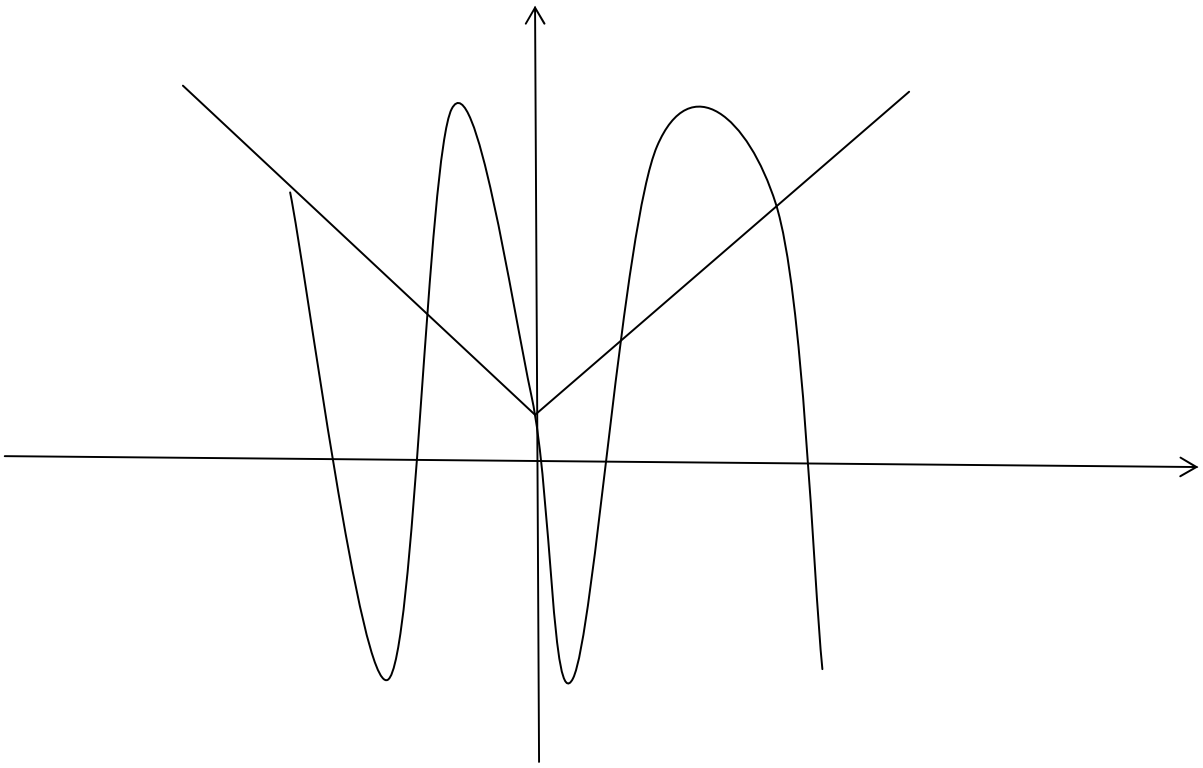
Studiamo la convergenza dell'interpolante di Lagrange scegliendo punti di interpolazione equidistanti.

$f(x) = |x| = \begin{cases} -x,\ per\ \ x < 0 \\ x,\ per\ \ x > 0 \end{cases}$ funzione continua ma non derivabile

$E_n(f) \to 0$ va a zero più lentamente di $1/n$

$\lambda_n \approx e^n$ crescita esponenziale

$E_n(f)\lambda_n = E_n(f)e^n$ nodi equidistanti (Questa quantità esplode)



I nodi equidistanti sono la scelta peggiore ma sono i più naturali da considerare perché spesso nei fenomeni vengono fatte misure regolari.

Esempio scelta migliore:

$$f(x) = x \cdot |x| = \begin{cases} -x^2, \ per \ x < 0 \\ x^2, \ per \ x > 0 \end{cases}$$

$$f'(x) = \begin{cases} -2x \\ 2x \end{cases} \qquad \text{in } x = 0 \text{ è derivabile}$$

La funzione è derivabile una volta

$$f \in \mathcal{C}' \qquad E_n(f) \approx \frac{1}{n}$$

x_i Cebischev $\lambda_n \approx c \cdot \log n$

$$E_n(f)\lambda_n \approx \frac{\log n}{n} \to 0$$

Questo è un caso in cui al crescere di n l'interpolante si schiaccia alla funzione e quindi approssima bene la funzione.

Capitolo 4 - L'approssimazione

I problemi dell'interpolazione e dell'approssimazione sono problemi ben distinti ma notevolmente correlati.
Per quanto riguarda il primo problema data una sequenza di n numeri reali x_k distinti e per ciascuno di questi un secondo numero y_k ci proponiamo di individuare una funzione di una certa famiglia tale che sia $f(x_k) = y_k$ per $k = 1, 2, \dots, n$. Nei problemi di approssimazione, anche detti di regressione o di adattamento di curve (curve fitting), si abbandona il vincolo che impone alla funzione interpolante di passare esattamente per i punti dati e si richiede solo una curva approssimante che avvicini il più possibile i punti dati.

4.1 - Definizione

Supponiamo che io conosca una funzione f in *n+1* punti (non sempre equidistanti), ossia conosco

$$f(x_o),\ f(x_1), \dots f(x_n)$$

Questi punti potrebbero essere stati ottenuti attraverso:

- misure sperimentali
- l'utilizzo di un algoritmo molto complesso il cui valore numerico è noto solo in alcuni punti discreti.

Quello che si vuole ottenere è di ricostruire la funzione per ogni valore di x nell'intervallo considerato.

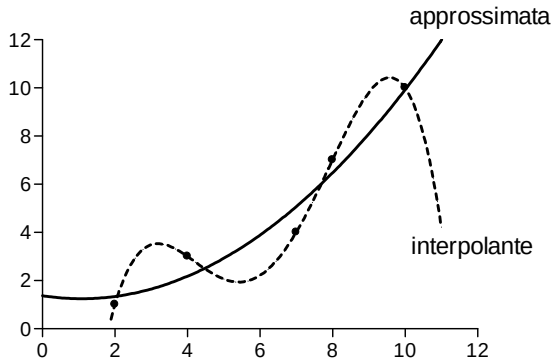


Figura 1. I cinque punti sono stati interpolati mediante il metodo di Lagrange (curva tratteggiata). La curva che approssima i punti (curva continua) è un polinomio di secondo grado.

Se si volesse ottenere una funzione che passi per tutti i punti dati, bisognerebbe utilizzare i metodi di interpolazione già descritti. Tuttavia in molte applicazioni pratiche (scienze sperimentali, scienze economiche, grafica …) non è richiesto che la curva passi per tutti i punti.

Per esempio, se i dati derivano da misure sperimentali, questi potrebbero essere affetti da fluttuazioni statistiche delle quali non si vuole tener conto. In altri casi può essere richiesto che la curva abbia un andamento dolce per ragioni estetiche. Bisogna infine tener conto che un problema generale dei metodi di interpolazione è quello di generare curve affette da fenomeni indesiderati di oscillazione.

Nel caso dell'approssimazione si vuole invece costruire la curva *g(x)* in modo da simulare bene l'andamento dei punti nel loro complesso.

La funzione *g(x)* deve avere le seguenti caratteristiche:

- Deve essere "semplice", ossia si deve poter operare facilmente su di essa (determinarne le soluzioni, la derivata, l'integrale, etc.). A tal fine la si può scegliere tra classi di funzioni semplici: polinomiali, trigonometriche, razionali, esponenziali.
- Non deve scostarsi troppo da *f(x)* nell'intervallo richiesto. Quindi l'errore $|f(x)-g(x)|$ deve essere ragionevolmente piccolo.

Possiamo distinguere due casi in base alle proprietà della funzione *f(x)*:

4.1.1 - Caso 1: approssimazione uniforme

Sia *f(x)* una funzione continua nell'intervallo richiesto:

$$f(x) \in C^o \text{ in } [a,b]$$

Per misurare l'errore $|f(x)-g(x)|$ si usa la norma infinito:

$$\|f(x)-g(x)\|_\infty = \max_{[a,b]} |f(x)-g(x)|$$

Il nostro scopo è quindi di trovare una *g(x)* che minimizzi il valore della norma infinito.

Nel caso dell'approssimazione polinomiale il problema è se sia sempre possibile costruire una successione di polinomi $\{P_n(x)\}$ che converga uniformemente in [*a,b*] alla funzione *f(x)*

$$\|f(x)-P_n(x)\|_\infty \overset{?}{\underset{n\to\infty}{\to}} 0$$

La soluzione a questo problema è data dal teorema di Weierstrass.

4.1.2 - Teorema di Weierstrass

Assegnata una funzione continua in un intervallo chiuso e limitato $[a,b]$, esiste almeno una successione di polinomi $P_n(x)$ convergente uniformemente verso $f(x)$ in $[a,b]$. Una qualsiasi successione di polinomi di questo tipo si chiama successione di polinomi approssimanti.

$$\exists P_n(x) \text{ tale che } \lim_{n\to\infty} \left\| f(x) - P_n(x) \right\|_\infty = 0$$

Esempio:

Un esempio di polinomio di questo tipo è il polinomio di Bernstein:

$$B_n(f,x) = \sum_{k=0}^{n} \binom{n}{k} \cdot x^k (1-x)^{n-k} \cdot f\left(\frac{k}{n}\right), \qquad per\ x \in [0,1]$$

Il polinomio di Bernstein ha trovato importanti applicazioni in ambito grafico con le curve di Bezier.

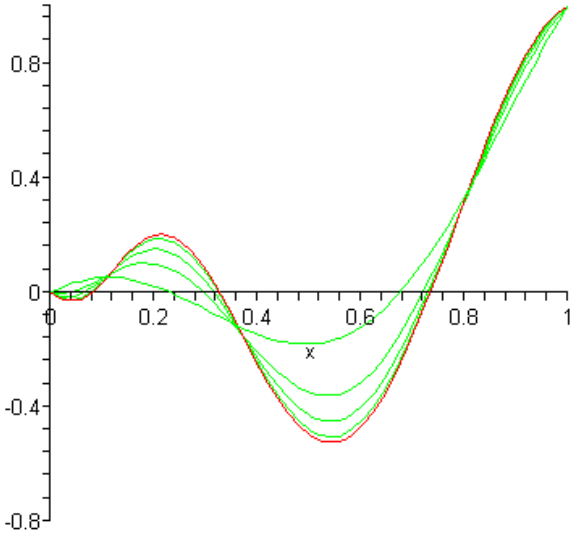


Figura 2. Polinomi di Bernstein che approssimano una curva. Come si può vedere, ad ogni iterazione le curve di Bernstein (curve verdi) si schiacciano sempre più sulla curva da approssimare (curva rossa).

4.1.3 - Caso 2: approssimazione in media quadratica

Sia *f(x)* una funzione quadrato integrabile non necessariamente continua:

$$\int_a^b f^2(x) \cdot dx < \infty$$

Per questa classe di funzioni si definisce lo scostamento della funzione approssimante *g(x)* dalla *f(x)* mediante la norma quadratica o errore quadratico medio:

$$\|f(x) - g(x)\|_2 = \sqrt{\int_a^b |f(x) - g(x)|^2 \cdot dx}$$

Analogamente al caso 1, anche qui si può dimostrare un teorema che garantisce l'esistenza della funzione approssimante *g(x)* nel caso che questa sia polinomiale:

Teorema

$$\exists P_n(x) \text{ tale che } \lim_{n\to\infty} \|f(x) - P_n(x)\|_2 = 0$$

Esempio:

E' possibile trovare delle funzioni $g_k(x)$ continue che formano un sistema ortonormale e, a partire da queste funzioni, introdurre delle approssimanti $f_n(x)$ definite come:

$$f_n(x) = \sum_{k=1}^{n} c_k \cdot g_k(x)$$

Si dimostra che le funzioni f_n si scostano il meno possibile dalla funzione *f(x)*, cioè rendono minima la norma quadratica:

$$\mu_n = \sqrt{\int_a^b \left| f(x) - \sum_{k=1}^{n} c_k \cdot g_k(x) \right|^2 dx}$$

a patto che i coefficienti c_k siano uguali ai coefficienti di Fourier a_k:

$$c_k = a_k = \int_a^b f(x) g_k(x) dx$$

4.2 - Approssimazione dei minimi quadrati

4.2.1 - Descrizione della tecnica

Supponiamo di avere $n + 1$ coppie di punti distinti $(x_0, y_0), \dots, (x_n, y_n)$ e di voler approssimare la successione di questi punti con la retta $y = a_0 + a_1 x$ tale che lo scostamento, cioè la distanza euclidea fra gli y_i ed i valori assunti dalla retta nei punti x_i sia minima. Questo equivale a calcolare i valori di $a_0,\ a_1$ e che minimizzano tale scostamento, ovvero

$$\min_{a_0,a_1} S = \min_{a_0,a_1} \sum_{i=1}^{n} (y_i - a_0 - a_1 x)^2$$

dove

$$S = \sum_{i=0}^{n} \left(y_i^2 + a_0^2 + a_1^2 x_i^2 - 2\, a_0 y_i - 2\, a_1 x_i y_i + 2\, a_0 x_i y_i \right)$$

Da ciò deriva il nome *minimi quadrati*. Ma perché non si utilizza la somma degli scarti anziché la somma dei loro quadrati? È facile vedere che la somma degli scarti non è adatta a quantificare l'aderenza della retta agli *n* punti. Gli scarti possono essere infatti positivi o negativi e la loro somma può essere piccola in valore assoluto anche per rette palesemente inadatte a descrivere gli *n* punti. Per esempio, siano dati i tre punti allineati (1,1), (2,2), (3,3). Ovviamente la miglior retta è $y = x$; la somma degli scarti è nulla. Ma è nulla anche per qualunque retta passi per (2,2), quindi di equazione $y = m\,(x - 2) + 2$. Dunque ci servono scarti che siano misurati da valori positivi. Inoltre la somma dei quadrati anziché dei valori assoluti si sposa in modo naturale con la *media aritmetica*: la media aritmetica di una sequenza di numeri gode della proprietà di rendere minima la somma dei quadrati degli scarti.

Ritornando al calcolo dei coefficienti a_0, a_1 dall'analisi sappiamo che il minimo di una unzione a due variabili si ottiene derivando rispetto alle singole variabili, ponendo quindi

$$\begin{cases} \dfrac{\partial S}{\partial a_0} = -2 \displaystyle\sum_{i=1}^{n} (y_i - a_0 - a_1 x_i) = 0 \\ \dfrac{\partial S}{\partial a_1} = -2 \displaystyle\sum_{i=1}^{n} (y_i - a_0 - a_1 x_i) \cdot x_i = 0 \end{cases}$$

essendo le due quantità nulle otteniamo

$$\begin{cases} N \cdot a_0 + \sum x_i \cdot a_1 = \sum y_i \\ \sum x_i \cdot a_0 + \sum x_i^2 \cdot a_1 = \sum x_i y_i \end{cases}$$

La cui soluzione è quindi

$$\begin{cases} a_0 = \dfrac{\sum x_i^2 \cdot \sum y_i - \sum x_i \cdot \sum x_i y_i}{N \cdot \sum x_i^2 + (\sum x_i)^2} \\ a_1 = \dfrac{N \cdot \sum x_i y_i - \sum x_i \cdot \sum y_i}{N \cdot \sum x_i^2 + (\sum x_i)^2} \end{cases}$$

in questo modo ricaviamo i coefficienti a_0 e a_1 del polinomio di primo grado e possiamo tracciare la retta che passa per i punti, detta retta ai minimi quadrati

4.2.2 - Generalizzazione all'approssimazione con polinomi di grado M arbitrario

Analogamente possiamo approssimare la successione dei punti con un polinomio di grado M della forma

$$y = a_0 + a_1 x + a_2 x^2 + \ldots + a_M x^M$$

per fare questo basta calcolare i coefficienti $a_0, \ldots, a_M$ della seguente equazione

$$S = \sum_{i=0}^{n} \left(y_i - a_0 - a_1 x - a_2 x_i^2 - \ldots - a_M x_i^M\right)^2$$

imponendo che tutte le sue derivate parziali siano uguali a 0

$$\begin{cases} \dfrac{\partial S}{\partial a_0} = -2 \cdot \sum_{i=1}^{n} \left(y_i - a_0 - a_1 x - a_2 x_i^2 - \ldots - a_M x_i^M\right) = 0 \\ \dfrac{\partial S}{\partial a_1} = -2 \cdot \sum_{i=1}^{n} \left(y_i - a_0 - a_1 x - a_2 x_i^2 - \ldots - a_M x_i^M\right) \cdot x_i = 0 \\ \dfrac{\partial S}{\partial a_2} = -2 \cdot \sum_{i=1}^{n} \left(y_i - a_0 - a_1 x - a_2 x_i^2 - \ldots - a_M x_i^M\right) \cdot x_i^2 = 0 \\ \vdots \qquad\qquad \vdots \qquad\qquad \vdots \\ \dfrac{\partial S}{\partial a_M} = -2 \cdot \sum_{i=1}^{n} \left(y_i - a_0 - a_1 x - a_2 x_i^2 - \ldots - a_M x_i^M\right) \cdot x_i^M = 0 \end{cases}$$

Otteniamo alla fine il seguente sistema di equazioni

$$\begin{cases} N \cdot a_0 + \sum x_i \cdot a_1 + \sum x_i^2 \cdot a_2 + \cdots + \sum x_i^M \cdot a_M = \sum y_i \\ \sum x_i \cdot a_0 + \sum x_i^2 \cdot a_1 + \sum x_i^3 \cdot a_2 + \cdots + \sum x_i^{M+1} \cdot a_M = \sum x_i y_i \\ \sum x_i^2 \cdot a_0 + \sum x_i^3 \cdot a_1 + \sum x_i^4 \cdot a_2 + \cdots + \sum x_i^{M+2} \cdot a_M = \sum x_i^2 y_i \\ \vdots \quad \vdots \quad \vdots \quad \ddots \quad \vdots \quad \vdots \\ \sum x_i^M \cdot a_0 + \sum x_i^{M+1} \cdot a_1 + \sum x_i^{M+2} \cdot a_2 + \cdots + \sum x_i^{2M} \cdot a_M = \sum x_i^M y_i \end{cases}$$

che possiamo riscrivere in forma compatta come

$$H \cdot \bar{a} = \bar{b}$$

dove H è la matrice dei coefficienti

$$H = \begin{bmatrix} N & \sum x_i & \cdots & \sum x_i^M \\ \sum x_i & \sum x_i^2 & \cdots & \sum x_i^{M+1} \\ \sum x_i^2 & \sum x_i^3 & \cdots & \sum x_i^{M+2} \\ \vdots & \vdots & \ddots & \vdots \\ \sum x_i^M & \sum x_i^{M+1} & \cdots & \sum x_i^{2M} \end{bmatrix}$$

$\bar{a}$ è il vettore delle incognite

$$\bar{a} = (a_0, a_1, a_2, \dots, a_M)$$

e $\bar{b}$ è il vettore dei termini noti

$$\bar{b} = \left(\sum y_i, \sum x_i y_i, \sum x_i^2 y_i, \dots, \sum x_i^M y_i\right)$$

In generale è possibile estendere questo metodo ad un qualsiasi polinomio di grado r, tuttavia tale approccio è fortemente sconsigliato per polinomi di grado superiore a 2, in quanto porta alla risoluzione di un sistema di equazioni lineari malcondizionato e conseguentemente ad una soluzione poco affidabile.

Capitolo 5 - Quadratura Numerica

In breve:
La quadratura numerica è un metodo per il calcolo approssimato di integrali . I metodi per effettuare la quadratura numerica si dividono in due macrocategorie: le formule di Newton-Cotes e le formule di Gauss.

5.1 - Generalità

Il problema considerato dalla quadratura numerica è quello di calcolare una soluzione approssimata di un integrale definito. Questo approccio si rivela utile nei casi in cui l'integrazione risulti complicata o impossibile da calcolare attraverso metodi analitici, occorrenza frequente nelle applicazioni reali. L'obiettivo è costruire una opportuna *formula* o *somma di quadratura* tale che

$$\int_a^b f(x)\,dx \simeq \sum_{k=1}^{n} w_k \; f(x_k)$$

dove le x_k nelle quali viene valutata la funzione integranda prendono il nome di *nodi* e i coefficienti w_k di *pesi* della formula di quadratura.

La scelta ottimale di nodi e pesi è tutt'oggi oggetto di studio. Se al crescere del numero di nodi n la somma del modulo dei pesi è limitata, la quadratura converge all'integrale:

$$\sum_{k=1}^{n} (w_k) \leq C \Longrightarrow \lim_{n\to\infty} \sum_{k=1}^{n} w_k f(x_k) = \int_a^b f(x)\,dx$$

I metodi di quadratura numerica possono essere distinti in due categorie

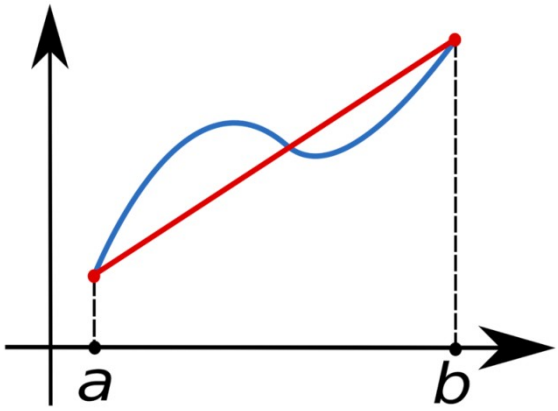
- Formule di Newton-Cotes
- Formule di Gauss

5.2 - Formule di Newton-Cotes

Le formule di quadratura di Newton-Cotes sono utilizzabili nel caso in cui la funzione integranda sia valutata in nodi equidistanti. Sono generalmente costruite su pochi nodi, dal momento che per n alti la somma del modulo dei pesi tende ad infinito e le formule non convergono all'integrale.

5.2.1 - Formula dei Trapezi

L'idea alla base di questo metodo è approssimare l'integrale di $f(x)$, cioè l'area sottesa dalla curva della funzione $f(x)$, con l'area del trapezio di vertici $(a,0),(b,0),(a,f(a)),(b,f(b))$

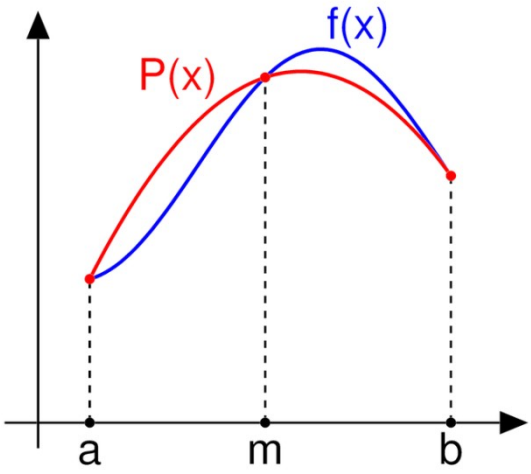


ossia $\int_a^b f(x)\ dx \simeq \frac{b-a}{2}\left(f(a)+f(b)\right)$.

È evidente come, al prezzo della semplicità di calcolo, venga sacrificata l'accuratezza dell'approssimazione.

5.2.2 - Formula di Cavalieri-Simpson

La regola di Cavalieri-Simpson prevede l'approssimazione dell'integrale di f(x) mediante archi di parabola, ossia polinomi di secondo grado.



ossia $\int_a^b f(x)\ dx \simeq \frac{h}{3}\left((f(a)+4f(m)+f(b)\right)$, dove h è l'ampiezza dell'intervallo (a,b) ed m è il punto medio tra a e b.

Pur essendo più accurata della regola dei trapezi, anche la formula di Cavalieri-Simpson non fornisce approssimazioni particolarmente precise, se non in casi specifici.

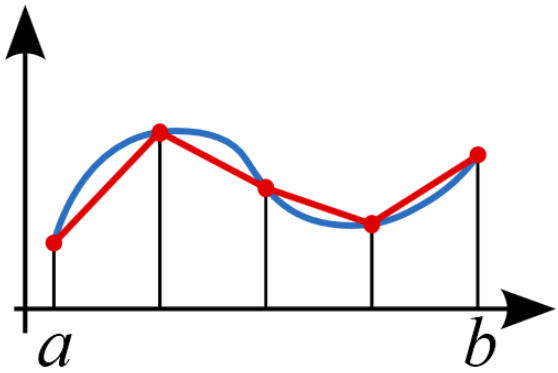
5.2.3 - Formule Composite

Vista l'inaffidabilità delle formule di Newton-Cotes per n alti, si può pensare di ridurre l'impatto dell'errore nell'approssimazione dividendo l'intervallo $[a, b]$ in sottointervalli di uguale lunghezza e calcolando l'integrale nelle singole porzioni utilizzando le formule viste in precedenza.

In generale quindi si ha $\int_a^b f(x)\ dx = \sum_{i=0}^{n-1} \int_{a_i}^{a_{i+1}} f(x)\ dx$

La formula dei Trapezi nella forma composita è dunque:

$$\int_a^b f(x)dx \simeq \frac{h}{2}\left(\sum_{i=0}^{n-1} f(a+ih)+f(a)+f(b)\right)$$



mentre la formula composita di Cavalieri-Simpsons è

$$\int_a^b f(x)dx \simeq \frac{h}{3}\left(f(a)+2\sum_{i=1}^{n-1} f(a+2ih)+4\sum_{i=1}^{n-1} f(a+(2i-1)h)+f(b)\right)$$

5.3 - Formule di Gauss

Le formule di quadratura di Gauss sono più precise delle precedenti, poiché la scelta dei pesi è tale che $\sum_{k=1}^{n}\left(\left|w_k\right|\right)\leq C$, con C costante positiva indipendente da n.

Per illustrare questa scelta è necessario introdurre le proprietà dei polinomi ortogonali.

Proprietà dei polinomi ortogonali

Sia $w:[a,b]\longmapsto R$ una funzione peso, cioè tale che $\forall\ x\in(a,\ b)\ \ w(x)\geq 0$ e $\int_a^b w(x)dx=1$. Una successione di polinomi $p_i(x),i\in N$ è *ortogonale* rispetto a w nell'intervallo $[a,b]$ se:

$$\forall\ \ n,\ m\ \in N\ \ \int_a^b w(x)\,p_n(x)\,p_m(x)dx=\begin{cases}0 & n\neq m\\ c & n=m\end{cases}$$

dove c è una costante reale non nulla. Nel caso particolare in cui $c=1$, diciamo che i polinomi sono *ortonormali* a w .

La particolarità di questi polinomi sta nelle proprietà delle loro radici, che sono tutte reali, distinte e comprese nell'intervallo $[a,b]$.

Le formule di Gauss permettono di approssimare integrali del tipo $\int_a^b w(x)f(x)dx$ scegliendo n radici della famiglia di polinomi ortogonali a $w(x)$ quali nodi $\{x_k\}_{k=1\cdots n}$ e calcolando opportunamente i pesi $(w)_{k=1\cdots n}$.

La quadratura di Gauss, dunque, assume la forma:

$$\int_a^b w(x)\ f(x)dx\ \simeq\ \sum_{k=1}^{n} w_k\ f(x_k)$$

Tale formula è convergente, poiché si può dimostrare che $\sum_{k=1}^{n} w_k = 1$, da cui abbiamo che $\sum_{k=1}^{n} \left(w_k \right) = \sum_{k=1}^{n} w_k = 1$, semplicemente rimuovendo il modulo, essendo i pesi w_k non negativi.

Inoltre, è possibile dimostrare che se $f \in P_{2n-1}$, allora la formula diventa un'uguaglianza, non essendovi errore nella quadratura.

5.4 - Routines Automatiche

Possiamo ora elencare alcune delle più note funzioni peso, con i relativi polinomi ortogonali.

$w(x)$	**Intervallo**	**Polinomi ortogonali**
$\frac{1}{\sqrt{1-x^2}}$	$(-1,\ 1)$	Polinomi di Čebyšëv di I specie
$\sqrt{1-x^2}$	$[-1,\ 1]$	Polinomi di Čebyšëv di II specie
1	$[-1,1]$	Polinomi di Legendre
e^{-x}	$[0,+\infty)$	Polinomi di Laguerre
e^{-x^2}	$(-\infty,+\infty)$	Polinomi di Hermite

Il calcolo dei pesi e dei nodi può essere inoltre automatizzato, grazie a librerie come QUADPACK, dando in input la funzione peso e l'intervallo.

Esempio:
Supponiamo di voler calcolare il seguente integrale

$$\int_{-1}^{1} \frac{x^5}{\sqrt{1-x^2}} dx$$

In questo caso è conveniente considerare come funzione peso $w(x)=\frac{1}{\sqrt{1-x^2}}$, che ci permette di utilizzare i polinomi di Čebyšëv di I specie, per i quali sono noti sia le radici x_k, sia i pesi w_k :

$$\begin{cases} x_k=\cos\left(\frac{\pi}{2n}(2k-1)\right) & k=1\cdots n \\ w_k=\frac{\pi}{n} \end{cases}$$

Poiché $f=x^5\in P_5$, possiamo scegliere $n=3$ per calcolare senza errori e minimizzando i calcoli:

$$\int_{-1}^{1}\frac{x^5}{\sqrt{1-x^2}}dx=\sum_{i=1}^{3}\left(\frac{\pi}{3}\cos\left(\frac{\pi}{6}(2i-1)\right)\right)=\frac{\pi}{3}\left(\cos\frac{\pi}{6}+\cos\frac{\pi}{2}+\cos\frac{5}{6}\pi\right)=0$$

Capitolo 6 - Equazioni non lineari

6.1 - Definizione

Una equazione lineare è una equazione del tipo $ax + b = 0$

Tutte le altre equazioni $f(x) = 0$ sono dette non lineari e la maggior parte di esse devono essere risolte mediante tecniche di tipo numerico.

Esempi di funzioni non lineari $f(x)$ sono:

- polinomi di grado superiore al primo
- funzioni trascendenti
- algoritmi non esprimibili in forma analitica.

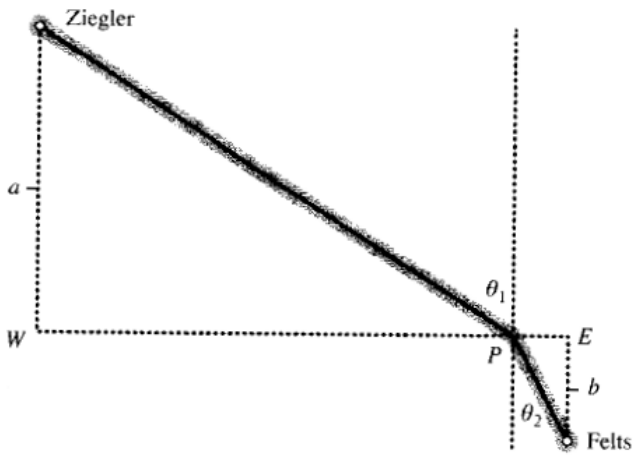
Risolvere un'equazione significa trovare quel valore di $x = \bar{x}$ tale che $f(\bar{x}) = 0$.

Il valore $\bar{x}$ viene detto radice o zero della funzione $f(x)$.

6.2 - Esempi di utilizzo di equazioni non lineari in vari campi

6.2.1 - Ottimizzazione del tracciato di un oleodotto

Si vuole costruire un oleodotto che connetta due città (Ziegler e Felts). Come si possono minimizzare i costi di costruzione, considerando che a nord della linea WE il costo per km è C1 milioni di dollari, mentre a sud della linea è C2 milioni di dollari?



Parametri:
a = 3
b = 1
L = WE = 4
$C_1 = 1$
$C_2 = 2$

Incognita:
WP = x

Soluzione

Assumiamo che il seno dell'angolo di incidenza θ dell'oleodotto sulla linea WE sia inversamente proporzionale al costo unitario, ossia

$$C_1 \sin\theta_1 = C_2 \sin\theta_2$$

Poiché $\sin\theta_1 = \frac{x}{\sqrt{a^2 + x^2}}$ e $\sin\theta_2 = \frac{L - x}{\sqrt{b^2 + (L-x)^2}}$

Allora

$$C_1^2 \frac{x^2}{a^2 + x^2} = C_2^2 \frac{(L-x)^2}{b^2 + (L-x)^2}$$

Riordinando i termini e sostituendo ai parametri i valori numerici, si ottiene

$$3x^4 - 24x^3 + 83x^2 - 288x + 276 = 0$$

Per risolvere il problema bisogna trovare la radice di questa equazione non lineare mediante un procedimento iterativo.

6.2.2 - Programmazione di un sistema robotico a due braccia

Lungo una catena di montaggio di una industria automobilistica sono in azione contemporaneamente due braccia meccaniche, A e B, che modellano il profilo della stessa autovettura. Le due braccia si muovono secondo traiettorie descritte da due polinomi: $f_A(x)$ e $f_B(x)$. Si vuole evitare che le due braccia si urtino e quindi si deve trovare qual è il loro punto di collisione, ossia quando:

$$f_A(x) = f_B(x)$$

Questo problema si riduce nel dover trovare la soluzione dell'equazione non lineare:

$$f_A(x) - f_B(x) = 0$$

6.2.3 - Analisi di dati di microscopia ottica

Per analizzare i risultati di uno studio di microscopia, un biologo deve trovare la soluzione di una equazione del tipo:

$$x^d - 3x + 1 = 0$$

dove *d* è la distanza fra due vetrini. Il problema è facilmente risolvibile quando la distanza è uguale a 2. Tuttavia a causa dell'errore sperimentale questa distanza può essere leggermente inferiore, ad esempio 1,98. In tal caso l'equazione diventa:

$$x^{1,98} - 3x + 1 = 0$$

La soluzione di questa equazione non può essere determinata analiticamente, ma richiede l'utilizzo di procedimenti iterativi.

6.3 - Procedimenti iterativi

Per trovare la radice $\bar{x}$ di un'equazione non lineare *f(x)* si usa il seguente metodo:

- si fissa un valore x_o, detto punto di innesco, che viene scelto in base alle condizioni poste dal problema in esame
- poi si costruisce una successione che sotto opportune ipotesi converge al punto $\bar{x}$ di *f(x)*.

In particolare, data una funzione *g(x)*, detta funzione di iterazione, si costruisce la successione

$$x_{n+1} = g(x_n)$$

cioè

$$x_1 = g(x_o)$$
$$x_2 = g(x_1)$$
$$x_3 = g(x_2)$$
$$\ldots$$

Tale successione si dice costruita mediante un procedimento iterativo.

Si dimostra che, sotto opportune ipotesi, il limite della successione x_n è uguale alla radice $\bar{x}$ dell'equazione $f(x) = 0$:

$$\lim_{n\to\infty} x_n = \bar{x} \qquad f(\bar{x}) = 0$$

6.3.1 - Caratteristiche che deve avere la funzione $f(x)$ per poter applicare procedimenti iterativi

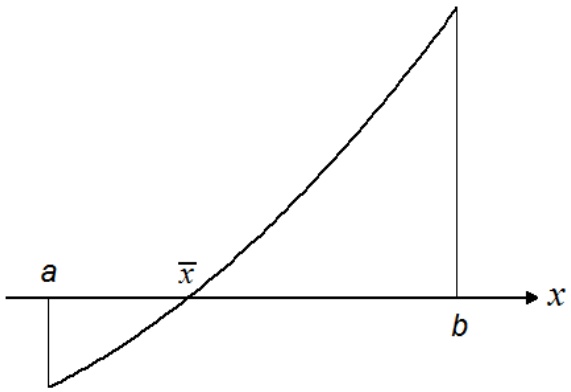
La funzione $f(x)$ deve:

0) essere continua nell'intervallo $[a,b]$

1) assumere segni alterni agli estremi dell'intervallo, ossia $f(a)f(b) < 0$

2) avere un'unica radice compresa nell'intervallo. Una condizione sufficiente (e a volte eccessiva) è che non vi siano punti di flesso nell'intervallo, ossia $f''(x) \neq 0$

Le prime due condizioni assicurano che vi sia almeno una radice nell'intervallo mentre la terza impone che la radice sia unica.



6.3.2 - Metodo delle tangenti o di Newton-Raphson

Nel caso del metodo di Newton-Raphson la funzione $g(x)$ vale:

$$g(x) = x - \frac{f(x)}{f'(x)}$$

Partendo dal punto di innesco x_o viene costruita la successione $x_{n+1} = g(x_n)$ che assume i seguenti valori:

$$x_1 = g(x_o) = x_o - \frac{f(x_o)}{f'(x_o)}$$

$$x_2 = g(x_1) = x_1 - \frac{f(x_1)}{f'(x_1)}$$

....

$$x_{n+1} = g(x_n) = x_n - \frac{f(x_n)}{f'(x_n)}$$

Per applicare questo metodo è necessario aggiungere altre due condizioni sulla $f(x)$:

3) $f'(x) \neq 0$

4) l'intervallo $[a,b]$ deve essere sufficientemente piccolo. Questa condizione deriva dal fatto che il metodo di Newton-Raphson è basato sullo sviluppo di Taylor, che ha un errore tanto più piccolo quanto più si è vicini a x_o.

Gli svantaggi presentati da questi due punti sono bilanciati dal fatto che il metodo è molto rapido (ordine di convergenza 2). Esso quindi è un buon compromesso fra medodi più semplici ma lenti e metodi più rapidi ma complicati.

6.3.3 - Esercizio: trovare il valore numerico di $\sqrt{2}$

Dati: $f(x) = x^2 - 2 = 0$

$x_o = 1$

Trovare il valore numerico della radice $\bar{x}$ positiva.

$$\bar{x} = \sqrt{2}$$

Risoluzione:

Per trovare il valore numerico applichiamo il metodo di Newton-Raphson.

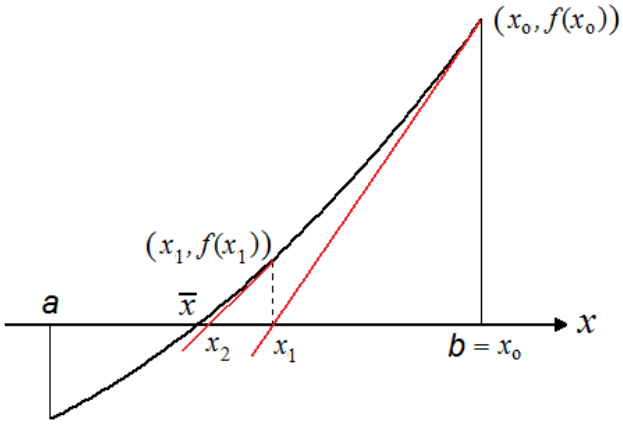
$$f'(x) = 2x$$

$$x_1 = 1 + \frac{1}{2} = \frac{3}{2} = 1{,}5$$

$$x_2 = \frac{3}{2} - \left(\frac{9}{4} - 2\right)\frac{1}{3} = \frac{17}{12} = 1{,}416$$

Come si può vedere con solo due iterazioni si arriva ad un valore sufficientemente accurato.

6.3.4 - Perché si chiama metodo delle tangenti?



Consideriamo $x_o = b$. In base al metodo di Newton-Raphson il valore x_1 è uguale a:

$$x_1 = g(x_o) = x_o - \frac{f(x_o)}{f'(x_o)}$$

Dal punto di vista geometrico, x_1 rappresenta l'intercetta, sull'asse delle *x*, della tangente alla curva $f(x)$ nel punto $(x_o, f(x_o))$.

Alla seconda iterazione si ha:

$$x_2 = g(x_1) = x_1 - \frac{f(x_1)}{f'(x_1)}$$

dove x_2 è l'intercetta, sull'asse delle *x*, della tangente alla curva $f(x)$ nel punto $(x_1, f(x_1))$.

In generale il valore x_{n+1} è l'intercetta, sull'asse delle *x*, della tangente alla curva $f(x)$ nel punto $(x_n, f(x_n))$.

Come si vede dal grafico utilizzando il metodo di Newton-Raphson, ad ogni iterazione ci avviciniamo sempre più rapidamente al valore $\bar{x}$.

6.4 - Velocità di convergenza

La velocità di convergenza α viene definita dal limite:

$$\lim_{n\to\infty} \frac{|\bar{x} - x_{n+1}|}{|\bar{x} - x_n|^{\alpha}} = c \neq 0$$

Il metodo di Newton-Raphson ha una velocità di convergenza quadratica ($\alpha = 2$).

6.4.1 - Test di convergenza o test di stop

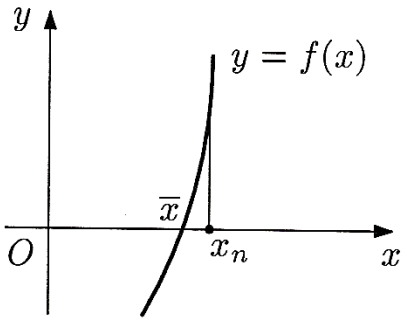
Come si può sapere se il valore x_n, che si è trovato, sia una buona approssimazione del valore $\bar{x}$ oppure se si debba continuare con le iterazioni? Questo ce lo dicono i criteri di convergenza.

Potremmo considerare come criterio di convergenza l'espressione:

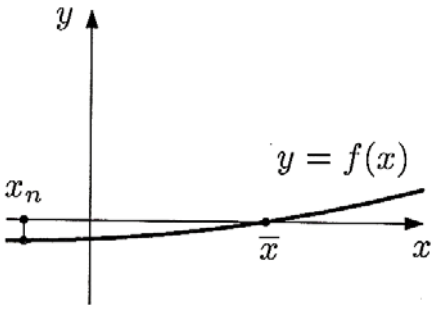
$$|f(x_n)| < \varepsilon$$

Tuttavia le seguenti due situazioni mettono in evidenza i limiti di questo test.

1) In un primo caso, la funzione $f(x)$ potrebbe essere molto ripida. Di conseguenza, sebbene il valore da noi trovato sia molto vicino al valore vero ($x_n \cong \bar{x}$), il test di convergenza fallisce in quanto $|f(x_n)|$ è un valore numerico grande.



2) In un secondo caso, la funzione potrebbe salire molto lentamente e rimanere per un lungo tratto prossima ma diversa da zero. Pertanto, sebbene il valore trovato sia molto distante da quello vero ($x_n \neq \bar{x}$), il test di convergenza ci indica erroneamente che abbiamo trovato la radice della funzione in quanto $|f(x_n)|$ è molto piccolo.



Un valido criterio di stop è l'applicazione di queste due espressioni:

$$|x_{n+1} - x_n| < \varepsilon$$

$$|x_{n+1} - x_n| < |x_n|\varepsilon$$

Capitolo 7 - Sistemi lineari

7.1 - Metodi per la risoluzione dei sistemi lineari

Dato un sistema lineare:

$$A \cdot x = b$$

con:

- A matrice dei coefficienti di dimensione $n \cdot n$;
- b vettore dei termini noti di n componenti;
- x vettore delle incognite di n componenti;

I metodi numerici per la risoluzione del sistema lineare vengono suddivisi in due classi:

I. **Metodi diretti**

II. **Metodi Iterativi**

Nel primo caso l'esatta soluzione viene costruita in assenza di errori di arrotondamento in un numero finito di passi (vengono usati in sistemi con matrici A dense cioè con una piccola quantità di zeri). Questi metodi alterano la matrice A.

Nel secondo caso ci fornisce come soluzione una successione di vettori che convergono alla soluzione esatta (vengono usati in sistemi con matrici A sparse cioè con una elevata quantità di elementi uguali a zero e di ordine elevato). Questi metodi non alterano la matrice A.

Sistemi sparsi sono presenti in numerose applicazioni. A causa dell'elevato ordine delle matrici coinvolte in problemi di questo tipo, i metodi diretti non sono sempre utilizzabili. Infatti, questi ultimi ottengono la soluzione x in un numero finito di passi mediante una successione (finita) di trasformazioni del problema iniziale in problemi equivalenti, cioè con la stessa soluzione x, ma con matrici dei coefficienti diverse; anzi, con il procedere del metodo di risoluzione, il numero di elementi non nulli presenti in queste matrici generalmente cresce, e può ben presto saturare lo spazio disponibile nella memoria centrale del calcolatore.

In questi casi è utile, e spesso indispensabile, utilizzare metodi iterativi, i quali, operando sempre e solo con gli elementi della matrice iniziale A, generano una successione infinita di vettori convergente, sotto opportune condizioni, alla soluzione cercata. Poiché il processo iterativo lascia inalterata la matrice A, è sufficiente memorizzare gli elementi non nulli di A.

7.2 - Metodi iterativi

Questi metodi a differenza di quelli diretti non alterano mai la matrice iniziale A; ma partendo da un'approssimazione iniziale $x^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)}\right)$ essi definiscono una successione di approssimazioni $x^1, x^2, \dots, x^n$, convergente, sotto opportune ipotesi, alla soluzione x del sistema non singolare $A \cdot x = b$.

7.2.1 - Metodo di Jacobi

Dato il sistema :

$$A \cdot x = b;$$

Scriviamo il sistema in modo esplicito:

Equazioni (1);

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + a_{13} \cdot x_3 + \cdots + a_{1n} \cdot x_n = b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + a_{13} \cdot x_3 + \cdots + a_{2n} \cdot x_n = b_2 \\ a_{31} \cdot x_1 + a_{32} \cdot x_2 + a_{33} \cdot x_3 + \cdots + a_{3n} \cdot x_n = b_3 \\ \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots = \cdots \\ a_{n1} \cdot x_1 + a_{n2} \cdot x_2 + a_{n3} \cdot x_3 + \cdots + a_{nn} \cdot x_n = b_n \end{cases}$$

$$\Downarrow\Downarrow$$

Equazioni (2);

$$\begin{cases} x_1 = \frac{b_1 - \sum_{j \neq 1} a_{1j} x_j}{a_{11}} \\ x_2 = \frac{b_2 - \sum_{j \neq 2} a_{2j} x_j}{a_{22}} \\ x_3 = \frac{b_3 - \sum_{j \neq 3} a_{3j} x_j}{a_{33}} \\ \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\ x_n = \frac{b_n - \sum_{j \neq n} a_{nj} x_j}{a_{nn}} \end{cases}$$

A questo punto applichiamo il metodo di Jacobi, il quale a partire da un vettore iniziale : $x^{(0)} = \left(x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)}\right)$ (approssimazione grossolana), calcola le approssimazioni successive:

$$x^{(k+1)} = \left(x_1^{(k+1)}, x_2^{(k+1)}, \dots, x_n^{(k+1)}\right); \quad \text{con } k = 1, 2, \dots$$

Vediamo in dettaglio come costruire queste approssimazioni.

Dall'equazione (2), sostituendo al posto di $x_1, x_2, x_3, \cdots, x_n$ i valori $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \cdots, x_n^{(0)})$

otteniamo:

$$\begin{cases} x_1^{(1)} = \frac{b_1 - \sum_{j\neq 1} a_{1j} x_j^{(0)}}{a_{11}} \\ x_2^{(1)} = \frac{b_2 - \sum_{j\neq 2} a_{2j} x_j^{(0)}}{a_{22}} \\ x_3^{(1)} = \frac{b_3 - \sum_{j\neq 3} a_{3j} x_j^{(0)}}{a_{33}} \\ \dots\dots\dots\dots\dots\dots\dots \\ \dots\dots\dots\dots\dots\dots\dots \\ x_n^{(1)} = \frac{b_n - \sum_{j\neq n} a_{nj} x_j^{(0)}}{a_{nn}} \end{cases}$$

ottenendo un primo raffinamento del vettore:

$$x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \cdots, x_n^{(1)})$$

Ottenuto il vettore approssimato $x^{(1)}$ andrò a ripetere la sostituzione ricavando un secondo raffinamento del vettore:

$$\begin{cases} x_1^{(2)} = \frac{b_1 - \sum_{j\neq 1} a_{1j} x_j^{(1)}}{a_{11}} \\ x_2^{(2)} = \frac{b_2 - \sum_{j\neq 2} a_{2j} x_j^{(1)}}{a_{22}} \\ x_3^{(2)} = \frac{b_3 - \sum_{j\neq 3} a_{3j} x_j^{(1)}}{a_{33}} \\ \dots\dots\dots\dots\dots\dots\dots \\ x_n^{(2)} = \frac{b_n - \sum_{j\neq n} a_{nj} x_j^{(1)}}{a_{nn}} \end{cases}$$

Iterando questa operazione otterrò sempre un approssimazione migliore del vettore delle incognite (cioè più vicina alla soluzione esatta).

Generalizzando la formula otteniamo :

$$x_i^{(k+1)} = \frac{b_i - \sum_{j\neq i} a_{ij} x_j^{(k)}}{a_{ii}}$$

N.B. Questo metodo converge solo per alcuni tipi di matrice.

Ad esempio per matrici a diagonale dominante per righe o per colonne:

$| a_{ii} | > \sum_j | a_{ij} |$ per righe (i bloccato);

$| a_{jj} | > \sum_i | a_{ij} |$ per colonne (j bloccato).

7.2.2 - Metodo di Gauss-Seidel

Ora vediamo come si può migliorare il Metodo di Jacobi.

Nel metodo di Jacobi ogni singola componente $x^{(k+1)}$ viene calcolata solo tramite il valore precedente $x^{(k)}$:

$$x_i^{(k+1)} = \frac{b_i - \sum_{j \neq i} a_{ij} x_j^{(k)}}{a_{ii}}$$

Si può notare che quando si va a calcolare ad esempio $x_2^{(1)}$ inserirò il valore $x_1^{(0)}$, in realtà ho calcolato il valore $x_1^{(1)}$ che è "migliore" del precedente; pertanto posso scrivere:

$$x_2^{(1)} = \frac{b_2 - a_{21} x_1^{(1)} - \sum_{j \neq 2,1} a_{2j} x_j^{(0)}}{a_{22}}$$

(utilizzo subito il valore più accurato che ho trovato)

Questo vale anche per i successivi valori con $j > 1$.

Allora più genericamente si potrà dire:

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij} \cdot x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} \cdot x_j^{(k)}}{a_{ii}}$$

N.B. Questo metodo converge solo per alcuni tipi di matrice.

Ad esempio per matrici a diagonale dominante per righe o per colonne:

$| a_{ii} | > \sum_j | a_{ij} |$ per righe (i bloccato);

$| a_{jj} | > \sum_i | a_{ij} |$ per colonne(j bloccato).

7.2.3 - Un metodo di rilassamento

Il metodo che stiamo per presentare è detto metodo di sovrarilassamento (SOR). Riprendiamo il metodo di Gauss-Seidel, che ricordiamo essere un metodo iterativo. Dalla relazione ivi presentata, sottraendo $x_i^{(k)}$ da entrambi i membri, otteniamo

$$r_i^{(k)} = x_i^{(k+1)} - x_i^{(k)} = \frac{1}{a_{ii}} \cdot \left[b_i - \sum_{j=1}^{i-1} a_{ij} \cdot x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} \cdot x_j^{(k)} \right], \quad i = 1, \dots, n$$

Dove $r_i^{(k)}$ rappresenta la correzione da apportare a $x_i^{(k)}$ per ottenere la nuova approssimazione

$$x_i^{(k+1)} = x_i^{(k)} + r_i^{(k)}, \quad k = 0,1,2, \dots$$

Le due relazioni appena presentate definiscono ancora un metodo di Gauss-Seidel; tuttavia la forma della seconda relazione ci suggerisce l'introduzione di un parametro ω

$$x_i^{(k+1)} = x_i^{(k)} + \omega \cdot r_i^{(k)}, \qquad 1 < \omega < 2$$

Scegliere il parametro ω non è semplice in quanto la scelta dovrà essere finalizzata all'ottenimento di una convergenza più rapida possibile della successione $x^{(k)}$. In questo caso ($1 < \omega < 2$) il valore ottenuto dall'iterazione attuale viene incrementato in valore assoluto: si sta quindi assumendo che la soluzione stia convergendo troppo lentamente, e si parla in questo caso di sovrarilassamento. Possiamo osservare che per $\omega = 1$ si ottiene proprio il metodo di Gauss-Seidel. Infatti per $\omega = 1$ si ha

$$x_i^{(k+1)} = x_i^{(k)} + 1 \cdot r_i^{(k)}$$

$$x_i^{(k+1)} = x_i^{(k)} + r_i^{(k)}$$

ma dato che

$$r_i^{(k)} = x_i^{(k+1)} - x_i^{(k)} = \frac{1}{a_{ii}} \cdot \left[b_i - \sum_{j=1}^{i-1} a_{ij} \cdot x_j^{(k+1)} - \sum_{j=i}^{n} a_{ij} \cdot x_j^{(k)} \right], \quad i = 1, \dots, n$$

si ottiene quindi

$$x_i^{(k+1)} = r_i^{(k)} - x_i^{(k)} = \frac{1}{a_{ii}} \cdot \left[b_i - \sum_{j=1}^{i-1} a_{ij} \cdot x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} \cdot x_j^{(k)} \right]$$

cioè la formula di Gauss-Seidel.

7.3 - Metodi diretti

L'idea centrale dei metodi diretti è l'eliminazione, che come è noto consiste nel ricavare (eliminare) da una fissata equazione una particolare incognita e nella sua sostituzione nelle equazioni rimanenti. La sostituzione diminuisce il problema: $n \to n-1$. Iterando il procedimento, si riduce il problema originario ad un problema in una sola incognita. Determinata tale incognita, le altre componenti della soluzione sono successivamente ottenute mediante una procedura di sostituzione all'indietro.

7.3.1 - Il metodo di eliminazione di Gauss

Il metodo diretto più noto e più ed utilizzato è senza dubbio quello delle eliminazioni successive di Gauss.

Osserviamo preliminarmente che la soluzione di un sistema non singolare di forma triangolare, superiore o inferiore, è pressoché immediata. Infatti nel caso di un sistema triangolare superiore

$$\begin{cases} a_{11}\cdot x_1 + a_{12}\cdot x_2 + a_{13}\cdot x_3 + \cdots + a_{1n}\cdot x_n = b_1 \\ a_{22}\cdot x_2 + a_{23}\cdot x_3 + \cdots + a_{2n}\cdot x_n = b_2 \\ a_{33}\cdot x_3 + \cdots + a_{3n}\cdot x_n = b_3 \\ \cdots + \cdots = \cdots \\ a_{nn}\cdot x_n = b_n \end{cases}$$

con gli elementi della diagonale $a_{ii} \neq 0$, con $i = 1, 2, \dots, n$, abbiamo

$$\begin{cases} x_n = \dfrac{b_n}{a_{nn}} \\ x_k = \dfrac{b_k - \sum_{j=k+1}^{n} a_{kj}\cdot x_j}{a_{kk}}; \quad k = n-1, \dots, 1 \end{cases}$$

In questo caso per determinare il valore delle incognite basta lavorare a ritroso partendo dall'ultima equazione $a_{nn}\cdot x_n = b_n$ e quindi con dei semplici calcoli ottenere tutte le altre.

Il costo computazionale per la determinazione dei valori delle incognite in questo caso è basso, risulta infatti dell'ordine di $O\left(\frac{n^2}{2}\right)$, il che è abbastanza vantaggioso visto che con i sistemi pieni il costo è normalmente di $O(n^3)$.

Questa considerazione ci suggerisce pertanto di esaminare la possibilità di trasformare un generico sistema non singolare in un sistema equivalente, cioè con la stessa soluzione, di forma triangolare. Cerco quindi metodi che mi consentano di effettuare questa trasformazione.

È noto che, quando a un'equazione del sistema sostituiamo una combinazione lineare dell'equazione con un'altra dello stesso sistema, il nuovo sistema risulta equivalente al precedente.

Con il metodo di Gauss è sempre possibile trasformare, mediante un numero finito di combinazioni lineari ed eventualmente permutazioni di equazioni, un generico sistema non singolare in uno equivalente di forma triangolare.

Permutare significa scambiare tra loro le righe (o le colonne). Se cambia l'ordine delle equazioni (o delle incognite) non cambia la soluzione del sistema. Talvolta nell'applicare il metodo vengono eseguite delle permutazioni cambiando di posto le equazioni (o le incognite).

Per meglio illustrare il processo delle eliminazioni di Gauss scriviamo il sistema $A \cdot x = b$ esplicitamente:

$$\begin{cases} a_{11} \cdot x_1 + a_{12} \cdot x_2 + a_{13} \cdot x_3 + \cdots + a_{1n} \cdot x_n = b_1 \\ a_{21} \cdot x_1 + a_{22} \cdot x_2 + a_{13} \cdot x_3 + \cdots + a_{2n} \cdot x_n = b_2 \\ a_{31} \cdot x_1 + a_{32} \cdot x_2 + a_{33} \cdot x_3 + \cdots + a_{3n} \cdot x_n = b_3 \\ \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots = \cdots \\ a_{n1} \cdot x_1 + a_{n2} \cdot x_2 + a_{n3} \cdot x_3 + \cdots + a_{nn} \cdot x_n = b_n \end{cases}$$

Per semplicità introduciamo l'ipotesi che i termini sulla diagonali siano diversi da zero ($a_{ii} \neq 0$).

Se $a_{11} = 0$ è infatti sufficiente permutare le equazioni, ad esempio, scambiando la prima con la seconda. Se anche il nuovo coefficiente è uguale a zero passo alla successiva e così via fino a che uno di questi coefficienti è diverso da zero. Uno lo sarà senz'altro in quanto affinché il sistema sia non degenere ogni incognita deve avere almeno un coefficiente diverso da zero.

Possiamo eliminare l'incognita x_1 dalle ultime $(n-1)$ equazioni del sistema, cioè dalla 2ª, 3ª, n-esima, sommando alla i-esima equazione, $i = 2, 3, \ldots, n$, la prima moltiplicata per

$$m_{i1} = -\frac{a_{i1}}{a_{11}}, \qquad i = 2, 3, \ldots, n$$

Riscriviamo la prima equazione moltiplicando ambo i membri per il coefficiente $m_{21}=\left(-\frac{a_{21}}{a_{11}}\right)$

$$\left(-\frac{a_{21}}{a_{11}}a_{11}\right) \cdot x_1 + \left(-\frac{a_{21}}{a_{11}}a_{12}\right) \cdot x_2 + \left(-\frac{a_{21}}{a_{11}}a_{13}\right) \cdot x_3 + \cdots + \left(-\frac{a_{21}}{a_{11}}a_{1n}\right) \cdot x_n = \left(-\frac{a_{21}}{a_{11}}\right) \cdot b_1$$

Aggiungiamo ora a questa quantità la seconda equazione del sistema

$$a_{21} \cdot x_1 + a_{22} \cdot x_2 + a_{23} \cdot x_3 + \cdots + a_{2n} \cdot x_n = b_2$$

Eseguiamo quindi una combinazione lineare di queste due equazioni sommandole.

$$\left(a_{21}-\frac{a_{21}}{a_{11}}a_{11}\right)x_1+\left(a_{22}-\frac{a_{21}}{a_{11}}a_{12}\right)x_2+\left(a_{23}-\frac{a_{21}}{a_{11}}a_{13}\right)x_3+\cdots+\left(a_{2n}-\frac{a_{21}}{a_{11}}a_{1n}\right)x_n=b_2-\frac{a_{21}}{a_{11}}b_1$$

Riscriviamo ora il risultato come:

$$0\cdot x_1+a_{22}^{(2)}\cdot x_2+a_{23}^{(2)}\cdot x_3+\cdots+a_{2n}^{(2)}\cdot x_n=b_2^{(2)}$$

L'indice $^{(2)}$ indica che si tratta della seconda trasformazione del sistema ottenuta lavorando con le combinazioni lineari.

Abbiamo preso la prima riga, moltiplicata per un coefficiente e sommata alla seconda, ottenendo una nuova equazione, dalla quale cui il coefficiente di x_1 è scomparso, che sostituiamo alla seconda equazione. In modo analogo lavoriamo sulla terza equazione e così via per tutte le altre.

Dopo queste operazioni, il nuovo sistema, equivalente al precedente, assume la forma

$$\begin{cases} a_{11}\cdot x_1 + a_{12}\cdot x_2 + a_{13}\cdot x_3 +\cdots+ a_{1n}\cdot x_n = b_1 \\ a_{22}^{(2)}\cdot x_2 + a_{23}^{(2)}\cdot x_3 +\cdots+ a_{2n}^{(2)}\cdot x_n = b_2^{(2)} \\ a_{32}^{(2)}\cdot x_2 + a_{33}^{(2)}\cdot x_3 +\cdots+ a_{3n}^{(2)}\cdot x_n = b_3^{(2)} \\ \cdots + \cdots +\cdots+ \cdots = \cdots \\ a_{n2}^{(2)}\cdot x_2 + a_{n3}^{(2)}\cdot x_3 +\cdots+ a_{nn}^{(2)}\cdot x_n = b_n^{(2)} \end{cases}$$

dove

$$i=2,\dots,n:\begin{cases} a_{ij}^{(2)}=a_{ij}+m_{i1}\cdot a_{1j}, & j=2,\dots,n \\ b_i^{(2)}=b_i+m_{i1}\cdot b_1 \end{cases}$$

Riapplichiamo il procedimento (di eliminazione) alle ultime (n-1) equazioni. Se $a_{22}^{(2)}\neq 0$ possiamo eliminare l'incognita x_2 dalla 3ª, 4ª, ..., n-esima equazione; è sufficiente porre

$$m_{i2}=-\frac{a_{i2}^{(2)}}{a_{22}^{(2)}},\qquad i=3,\dots,n$$

e sommare alla i-esima equazione, $i=3,\dots,n$, la 2ª moltiplicata per m_{i2}. Avremo un nuovo sistema equivalente a quello di partenza:

$$\begin{cases} a_{11}\cdot x_1 + a_{12}\cdot x_2 + a_{13}\cdot x_3 +\cdots+ a_{1n}\cdot x_n = b_1 \\ a_{22}^{(2)}\cdot x_2 + a_{23}^{(2)}\cdot x_3 +\cdots+ a_{2n}^{(2)}\cdot x_n = b_2^{(2)} \\ a_{33}^{(3)}\cdot x_3 +\cdots+ a_{3n}^{(3)}\cdot x_n = b_3^{(3)} \\ \cdots +\cdots+ \cdots = \cdots \\ a_{n3}^{(3)}\cdot x_3 +\cdots+ a_{nn}^{(3)}\cdot x_n = b_n^{(3)} \end{cases}$$

dove

$$i = 3, \dots, n : \begin{cases} a_{ij}^{(3)} = a_{ij}^{(2)} + m_{i2} \cdot a_{2j}^{(2)}, & j = 3, \dots, n \\ b_i^{(3)} = b_i^{(2)} + m_{i2} \cdot b_2^{(2)} \end{cases}$$

Gli elementi a_{11}, $a_{22}^{(2)}$, $a_{33}^{(3)}$, ..., che compaiono durante le successive eliminazioni vengono chiamati elementi pivot.

Dopo $(n-1)$ passi arriveremo, supponendo tutti gli elementi pivot non nulli, al seguente sistema triangolare

$$\begin{cases} a_{11}^{(1)} \cdot x_1 + a_{12}^{(1)} \cdot x_2 + a_{13}^{(1)} \cdot x_3 + \dots + a_{1n}^{(1)} \cdot x_n = b_1^{(1)} \\ a_{22}^{(2)} \cdot x_2 + a_{23}^{(2)} \cdot x_3 + \dots + a_{2n}^{(2)} \cdot x_n = b_2^{(2)} \\ a_{33}^{(3)} \cdot x_3 + \dots + a_{3n}^{(3)} \cdot x_n = b_3^{(3)} \\ \dots + \dots = \dots \\ a_{nn}^{(n)} \cdot x_n = b_n^{(n)} \end{cases}$$

dove per convenienza di notazione abbiamo posto $a_{1j}^{(1)} = a_{1j}$, $i = 1, \dots, n$, e $b_1^{(1)} = b_1$, la cui soluzione è, come rilevato all'inizio, pressoché immediata. Osserviamo che il termine noto b viene trasformato esattamente come se fosse un'ulteriore colonna di A

Lo schema di calcolo seguente riassume la descrizione del metodo Gauss:

l'eliminazione delle variabili viene eseguita in $(n-1)$ passi;

al k-esimo passo con $k = 1, 2, \dots, n-1$

gli elementi $a_{ij}^{(k)}$, i e $j > k$, e $b_i^{(k)}$con vengono trasformati in accordo con le formule

$$i = k+1, \dots, n : \begin{cases} m_{ik} = -a_{ik}^{(k)} / a_{kk}^{(k)} \\ a_{ij}^{(k+1)} = a_{ij}^{(k)} + m_{ik} \cdot a_{kj}^{(k)}, & j = k+1, \dots, n \\ b_i^{(k+1)} = b_i^{(k)} + m_{ik} \cdot b_k^{(k)} \end{cases}$$

la soluzione del sistema triangolare finale risulta

$$\begin{cases} x_n = \dfrac{b_n^{(n)}}{a_{nn}^{(n)}} \\ x_k = \left(b_k^{(k)} - \displaystyle\sum_{j=k+1}^{n} a_{kj}^{(k)} \cdot x_j \right) \Big/ a_{kk}^{(k)}, \quad k = n-1, \dots, 1 \end{cases}$$

7.3.2 - Esempio di applicazione del metodo di eliminazione di Gauss

Consideriamo il seguente sistema di equazioni lineari di quattro equazioni in quattro incognite:

$$\begin{cases} 2 \cdot x_1 - x_2 + x_3 - 2 \cdot x_4 = 0 \\ 2 \cdot x_2 - x_4 = 1 \\ x_1 - 2 \cdot x_3 + x_4 = 0 \\ 2 \cdot x_2 + x_3 + x_4 = 4 \end{cases}$$

Il nostro scopo è ricondurlo a un sistema triangolare superiore, iniziamo nel primo passaggio con rendere nulli i coefficienti dell'incognita x_1 in tutte le equazioni successive alla prima. L'unica equazione che ha il coefficiente non nullo è la terza. Dobbiamo quindi sostituire questa equazione con una sua combinazione lineare con la prima equazione in modo da rendere nullo il coefficiente $a_{31}^{(2)}$.

Applichiamo l'algoritmo di Gauss con $k = 1$ (primo passaggio), $i = 3$ (terza riga) e $j = 1..4$ (tutte le colonne)

$$\begin{cases} m_{31} = -a_{31}^{(1)}/a_{11}^{(1)} \\ a_{31}^{(1+1)} = a_{31}^{(1)} + m_{31} \cdot a_{11}^{(1)} \\ a_{32}^{(1+1)} = a_{32}^{(1)} + m_{31} \cdot a_{12}^{(1)} \\ a_{33}^{(1+1)} = a_{33}^{(1)} + m_{31} \cdot a_{13}^{(1)} \\ a_{34}^{(1+1)} = a_{34}^{(1)} + m_{31} \cdot a_{14}^{(1)} \\ b_3^{(1+1)} = b_3^{(1)} + m_{31} \cdot b_1^{(1)} \end{cases} \Rightarrow \begin{cases} m_{31} = -1/2 \\ a_{31}^{(2)} = 1 + (-1/2) \cdot 2 \\ a_{32}^{(2)} = 0 + (-1/2) \cdot (-1) \\ a_{33}^{(2)} = (-2) + (-1/2) \cdot 1 \\ a_{34}^{(2)} = 1 + (-1/2) \cdot (-2) \\ b_3^{(2)} = 0 + (-1/2) \cdot 0 \end{cases} \Rightarrow \begin{cases} m_{31} = -1/2 \\ a_{31}^{(2)} = 0 \\ a_{32}^{(2)} = 1/2 \\ a_{33}^{(2)} = -5/2 \\ a_{34}^{(2)} = 2 \\ b_3^{(2)} = 0 \end{cases}$$

Effettuando la sostituzione della terza equazione con quella ottenuta applicando i coefficienti sopra determinati otteniamo il nuovo sistema, al termine del primo passaggio, nel quale sono nulli tutti i coefficienti dell'incognita x_1 a partire dalla seconda equazione.

$$\begin{cases} 2 \cdot x_1 - x_2 + x_3 - 2 \cdot x_4 = 0 \\ 2 \cdot x_2 - x_4 = 1 \\ \frac{1}{2} \cdot x_2 - \frac{5}{2} \cdot x_3 + 2 \cdot x_4 = 0 \\ 2 \cdot x_2 + x_3 + x_4 = 4 \end{cases}$$

L'algoritmo contempla anche il calcolo dei coefficienti di tutte le nuove equazioni successive alla k-esima, dove k indica il passo computazionale, e quindi anche dalla seconda e quarta equazione. È immediato però notare che queste restano invariate essendo nullo il coefficiente della k-esima incognita e quindi anche il coefficiente m_{ik}.

Dobbiamo ora rendere nulli i coefficienti dell'incognita x_2 nella terza e quarta equazione.

Per ottenere la nuova terza equazione applichiamo l'algoritmo di Gauss con $k = 2$ (secondo passaggio), $i = 3$ (terza riga) e $j = 2..4$ (ultime tre colonne)

$$\begin{cases} m_{32} = -a_{32}^{(2)}/a_{22}^{(2)} \\ a_{32}^{(2+1)} = a_{32}^{(2)} + m_{32} \cdot a_{22}^{(2)} \\ a_{33}^{(2+1)} = a_{33}^{(2)} + m_{32} \cdot a_{23}^{(2)} \\ a_{34}^{(2+1)} = a_{34}^{(2)} + m_{32} \cdot a_{24}^{(2)} \\ b_3^{(2+1)} = b_3^{(2)} + m_{32} \cdot b_2^{(2)} \end{cases} \Rightarrow \begin{cases} m_{32} = -(1/2)/2 \\ a_{32}^{(3)} = (1/2) + (-1/4) \cdot 2 \\ a_{33}^{(3)} = (-5/2) + (-1/4) \cdot 0 \\ a_{34}^{(3)} = 2 + (-1/4) \cdot (-1) \\ b_3^{(3)} = 0 + (-1/4) \cdot 1 \end{cases} \Rightarrow \begin{cases} m_{32} = -1/4 \\ a_{32}^{(3)} = 0 \\ a_{33}^{(3)} = -5/2 \\ a_{34}^{(3)} = 2 \\ b_3^{(3)} = -1/4 \end{cases}$$

Quindi per ottenere la nuova quarta equazione applichiamo l'algoritmo di Gauss con $k = 2$ (secondo passaggio), $i = 4$ (quarta riga) e $j = 2..4$ (ultime tre colonne)

$$\begin{cases} m_{42} = -a_{42}^{(2)}/a_{22}^{(2)} \\ a_{42}^{(2+1)} = a_{42}^{(2)} + m_{42} \cdot a_{22}^{(2)} \\ a_{43}^{(2+1)} = a_{43}^{(2)} + m_{42} \cdot a_{23}^{(2)} \\ a_{44}^{(2+1)} = a_{44}^{(2)} + m_{42} \cdot a_{24}^{(2)} \\ b_4^{(2+1)} = b_4^{(2)} + m_{42} \cdot b_2^{(2)} \end{cases} \Rightarrow \begin{cases} m_{42} = -2/2 \\ a_{42}^{(3)} = 2 + (-1) \cdot 2 \\ a_{43}^{(3)} = 1 + (-1) \cdot 0 \\ a_{44}^{(3)} = 1 + (-1) \cdot (-1) \\ b_4^{(3)} = 4 + (-1) \cdot 1 \end{cases} \Rightarrow \begin{cases} m_{42} = -1 \\ a_{42}^{(3)} = 0 \\ a_{43}^{(3)} = 1 \\ a_{44}^{(3)} = 2 \\ b_4^{(3)} = 3 \end{cases}$$

Sostituendo ora le nuove terza e quarta equazione con quelle ottenuta applicando i coefficienti sopra determinati otteniamo il nuovo sistema, al termine del secondo passaggio, nel quale sono nulli tutti i coefficienti dell'incognita x_2 a partire dalla terza equazione.

$$\begin{cases} 2 \cdot x_1 - x_2 + x_3 - 2 \cdot x_4 = 0 \\ 2 \cdot x_2 - x_4 = 1 \\ -\frac{5}{2} \cdot x_3 + \frac{9}{4} \cdot x_4 = -\frac{1}{4} \\ + x_3 + 2 \cdot x_4 = 3 \end{cases}$$

Dobbiamo infine rendere nulli i coefficienti dell'incognita x_3 nella quarta equazione.

Per ottenere la nuova quarta equazione applichiamo l'algoritmo di Gauss con $k = 3$ (sterzo passaggio), $i = 3$ (quarta riga) e $j = 3..4$ (ultime due colonne)

$$\begin{cases} m_{43} = -a_{43}^{(3)}/a_{33}^{(3)} \\ a_{43}^{(3+1)} = a_{43}^{(3)} + m_{43} \cdot a_{33}^{(3)} \\ a_{44}^{(3+1)} = a_{44}^{(3)} + m_{43} \cdot a_{34}^{(3)} \\ b_4^{(3+1)} = b_4^{(3)} + m_{43} \cdot b_3^{(3)} \end{cases} \Rightarrow \begin{cases} m_{43} = -1/(-5/2) \\ a_{43}^{(4)} = 1 + (2/5) \cdot (-5/2) \\ a_{44}^{(4)} = 2 + (2/5) \cdot (9/4) \\ b_4^{(4)} = 3 + (2/5) \cdot (-1/4) \end{cases} \Rightarrow \begin{cases} m_{43} = 2/5 \\ a_{43}^{(4)} = 0 \\ a_{44}^{(4)} = 29/10 \\ b_4^{(4)} = 29/10 \end{cases}$$

Sostituendo ora le nuova quarta equazione con quella ottenuta applicando i coefficienti sopra determinati otteniamo il nuovo sistema, al termine del terzo ed ultimo passaggio, nel quale sono nulli tutti i coefficienti delle incognite che stanno al di sotto della diagonale principale.

$$\begin{cases} 2 \cdot x_1 - x_2 + x_3 - 2 \cdot x_4 = 0 \\ 2 \cdot x_2 - x_4 = 1 \\ -\frac{5}{2} \cdot x_3 + \frac{9}{4} \cdot x_4 = -\frac{1}{4} \\ + \frac{29}{10} \cdot x_4 = \frac{29}{10} \end{cases}$$

Possiamo ora procedere alla risoluzione del sistema triangolare, equivalente al sistema iniziale, ottenuto applicando l'algoritmo di riduzione di Gauss.

Partendo dall'ultima equazione otteniamo in successione i valori di tutte le incognite.

$$\begin{cases} x_4 = b_4^{(4)} / a_{44}^{(4)} \\ x_3 = \left(b_3^{(3)} - a_{34}^{(3)} \cdot x_4\right) / a_{33}^{(3)} \\ x_2 = \left(b_2^{(2)} - \left(a_{23}^{(2)} \cdot x_3 + a_{24}^{(2)} \cdot x_4\right)\right) / a_{22}^{(2)} \\ x_1 = \left(b_1^{(1)} - \left(a_{12}^{(1)} \cdot x_2 + a_{13}^{(1)} \cdot x_3 + a_{14}^{(1)} \cdot x_4\right)\right) / a_{11}^{(1)} \end{cases}$$

Sostituiamo nelle formule sopra riportate, ottenute a partire dall'algoritmo risolutivo, i valori calcolati:

$$\begin{cases} x_4 = \frac{29}{10} \cdot \frac{10}{29} = 1 \\ x_3 = \left(-\frac{1}{4} - \frac{9}{4} \cdot 1\right) \Big/ \left(-\frac{5}{2}\right) = 1 \\ x_2 = \left(1 - (0 \cdot 1 + (-1) \cdot 1)\right) / 2 = 1 \\ x_1 = \left(0 - \left((-1) \cdot 1 + 1 \cdot 1 + (-2) \cdot 1\right)\right) / 2 = 1 \end{cases}$$

Abbiamo quindi risulto il sistema iniziale per questo semplice esempio applicando passo a passo l'algoritmo di riduzione di Gauss.

7.3.3 - Pivoting e scaling

Se nello svolgimento del processo delle eliminazioni di Gauss, al passo k-esimo troviamo $a_{kk}^{(k)} = 0$, allora il metodo così come descritto non può proseguire. Supposto il sistema non singolare, il rimedio consiste nel scambiare di posto due equazioni (la k-esima con una delle successive) in modo che il nuovo $a_{kk}^{(k)}$ sia diverso da zero. Infatti, se $a_{kk}^{(k)} = 0$ necessariamente qualche altro elemento $a_{ik}^{(k)}$, $i = k+1, \dots, n$, della colonna k-esima della matrice dei coefficienti deve essere non nullo, altrimenti il sistema risulta singolare.

Supponiamo per esempio che $a_{rk}^{(k)} \neq 0$; in questo caso basta scambiare l'equazione k-esima con la r-esima e poi procedere con le eliminazioni. Ne segue che ogni sistema non singolare può sempre essere ricondotto alla forma triangolare (superiore) con il metodo di Gauss (più eventuali scambi di equazioni).

Per assicurare una migliore stabilità numerica al metodo è spesso necessario permutare l'ordine delle equazioni anche quando l'elemento pivot non è esattamente zero, ma è molto piccolo (in valore assoluto) rispetto agli altri elementi. Infatti, in quest'ultimo caso l'elemento pivot potrebbe essere stato generato come differenza di due numeri quasi uguali e quindi essere stato contaminato dal fenomeno della cancellazione numerica.

Per cercare di evitare catastrofiche propagazioni di errori è di solito necessario scegliere, al generico passo k-esimo, l'elemento pivot scegliendo una delle due seguenti strategie:

Pivoting parziale: scegliere r uguale al più piccolo intero $\geq k$ tale che

$$a_{rk}^{(k)} = \max_{k \leq i \leq n} \left| a_{ik}^{(k)} \right|$$

e, se $r \neq k$, scambiare l'equazione k-esima con la r-esima.

Pivoting completo: scegliere una coppia (r, s), con r e $s \geq k$ (la più vicina a (k, k) per esempio) tale che

$$a_{rs}^{(k)} = \max_{k \leq i,j \leq n} \left| a_{ij}^{(k)} \right|$$

e scambiare l'equazione k-esima con la r-esima e l'incognita k-esima (con il suo coefficiente) con la s-esima.

Poiché il pivoting parziale risulta generalmente soddisfacente, quello completo, a causa dell'eccessivo lavoro di ricerca dell'elemento massimo è poco usato.

Nel caso di matrici dei coefficienti simmetriche e a diagonale dominante è possibile dimostrare che l'introduzione della strategia del pivoting parziale è del tutto superflua in quanto essa non provoca alcun scambio di equazioni. Ricordiamo infine che quando la matrice A è simmetrica definita positiva, l'algoritmo di Gauss senza pivoting risulta numericamente stabile.

7.4 - Fattorizzazione LU

La fattorizzazione LU permette di calcolare in maniera molto efficiente la soluzione di un sistema di equazioni lineari tramite il calcolo di due particolari matrici L e U, rispettivamente triangolare superiore e triangolare inferiore.

7.4.1 - Risoluzione di sistemi tramite fattorizzazione LU

Risolvere una generica equazione tra matrici $Ax = b$ utilizzando il metodo della fattorizzazione LU consiste nel trovare due matrici L (per Lower, matrice triangolare inferiore) e U (per Upper, matrice triangolare superiore) tali che $PA = LU$, ove P è una matrice permutazione, ossia una matrice che moltiplicando A ne permuta le righe.

Una volta che le matrici L e U sono a nostra disposizione, come possiamo trovare il corretto valore di x? Ricordiamo le ipotesi fatte finora:

(1) $PA = LU$

(2) $Ax = b$

Da (2) abbiamo quindi che $Pb = (PA)x$, per cui per (1) $Pb = (LU)x$. Assegnando ad una variabile y il risultato di Ux, abbiamo quindi che la risoluzione il sistema originale (trovare i valori delle incognite x) è equivalente alla risoluzione del seguente sistema:

$$\begin{cases} Ly = Pb \\ Ux = y \end{cases}$$

Il vantaggio di risolvere un tale sistema di equazioni è evidente: essendo L e U matrici triangolari il costo computazione delle operazioni in cui sono coinvolte è molto basso.

Ma il vantaggio della fattorizzazione LU, in termini di costo computazionale, non si limita a questo: supponiamo infatti di voler risolvere i seguenti sistemi:

$$\begin{gathered} Ax_1 = b_1 \\ Ax_2 = b_2 \\ Ax_3 = b_3 \\ \dots \\ Ax_n = b_n \end{gathered}$$

Questi sistemi presentano tutti la medesima matrice dei coefficienti A, ma vettore dei termini noti diverso. Poiché $LU = PA$ è evidente che il cambio nella matrice dei coefficienti b non influirà su L e U: sarà quindi sufficiente calcolare L e U la prima volta per poter risolvere molto velocemente tutti questi sistemi.

Ci proponiamo di interpretare il metodo di Gauss come successione finita di trasformazioni della matrice dei coefficienti A e del termine noto b, ovvero come moltiplicazione di A e b per un numero finito di opportune matrici. Questa particolare interpretazione ci consentirà di riformulare un algoritmo di Gauss in 2 parti distinte: una, la più costosa in termini di operazioni aritmetiche, determinerà una matrice triangolare non singolare G, tale che $GA = U$ ed è di forma triangolare superiore; l'altra ci consentirà di risolvere il sistema $Ax = b$ utilizzando la matrice G.

Infatti $GAx = Gb \Rightarrow Ux = Gb = y$. Si noti che $y = Ux$ è una delle equazioni del sistema in 7.4.1.

Osserviamo che lo scambio di 2 equazioni nel sistema $Ax = b$ (per esempio la riga i-esima con la riga j-esima) può essere interpretato come il risultato del prodotto di entrambi i membri del sistema per la matrice di permutazione P_{ij}, ossia la matrice ottenuta dalla matrice identità I scambiando le righe i e j.

Esempio: $P_{12}\begin{bmatrix}2 & 3\\ 4 & 5\end{bmatrix} = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\begin{bmatrix}2 & 3\\ 4 & 5\end{bmatrix} = \begin{bmatrix}4 & 5\\ 2 & 3\end{bmatrix}$

Analogamente la sostituzione nel sistema dell'equazione i-esima con la medesima più la j-esima moltiplicata per il coefficiente m_{ij}, (lo stesso di Gauss) può essere ottenuta moltiplicando $Ax = b$ per la matrice M_{ij},, ottenuta dalla matrice I ove la cella nella riga i e colonna j ha valore m_{ij}.

$$M_{ij} = \begin{pmatrix} 1 & 0 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & 0 & 1 & \dots & \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & 1 & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & m_{ij} & \dots & \dots & 0 \\ 0 & 0 & 0 & \dots & \dots & \dots & \dots & 1 \end{pmatrix}$$

In particolare possiamo poi definire M_j in questo modo:

$$M_j = M_{n,j} \dots M_{j+2,j} M_{j+1,j} = \begin{pmatrix} 1 & 0 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & 1 & 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & 0 & 1 & \dots & \dots & \dots & \dots & 0 \\ 0 & 0 & 0 & 1 & \dots & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & 1 & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & m_{ij} & \dots & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & m_{nj} & \dots & \dots & 1 \end{pmatrix}$$

Pertanto con il metodo di Gauss determiniamo implicitamente delle matrici $P_1, \cdots, P_{n-1}$ di tipo I se non vengono scambiate le righe i e j, o altrimenti di tipo P_{ij} e delle matrici $M_1, M_2, \dots, M_{n-1}$ con $M_j = M_{n,j}\, M_{n-1,j}\, \dots\, M_{j+2,j}\, M_{j+1,j}$ tali che il nuovo sistema

$$M_{n-1}P_{n-1} \dots M_1P_1Ax = M_{n-1}P_{n-1} \dots M_1P_1b$$

sia di forma triangolare superiore (come con Gauss) $Ux = y$ ossia:

$$M_{n-1}P_{n-1} \dots M_1P_1A = U$$

Posto $G = M_{n-1}P_{n-1} \dots M_1P_1$, denotiamo questa decomposizione come $GA = U$

Grazie alle particolari proprietà delle matrici M e P si possono riordinare le moltiplicazioni come:

$$M_{n-1}P_{n-1} \dots M_1P_1 = (\bar{M}_{n-1} \dots \bar{M}_1)(P_{n-1} \dots P_1)$$

ottenendo

$$\bar{M}_{n-1} \dots \bar{M}_1P_{n-1} \dots P_1Ax = \bar{M}_{n-1} \dots \bar{M}_1P_{n-1} \dots P_1b$$

le matrici M e $\bar{M}$ sono esattamente dello stesso tipo e differiscono solo per un diverso ordinamento (nell'ambito della stessa colonna e sotto l'elemento diagonale) dei moltiplicatori m_{ij} in considerazione degli scambi di riga effettuati in precedenza.

posto $\bar{M} = \bar{M}_{n-1} \dots \bar{M}_1$ e $P = P_{n-1} \dots P_1$, da cui segue che $G = \bar{M}P$, si ha:

$$\bar{M}PAx = \bar{M}Pb$$

Inoltre abbiamo visto in precedenza che $GA = U$, per cui:

$$\bar{M}PA = U$$

ossia

$$PA = \bar{M}^{-1}U$$

e di conseguenza siccome $LU = PA$ si ha proprio che $\bar{M}^{-1}$ è la matrice L cercata.

La forma di questa matrice L é proprio la seguente:

$$L = \bar{M}^{-1} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -\bar{m}_{21} & 1 & 0 & \cdots & 0 \\ -\bar{m}_{31} & -\bar{m}_{32} & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ -\bar{m}_{n1} & -\bar{m}_{n2} & -\bar{m}_{n3} & \cdots & 1 \end{pmatrix}$$

in questa matrice una colonna di coefficienti $\{\bar{m}_{j+1,j}, \bar{m}_{j+2,j}, \dots, \bar{m}_{n,j}\}$ denota un diverso ordinamento della colonna dei coefficienti $\{m_{j+1,j}, m_{j+2,j}, \dots, m_{n,j}\}$ nuovamente proprio in conseguenza degli scambi di riga effettuati in precedenza.

7.4.2 – Osservazioni sull'implementazione

Nelle applicazioni non serve costruire la matrice G in quanto è sufficiente memorizzare i moltiplicatori m_{ij} e le permutazioni effettuate. Possiamo inoltre salvare questi coefficienti negli spazi liberi della matrice G così che alla fine avremo:

$$G = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n} \\ m_{21} & u_{22} & u_{23} & \dots & u_{2n} \\ m_{31} & m_{32} & u_{33} & \dots & u_{3n} \\ \dots & \dots & \dots & \dots & \dots \\ m_{m1} & m_{m2} & m_{m3} & \dots & u_{mn} \end{bmatrix}$$

Possiamo inoltre ricordarci le permutazioni effettuate semplicemente utilizzando un vettore che chiameremo pivot. In questo vettore si ha $pivot[k] = j$ se k e j sono scambiate o $pivot[k] = k$ se k rimane al suo posto.

7.4.3 – Gli algoritmi **`solve`** e **`factor`**

Gli algoritmi necessari a risolvere un sistema con la fattorizzazione LU sono due: **`factor`** e **`solve`**:

`factor(n, A, pivot, det, ier)`: questo algoritmo si preoccupa di determinare la decomposizione $GA = U$ di una matrice A di ordine n. Alla fine dell'esecuzione A conterrà nella parte superiore i coefficienti della matrice triangolare superiore U, mentre nella parte inferiore i moltiplicatori m_{ij}. Il vettore pivot contiene tutti gli scambi di riga effettuati in precedenza, come descritto nel paragrafo precedente.

Infine **`det`** contiene il valore del determinante e **`ier`** eventuali codici di errore.

`solve(n, A, pivot, b)`: questo algoritmo utilizza i risultati forniti da **`factor`** e determina con n^2 operazioni aritmetiche la soluzione x, risolvendo il sistema non singolare di ordine n

$$Ux = M_{n-1}P_{n-1} \dots M_1P_1b$$

ove ricordiamo che i valori di U e dei coefficienti m_{ij} sono salvati nella matrice A come descritto in **`factor`**.

A è quindi la matrice ottenuta con **`factor`**, come anche `pivot` contiene gli scambi di riga così ottenuti: alla fine dell'esecuzione dell'algoritmo il vettore b conterrà la soluzione.

Il costo computazionale dei due algoritmi è $O(n^2)$ per il primo e $O(n^3/3)$ per il secondo.

7.4.4 - Calcolo dell'inversa di una matrice tramite fattorizzazione LU

Consideriamo una generica matrice non singolare A, e osserviamo che la conoscenza della sua inversa A^{-1} riduce la soluzione del sistema $Ax = b$ al semplice prodotto

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

Ciò sembra suggerire quindi il calcolo di A^{-1} specialmente quando si presenta la necessità di dover risolvere più sistemi del tipo

$$Ax_i = b_i, \qquad i = 1,2,3, \dots, p$$

tutti con la stessa matrice A. Allo scopo di calcolare la matrice inversa di una matrice non singolare generica A ci giunge in aiuto la fattorizzazione LU.

Partiamo dalla nostra ipotesi:

$$PA = LU$$

moltiplichiamo a sinistra per P^{-1}

$$P^{-1}PA = P^{-1}LU$$

$$A = P^{-1}LU$$

moltiplichiamo a destra per U^{-1}:

$$AU^{-1} = P^{-1}LUU^{-1}$$

$$AU^{-1} = P^{-1}L$$

moltiplichiamo a destra per L^{-1}:

$$AU^{-1}L^{-1} = P^{-1}LL^{-1}$$

$$AU^{-1}L^{-1} = P^{-1}$$

moltiplichiamo a destra per P

$$AU^{-1}L^{-1}P = P^{-1}P$$

$$AU^{-1}L^{-1}P = I$$

moltiplichiamo a sinistra per A^{-1}

$$A^{-1}AU^{-1}L^{-1}P = A^{-1}$$

$$U^{-1}L^{-1}P = A^{-1}$$

Come possiamo usare la formula trovata ($A^{-1} = U^{-1}L^{-1}P$) per calcolare facilmente A^{-1}? Ancora una volta il fatto che L e U siano matrici triangolari ci viene in aiuto: infatti, nonostante si debba calcolare l'inverso di queste matrici, grazie alle loro proprietà questo può essere fatto senza troppi passaggi, come vedremo nel paragrafo che segue.

7.4.5 - Calcolo della matrice inversa di una matrice triangolare

Supponiamo di voler trovare $Y = L^{-1}$, ossia la matrice tale che $YL = I$. Sappiamo che L è della forma:

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

ed inoltre sappiamo che I è quella matrice che ha per elementi tutti 0 tranne per la diagonale i cui elementi sono tutti 1. Poiché sappiamo che $LY = I$ automaticamente sappiamo che gli elementi di LY non sulla diagonale devono essere uguali a 0, mentre gli elementi sulla diagonale devono essere 1.

Supponendo quindi (grazie all'algebra sappiamo che l'inversa di una matrice triangolare inferiore è essa stessa una matrice triangolare inferiore) :

$$Y = \begin{bmatrix} y_{11} & 0 & 0 & \dots & 0 \\ y_{21} & y_{22} & 0 & \dots & 0 \\ y_{31} & y_{32} & y_{33} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \\ y_{n1} & y_{n2} & y_{n3} & \dots & y_{nn} \end{bmatrix}$$

possiamo scrivere LY in questo modo:

$$LY = \begin{bmatrix} l_{11}y_{11} & 0 & 0 & \dots & 0 \\ l_{21}y_{12} + l_{22}y_{21} & l_{22}y_{22} & 0 & \dots & 0 \\ l_{31}y_{11} + l_{32}y_{12} + l_{33}y_{13} & l_{32}y_{22} + l_{33}y_{33} & l_{33}y_{33} & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots \end{bmatrix}$$

si può quindi scrivere Y semplicemente osservando che:

$$l_{ii}y_{ii} = 1 \qquad \forall i \Rightarrow y_{ii} = \frac{1}{l_{ii}}$$

$$y_{ij} = \frac{-\sum_{k=j}^{i-1} l_{ik}y_{kj}}{l_{ii}}, \qquad i = j+1, \dots, n$$

pertanto calcolare l'inversa di una matrice triangolare è quindi un'operazione molto semplice.

7.4.6 – Metodo di Choleski

Il metodo di Choleski è un caso particolare della fattorizzazione LU. Questo metodo può essere applicato per risolvere equazioni del tipo $Ax = b$ ove A rispetta alcune particolari proprietà:

A simmetrica ($a_{ij} = a_{ji}$): se A è simmetrica si dimostra che $P = I$ (matrice identità) e che vale la seguente equazione $A = LDU$, con L e U matrici triangolare inferiore e superiore e D matrice diagonale (la matrice P è tale che $a_{ij} = 1$ se $i = j$ o $a_{ij} = 0$ altrimenti).

A definita positiva (ovvero non ci sono state permutazioni): se A è definita positiva si dimostra che gli elementi $(D)_{ii}$ sono tutti positivi, che $A = L_1 L_1^T$ dove $L_1 = LD^{1/2}$, con L matrice triangolare inferiore e $D^{1/2}$ la matrice diagonale tale che $(D^{1/2})_{ii} = \sqrt{(D_{ii})}$ con $i = 1,2, \dots, n$.

Gli elementi di tale matrice L_1 sono:

$$i = 1,2, \cdots, n: \qquad \begin{cases} l_{ii} = \sqrt{(a_{ii})} \\ l_{ij} = (a_{ij} - \sum_{k=1}^{j-1} l_{ik} l_{jk}) / l_{jj}\,, \qquad j = 1, \dots, i-1 \\ l_{ii} = (a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2)^{\frac{1}{2}} \end{cases}$$

Il costo computazione del metodo di Cholesky è $O(n^3/6)$, la metà di quello della fattorizzazione LU che è $O(n^3/3)$.

Capitolo 8 - Autovalori

8.1 - Generalità

8.1.1 - Definizione

Dati una matrice $A \in \mathbb{R}^{n \cdot n}$ e uno scalare $\lambda \in \mathbb{C}$, λ si dice *autovalore* per A se:

$$(1) \qquad \exists \vec{x} \neq \vec{0} \ : \ A\vec{x} = \lambda\vec{x}$$

In tal caso $\vec{x}$ si dice *autovettore* per A.

8.1.2 - Conseguenze

Dall'equazione (1) si ottiene il seguente risultato:

$$A\vec{x} - \lambda\vec{x} = \vec{0}, \quad \vec{x} \neq \vec{0}$$

$$(2) \qquad (A - \lambda I)\vec{x} = 0, \quad \vec{x} \neq 0$$

Dove I è la matrice identità, ovvero quella matrice che ha la diagonale principale di soli 1 e tutto il resto di soli 0. Questo ha come conseguenza che l'equazione (2) ammette una radice diversa da quella banale.

Questo significa che:

$$(3) \qquad det(A - \lambda I) = 0$$

Dal momento che il determinante è dato dalla seguente equazione:

$$det(A - \lambda I) = \lambda^n + \alpha_1\lambda^{n-1} + \alpha_2\lambda^{n-2} + \ldots + \alpha_{n-1}\lambda + \alpha_n$$

gli autovalori λ_i sono le radici dell'equazione (3).

Dato, dunque, che il calcolo degli autovalori equivale al calcolo delle radici dell'equazione, quello che ci interessa è trovare un metodo efficiente per il calcolo degli autovalori.

Per capire come trovare un siffatto metodo, è utile definire i termini di questa ricerca chiedendosi se:

- è sufficiente trovare l'autovalore di modulo maggiore;
- sono richiesti *tutti* gli autovalori;

- la matrice presenta una o più delle seguenti proprietà:
    - è simmetrica (matrice uguale alla sua trasposta);
    - è tridiagonale (matrice con soli 0 tranne sulla diagonale principale e sulle due diagonali adiacenti alla principale);
    - è sparsa (matrice con un'ampia maggioranza di 0);

La presenza di tali proprietà può infatti ridurre molto la complessità di operazioni quali il calcolo del determinante e sue correlate.

Inoltre una proprietà da cui partire per trovare una soluzione efficiente al nostro problema è la seguente:

- una matrice $A \in \mathbb{R}^{n \cdot n}$ ammetterà n autovalori λ_i ed n autovettori $\overrightarrow{x_i}$ corrispondenti:

$$A\overrightarrow{x_1} = \lambda_1\overrightarrow{x_1}$$

$$A\overrightarrow{x_2} = \lambda_2\overrightarrow{x_2}$$

$$\dots$$

$$\dots$$

$$A\overrightarrow{x_n} = \lambda_n\overrightarrow{x_n}$$

8.2 - Teorema di Gershgorin

8.2.1 - Definizioni

Per illustrare il t*eorema di Gershgorin* è necessario fornire alcune definizioni di base:

- somma del valore assoluto di tutti gli elementi della riga *i* eccetto quello che si trova sulla colonna *i*:

$$r_i = \sum_{j=1, j \neq i}^{n} |a_{ij}|$$

- somma del valore assoluto di tutti gli elementi della colonna *j* eccetto quello sulla colonna *j*:

$$c_i = \sum_{i=1, i \neq j}^{n} |a_{ij}|$$

- Cerchi di Gershgorin:

$$R_i = \{z : |z - a_{ii}| \leq r_i, \qquad z \in \mathbb{C}\}$$

Sono i cerchi del piano complesso $\mathbb{C}$ di centro a_{ii} e raggio r_i.

$$C_i = \{z : |z - a_{jj}| \leq c_j, \qquad z \in \mathbb{C}\}$$

Sono i cerchi del piano complesso $\mathbb{C}$ di centro a_{jj} e raggio c_j.

- Unione di tutti i cerchi R_i e C_i, rispettivamente:

$$R = \bigcup_{i=1}^{n} R_i$$

$$C = \bigcup_{i=1}^{n} C_j$$

- Dai suddetti insiemi possiamo derivare la seguente proprietà:

$$\exists \lambda : \lambda \in R$$

$$\exists \lambda : \lambda \in C$$

che implicano banalmente:

$$\exists \lambda : \lambda \in R \cup C$$

Una volta definiti questi concetti, si può passare alla definizione del teorema in oggetto.

8.2.2 - Il teorema di Gershgorin

Data una matrice A e gli insiemi R e C definiti sopra, gli autovalori di A appartengono a $R \cap C$.

Ogni componente di R o di C, cioè ogni unione connessa massimale di cerchi R_i o C_i, contiene tanti autovalori di A, quanti sono i cerchi della componente, tenendo conto della molteplicità di ogni autovalore e di ogni cerchio.

Il seguente esempio dovrebbe aiutare a chiarire le idee.

Applichiamo il teorema alla seguente matrice:

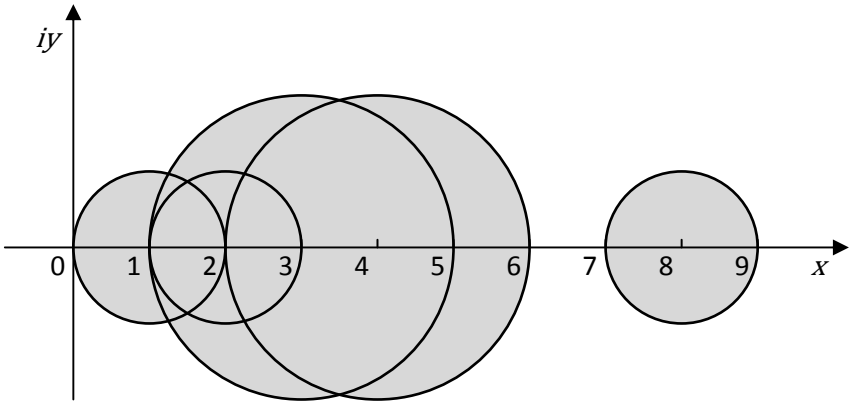
$$A = \begin{pmatrix} 4 & -1 & 1 & 0 & 0 \\ 1 & 3 & -1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 1 & 8 \end{pmatrix}$$

i cui autovalori sono:

$$\lambda_1 = 5 + \sqrt{10}$$
$$\lambda_2 = 3\,x$$
$$\lambda_3 = 3$$
$$\lambda_4 = 2$$
$$\lambda_5 = 5 - \sqrt{10}$$

Tutti gli autovalori appartengono al seguente insieme R di cerchi:

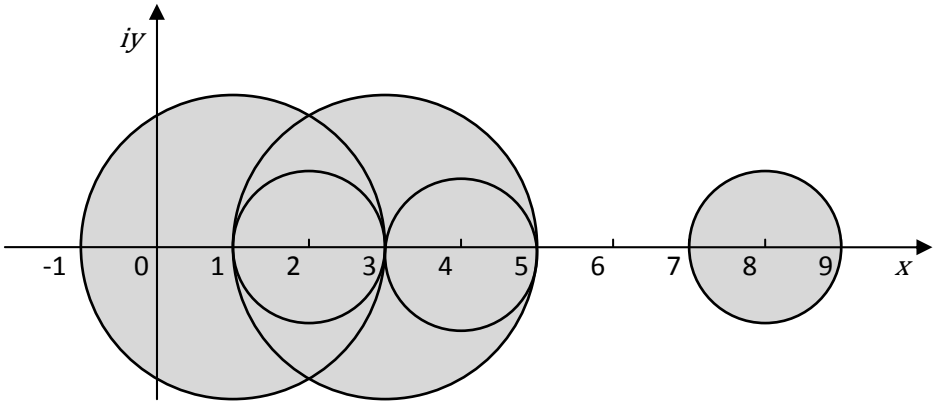
$$R_1 = \{z : |z-4| \leq 2\}$$
$$R_2 = \{z : |z-3| \leq 2\}$$
$$R_3 = \{z : |z-1| \leq 1\}$$
$$R_4 = \{z : |z-2| \leq 1\}$$
$$R_5 = \{z : |z-8| \leq 1\}$$



La regione R è formata dalle componenti colorate di grigio. In una vi sono 4 cerchi e quindi 4 autovalori; l'altra, essendo formata da 1 solo cerchio, contiene 1 autovalore.

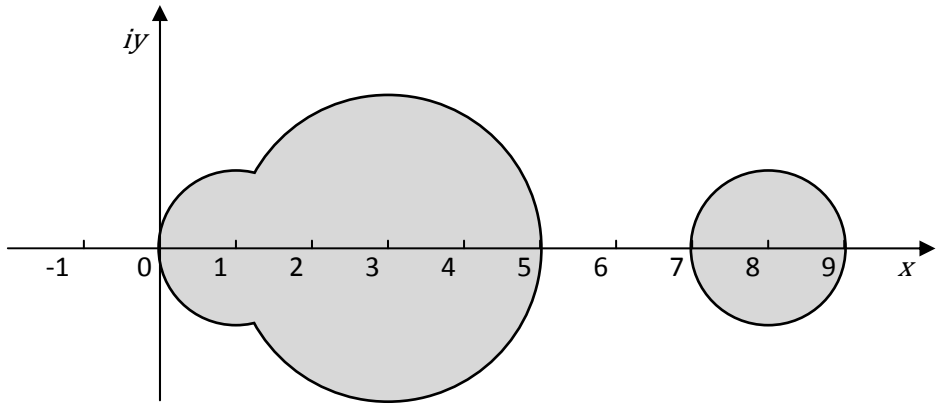
Tutti gli autovalori appartengono però anche alla regione C unione dei cerchi C_i:

$$C_1 = \{z : |z-4| \leq 1\}$$
$$C_2 = \{z : |z-3| \leq 2\}$$
$$C_3 = \{z : |z-1| \leq 2\}$$
$$C_4 = \{z : |z-2| \leq 1\}$$
$$C_5 = \{z : |z-8| \leq 1\}$$



La regione C è formata dalle componenti colorate di grigio: la prima contiene 4 autovalori, la seconda 1 .

Possiamo pertanto concludere che tutti gli autovalori devono giacere nella regione $R \cap C$:



8.3 - Metodo delle potenze

Il *metodo delle potenze* è un metodo utilizzato per trovare l'autovalore di modulo massimo. Tale autovalore, infatti, è di estrema importanza in molti problemi numerici come il calcolo del numero di condizionamento di un sistema di equazioni lineari.

8.3.1 - Descrizione del metodo delle potenze

Per applicare tale metodo è necessario che valgano le seguenti ipotesi:

$$\text{hp1: } |\lambda_1| > |\lambda_2| > \dots > |\lambda_n|$$

hp2: I corrispettivi autovettori $x_1, x_2, \dots, x_n$ formano un sistema linearmente indipendente:

$$Ax_1 = \lambda_1 x_1$$

$$Ax_2 = \lambda_2 x_2$$

$$\dots$$

$$Ax_n = \lambda_n x_n$$

Date queste ipotesi possiamo affermare che:

$$\lambda_1 \in \mathbb{R} \wedge \lambda_1 \notin \mathbb{C}$$

Infatti se $\lambda_1 = x + iy$ appartenesse a C sarebbe uguale in modulo al suo coniugato $x - iy$, contravvenendo alle nostre ipotesi. Più formalmente:

$$\lambda_1 \in \mathbb{C} \Rightarrow \exists \lambda_2 = \bar{\lambda_1} : |\lambda_2| = |\lambda_1|$$

Detto questo possiamo notare che ogni vettore v_0 può essere scritto nel seguente modo:

$$v_0 = \alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n, \qquad \alpha_1 \neq 0$$

Ora prendiamo un vettore v_0 qualsiasi e scriviamo il vettore v_1 come il prodotto della matrice A per v_0:

$$v_1 = Av_0 = A(\alpha_1 x_1 + \alpha_2 x_2 + \ldots + \alpha_n x_n) = \alpha_1 Ax_1 + \alpha_2 Ax_2 + \ldots + \alpha_n Ax_n$$

Ricordando l'**hp2** e le definizione di autovalore λ_i corrispondente al generico autovettore x_i:

$$Ax_i = \lambda_i x_i$$

possiamo sostituire tutti gli Ax_i in v_1 con $\lambda_i x_i$, ottenendo:

$$v_1 = \alpha_1\lambda_1 x_1 + \alpha_2\lambda_2 x_2 + \ldots + \alpha_n\lambda_n x_n$$

$$v_1 = \lambda_1\left(\alpha_1 x_1 + \alpha_2\frac{\lambda_2}{\lambda_1}x_2 + \ldots + \alpha_n\frac{\lambda_n}{\lambda_1}x_n\right)$$

Ripetiamo il passo 1. con il vettore v_2:

$$v_2 = Av_1 = A(\alpha_1\lambda_1 x_1 + \alpha_2\lambda_2 x_2 + \ldots + \alpha_n\lambda_n x_n) = \alpha_1\lambda_1 Ax_1 + \alpha_2\lambda_2 Ax_2 + \ldots + \alpha_n\lambda_n Ax_n$$

E ripetendo il passo 2. otteniamo:

$$v_2 = \alpha_1\lambda_1^2 x_1 + \alpha_2\lambda_2^2 x_2 + \ldots + \alpha_n\lambda_n^2 x_n$$

$$v_2 = \lambda_1^2\left[\alpha_1 x_1 + \alpha_2\left(\frac{\lambda_2}{\lambda_1}\right)^2 x_2 + \ldots + \alpha_n\left(\frac{\lambda_n}{\lambda_1}\right)^2 x_n\right]$$

E ancora:

$$v_3 = \lambda_1^3\left[\alpha_1 x_1 + \alpha_2\left(\frac{\lambda_2}{\lambda_1}\right)^3 x_2 + \ldots + \alpha_n\left(\frac{\lambda_n}{\lambda_1}\right)^3 x_n\right]$$

$$v_4 = \lambda_1^4\left[\alpha_1 x_1 + \alpha_2\left(\frac{\lambda_2}{\lambda_1}\right)^4 x_2 + \ldots + \alpha_n\left(\frac{\lambda_n}{\lambda_1}\right)^4 x_n\right]$$

...

Più in generale il vettore k-esimo sarà il seguente:

$$v_k = \lambda_1^k\left[\alpha_1 x_1 + \alpha_2\left(\frac{\lambda_2}{\lambda_1}\right)^k x_2 + \ldots + \alpha_n\left(\frac{\lambda_n}{\lambda_1}\right)^k x_n\right]$$

Dal momento che non è definita l'operazione di rapporto tra vettori, calcoliamo il rapporto delle componenti j-esime dei vettori v_k e v_{k+1}:

$$\frac{(v_{k+1})_j}{(v_k)_j} = \frac{\lambda_1^{k+1}\left[\alpha_1 x_1 + \alpha_2\left(\frac{\lambda_2}{\lambda_1}\right)^{k+1} x_2 + \ldots + \alpha_n\left(\frac{\lambda_n}{\lambda_1}\right)^{k+1} x_n\right]}{\lambda_1^{k}\left[\alpha_1 x_1 + \alpha_2\left(\frac{\lambda_2}{\lambda_1}\right)^{k} x_2 + \ldots + \alpha_n\left(\frac{\lambda_n}{\lambda_1}\right)^{k} x_n\right]}$$

Ora calcoliamo il limite per k che tende ad infinito del suddetto rapporto:

$$\lim_{k\to+\infty}\frac{(v_{k+1})_j}{(v_k)_j} = \lim_{k\to+\infty}\frac{\lambda_1^{k+1}\left[\alpha_1 x_1 + \alpha_2\left(\frac{\lambda_2}{\lambda_1}\right)^{k+1} x_2 + \ldots + \alpha_n\left(\frac{\lambda_n}{\lambda_1}\right)^{k+1} x_n\right]}{\lambda_1^{k}\left[\alpha_1 x_1 + \alpha_2\left(\frac{\lambda_2}{\lambda_1}\right)^{k} x_2 + \ldots + \alpha_n\left(\frac{\lambda_n}{\lambda_1}\right)^{k} x_n\right]}$$

$$\frac{\lim_{k\to+\infty}(v_{k+1})_j}{\lim_{k\to+\infty}(v_k)_j} = \frac{\lim_{k\to+\infty}\lambda_1^{k+1}\left[\alpha_1 x_1 + \alpha_2\left(\frac{\lambda_2}{\lambda_1}\right)^{k+1} x_2 + \ldots + \alpha_n\left(\frac{\lambda_n}{\lambda_1}\right)^{k+1} x_n\right]}{\lim_{k\to+\infty}\lambda_1^{k}\left[\alpha_1 x_1 + \alpha_2\left(\frac{\lambda_2}{\lambda_1}\right)^{k} x_2 + \ldots + \alpha_n\left(\frac{\lambda_n}{\lambda_1}\right)^{k} x_n\right]}$$

Osserviamo che dall'**hp1** segue che:

$$\left(\frac{\lambda_i}{\lambda_1}\right)^{k+1} < 1, \qquad i \neq 1$$

quindi tutti i termini con coefficienti di questa forma si annullano:

$$\lim_{k\to+\infty}\left(\frac{\lambda_i}{\lambda_1}\right)^{k+1} = 0$$

da cui segue:

$$\frac{\lim_{k\to+\infty}(v_{k+1})_j}{\lim_{k\to+\infty}(v_k)_j} = \frac{\lim_{k\to+\infty}\lambda_1^{k+1}(\alpha_1 x_1)_j}{\lim_{k\to+\infty}\lambda_1^{k}(\alpha_1 x_1)_j} = \lambda_1$$

Abbiamo così trovato quello che per l'**hp1** è l'autovalore di modulo massimo. Si noti che tale convergenza è tanto più rapida quanto più è piccolo il rapporto:

$$\left|\frac{\lambda_1}{\lambda_2}\right|$$

8.4 - Metodo delle potenze inverse

8.4.1 - Premessa

- λ = autovalore di A (numero reale o complesso)
- A = matrice
- x = vettore ($x \neq 0$)

Partiamo dalla definizione di autovalore:

$$Ax = \lambda x$$

moltiplichiamo ambo i membri per l'inversa di A

$$A^{-1}Ax = A^{-1}\lambda x$$

semplifichiamo $A^{-1}A$

$$(I)x = A^{-1}\lambda x$$

dove *I* è la matrice identità

ora moltiplichiamo ambo i membri per λ^{-1} e sfruttando il fatto che il prodotto di una matrice per uno scalare è commutativo , avremo come equazione:

$$\lambda^{-1}x = \lambda^{-1}\lambda\, A^{-1}x$$

semplificando $\lambda^{-1}\lambda$ otteniamo la seguente proprietà:

proprietà 1

$$\boldsymbol{\lambda^{-1}x = A^{-1}x}$$

Se λ è autovalore di A allora λ^{-1} è autovalore di A^{-1}.

8.4.2 - Descrizione del metodo delle potenze inverse

Consideriamo ora il problema di raffinare un approssimazione grossolana p dell'autovalore λ (cioè conosciamo un approssimazione p del autovalore λ e vogliamo migliorarne l'accuratezza).

Quindi possiamo scrivere :

$$Ax = px$$

(abbiamo sostituito λ con la sua approssimazione p)

Per la definizione di autovalore di A abbiamo $Ax = \lambda x$ per cui :

$$(A - pI)x = Ax - px = \lambda x - px = (\lambda - p)x$$

quindi:

$$(A - pI)x = (\lambda - p)x$$

Per cui lo scalare $(\lambda - p)$ è autovalore per la matrice $(A - pI)$

Allora dalla proprietà 1:

$$(\lambda - p)^{-1}$$

$(\lambda - p)^{-1}$ è autovalore per la matrice $(A - pI)^{-1}$

Ora definiamo μ autovalore di $(A - pI)^{-1}$

$$\mu = \frac{1}{\lambda - p}$$

Quando l'approssimazione di p è buona (p è molto vicino a λ) allora la differenza $\lambda - p$ e molto piccola quindi μ diventa molto grande. Possiamo dunque immaginare che μ è molto grande e sia l'autovalore di massimo modulo per la matrice $(A - pI)^{-1}$.

Possiamo quindi applicare il metodo delle potenze (visto in precedenza) per trovare il valore approssimato di μ. Poiché $\lambda = (1/\mu) + p$ troverà un valore più accurato per λ.

Capitolo 9 - Problemi mal condizionati

9.1 Norma

Per definire l'indice di mal condizionamento di una matrice bisogna introdurre la nozione di norma.

9.1.1 - Norma di vettore

La norma di un vettore, definita con $\| \|$, è una funzione che ad ogni vettore $x \in \mathbb{R}^n$ associa un numero reale, che indichiamo con $\|x\|$.

Ci sono vari tipi di norma ma tutte si equivalgono a meno di una costante, nel senso che per ogni coppia di norme $\| \|^{(1)}$ e $\| \|^{(2)}$ esistono due costanti positive m e M tali che:

$$m\|x\|^{(2)} \leq \|x\|^{(1)} \leq M\|x\|^{(2)}$$

per tutti i vettori $x \in \mathbb{R}^n$, quindi è indifferente quale usare.

La norma di vettore gode delle seguenti proprietà:

$\|v\| = 0 \Leftrightarrow v = 0$

$\|v\| > 0 \Leftrightarrow v \neq 0$

$\|\alpha \cdot v\| = |\alpha| \cdot \|v\|, \qquad \forall \alpha \in \mathbb{R}$

$\|v + u\| \leq \|u\| + \|v\|$

$\|u\|_\infty = \max_i |u_i|$, detta norma infinito

$\|u\|_1 = \sum_i |u_i|$, detta norma 1

$\|u\|_2 = \sqrt{\sum_i u_i^2}$, detta norma 2 o norma quadratica

9.1.2 - Norma di matrice

La norma di una matrice, definita anch'essa con $\| \|$, è una funzione che ad ogni matrice $A \in \mathbb{R}^{n \cdot n}$ associa un numero reale, che indichiamo con $\|A\|$.

Come per le norme di vettore anche le norme di matrice si equivalgono a meno di una costante, nel senso che per ogni coppia di norme $\| \|^{(1)}$ e $\| \|^{(2)}$ esistono due costanti positive m e M tali che:

$$m\|A\|^{(2)} \leq \|A\|^{(1)} \leq M\|A\|^{(2)}$$

per tutti le matrici $A \in \mathbb{R}^{n \cdot n}$, quindi è, anche per esse, indifferente quale usare.

La norma di matrice gode delle seguenti proprietà:

$A \neq 0 \Rightarrow \|A\| > 0$

$\|A\| = 0 \Leftrightarrow A = 0$

$\|\alpha \cdot A\| = |\alpha| \cdot \|A\|, \qquad \forall \alpha \in \mathbb{R}$

$\|A + B\| \leq \|A\| + \|B\|,$ disuguaglianza triangolare

$\|A\|_\infty = \max_i \sum_j |a_{ij}|$

$\|A\|_1 = \max_j \sum_i |a_{ij}|$

$\|A\|_2 = \max_i |\lambda_i|,$ con λ_i autovalore di A

$\|A\|_F = \sqrt{\sum_i a_{ij}^2},$ norma di Frobenius

9.1.3 - Perché proprio le norme misurano il condizionamento?

Osservazione:

Dato un sistema lineare:

$$A \cdot x = b$$

con:

- A matrice dei coefficienti di dimensione $n \cdot n$;
- b vettore dei termini noti di n componenti;
- x vettore delle incognite di n componenti;

Se effettuo una lieve modifica (perturbazione) sul vettore dei termini noti b con una certa quantità Δb, voglio vedere che conseguenze comporta questa modifica sulla soluzione del sistema.

In dettaglio, se il termine noto sarà sostituito da $b + \Delta b$ il nuovo sistema sarà:

$$A \cdot (x + \Delta x) = b + \Delta b$$

con $(x + \Delta x)$ che rappresenta il nuovo vettore delle incognite

Ricapitolando abbiamo:

$$A \cdot x = b \qquad (1)$$
$$A \cdot (x + \Delta x) = b + \Delta b \qquad (2)$$

Sottraendo la prima (1) dalla seconda (2), otteniamo:

$$A \cdot \Delta x = \Delta b$$

Moltiplicando ambo i membri per A^{-1} si ottiene:

$$\Delta x = A^{-1} \cdot \Delta b$$

Passando alle norme:

$$\|\Delta x\| = \|A^{-1} \cdot \Delta b\|$$

utilizzando la diseguaglianza triangolare si ottiene:

$$\|A^{-1} \cdot \Delta b\| \leq \|A^{-1}\| \cdot \|\Delta b\|$$

Utilizzando la (1) abbiamo che:

$$\|b\| = \|A \cdot x\|$$

reimpiegando la diseguaglianza triangolare si ha che:

$$\|A \cdot x\| \leq \|A\| \cdot \|x\|$$

Calcoliamo ora il seguente rapporto utilizzando i risultati prima ottenuti:

$$\frac{\|\Delta x\|}{\|A\| \cdot \|x\|} \leq \frac{\|A^{-1} \cdot \Delta b\|}{\|b\|} \leq \frac{\|A^{-1}\| \cdot \|\Delta b\|}{\|b\|}$$

da cui si ottiene:

$$\frac{\|\Delta x\|}{\|x\|} \leq \|A\| \cdot \|A^{-1}\| \leq \frac{\|\Delta b\|}{\|b\|}$$

in cui la parte centrale è proprio l'indice di condizionamento della matrice A.

Quindi se introduco un piccolo errore sul termine noto , questa variazione si scarica sulla soluzione con coefficiente moltiplicativo $\|A\| \cdot \|A^{-1}\| = k(A)$ detto **indice di condizionamento.**

Quindi per vedere se un sistema è mal condizionato bisogna calcolare il $k(A)$.

Infine dalla seguente relazione :

$$1 = \|I\|_\infty = \|A\ A^{-1}\| = \|A\| \cdot \|A^{-1}\| = k(A)$$

Dove $I =$ *matrice identità* (una matrice quadrata in cui tutti gli elementi della diagonale principale sono costituiti dal numero 1, mentre i restanti elementi sono costituiti dal numero 0) segue che il valore dell'indice di condizionamento è sempre maggiore uguale a 1, cioè:

$$k(A) \geq 1$$

9.1.4 - Come cresce l'indice di condizionamento *k(A)*?

Se l'indice di condizionamento $k(A)$ cresce come *n* (linearmente con *n*, *n = dimensione del sistema*) allora è ben condizionato, mentre se cresce più velocemente, per esempio come e^n sarà mal condizionato (come ad esempio la matrice di Vandermonde).

Capitolo 10 - Equazioni differenziali: modellistica differenziale

In breve: la modellistica differenziale si propone di creare un modello della realtà servendosi delle equazioni differenziali. Il calcolo numerico ci fornisce i metodi per la soluzione di queste equazioni.

10.1 - Definizione

$$y'(x) = f\big(x\,, y(x)\big) \quad , \quad y\,?$$

Equazione differenziale: equazione in cui l'incognita compare anche nella sua derivata prima.

La derivata prima è una misura della variazione: consente l'analisi di fenomeni che si evolvono nello spazio e nel tempo (e.g., propagazione di un'onda sismica, previsioni meteo, diffusione di una malattia, inquinante che si sposta nell'atmosfera, etc...).

L'**Analisi** riesce a fornire una soluzione solo nei casi più semplici: è necessario perciò ricorrere al **Calcolo Numerico**, che può fornire una soluzione approssimata in un insieme finito di punti.

10.2 – Esempi

Sviluppo di tumori

Un'applicazione medica delle equazioni differenziali è legata allo studio di alcuni tipi di tumori. Ipotizzando un tumore sferico, è possibile studiare la sua crescita (o la sua regressione) rispetto ad alcuni parametri significativi.

Il raggio del tumore (la nostra incognita) è legato ai parametri dalla seguente formula

$$R'(t) = \frac{1}{3} S_i\, R + \frac{2\lambda\sigma}{\mu R + \sqrt{\mu^2 R^2 + 4\sigma}}$$

$$R(0) = a$$

dove:

R indica il raggio del tumore (ipotizzato sferico);

μ , λ parametri di scala;

S_i misura la velocità con cui le cellule decadono;

σ livello mutativo.

Diffusione di epidemie

Un'altra applicazione molto famosa delle equazioni differenziali è lo studio della diffusione delle epidemie. Volendo stimare la pericolosità di un'epidemia si ricorrono a strumenti di tipo Numerico (e Statistico): anche l'hanno scorso sono stati condotti studi di questa natura per la diffusione dell'influenza.

Modello SIR (Morbillo): trasmissione dell'immunità a coloro che sono guariti dalla malattia.

Popolazione divisa in tre categorie:

- $S(t) := sani$ $\quad \frac{dS}{dt} = -rSI$
- $I(t) := infetti$ $\quad \frac{dI}{dt} = rSI - aI$
- $R(t) := guariti$ $\quad \frac{dR}{dt} = aI$

dove: r , a misurano la velocità di contagio e quella di guarigione.

Qual è il valore critico di I tale che la malattia si propaga a tutta la popolazione?

Le equazioni differenziali legano tra di loro queste tre categorie, modellando la propagazione dell'epidemia. La soluzione delle equazioni attraverso tecniche numeriche ci consente di rispondere a questa, ed altre, domanda.

Un'applicazione più complessa delle equazioni differenziali in campo medico è nella TAC (Tomografia Assiale Computerizzata).

Equazioni di LOTKA – VOLTERRA (preda – predatore)

Equazioni differenziali che riescono a modellare le dinamiche di un ecosistema composto solo da due specie: una preda x ed un predatore y(e.g., lepri e volpi). Le prede dispongono di una fonte inesauribile di cibo, mentre i predatori possono nutrirsi soltanto delle prede x. In un ecosistema di questo tipo, il modo in cui una popolazione cresce (o decresce) rispetto all'altra può essere perfettamente modellato da equazioni differenziali: è possibile prevedere la situazione di queste due popolazioni dopo un determinato lasso di tempo.

Il modo in cui varia la popolazione dei predatori dipende da un opportuno coefficiente di crescita e dalla quantità di prede disponibili che un predatore incontra, a cui devo sottrarre il numero di prede necessarie per il nutrimento del predatore.

Sia:

$y(t) := \#\ di\ predatori\ presenti\ al\ tempo\ t$;

$m := \ quantità\ di\ cibo\ necessario\ alla\ sussistenza\ per\ un\ individuo$;

$a := \ coefficiente\ di\ crescita.$

$$y'(t) = a\,(x(t) - m)\,y(t)$$

$$y'(t) = a\,(\,x(t)\,y(t) - m\,y(t))$$

incontri predatore preda ← quantità di prede per il nutrimento

ponendo $C = a\,;\ D = am$

$$y'(x) = (C\,x(t) - D)y(t)$$

Il modo in cui varia la popolazione delle prede dipende dalla quantità iniziale di prede presenti nell'ecosistema, a cui devo sottrarre la quantità di prede che soccombono perché mangiate dai predatori.

Sia:

$x(t) := \#\ di\ prede\ presenti\ al\ tempo\ t.$

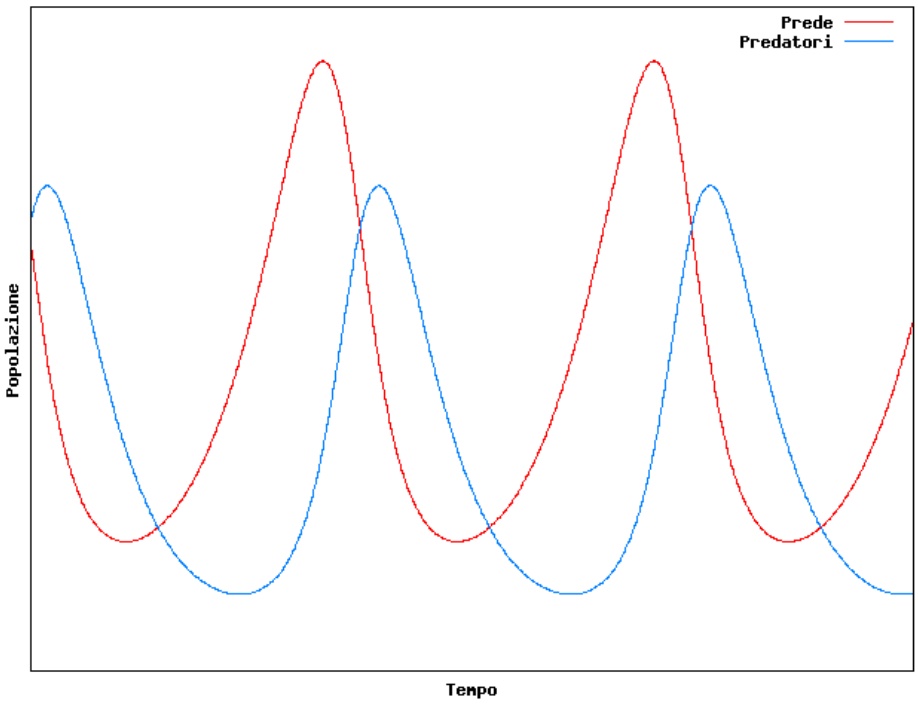
$$x'(t) = A\,x(t) - B\,x(t)\,y(t)$$

natalità al tempo t ← # di individui mangiati dai predatori

Possiamo riscrivere più semplicemente queste equazioni come:

$$\frac{dx}{dt} = (A - B_y)x \quad , \quad \frac{dy}{dt} = (C_x - D)y$$

Senza entrare nei dettagli su come si possano risolvere queste equazioni, è interessante notare come le due popolazioni abbiano un andamento ciclico. All'aumentare delle prede, inizialmente i predatori diminuiscono, ma trovandosi ora in una situazione di abbondanza di cibo iniziano ad aumentare, mangiando le prede. Queste iniziano perciò a diminuire, finché si raggiunge un punto in cui prede e predatori si equivalgono in numero. Ad un certo punto, i predatori sono troppi rispetto alle prede: ovvero non vi sono prede sufficienti a sfamare tutti i predatori. Questi iniziano perciò a morire, consentendo alle prede di proliferare di nuovo. Questo fenomeno si ripete sviluppando un andamento ciclico.



Le possibilità che risiedono nella modellistica differenziale hanno ampliato notevolmente i campi in cui viene impiegata: non solo in quelli matematici, ma anche in chimica, biologia, economia, etc... Per questo motivo, lo studio di metodi per la soluzione delle equazioni differenziali è stato e continua ad essere motivo di grande interesse.

10.3 – Tecniche numeriche

$$\begin{cases} y'(x) = y(x) \\ y(0) = 1 \end{cases} => \; y(x) = e^x$$

La prima formula è l'*equazione differenziale*, mentre la seconda è chiamata *condizione iniziale*. Quest'ultima ci fornisce un punto di partenza per l'equazione: rappresenta il valore della y valutata nel punto iniziale.

L'intervallo entro il quale si cerca la soluzione viene divisione in n intervalli uguali di ampiezza h :

$h := \; passo\ di\ integrazione,$

$x_i := \; nodo,$

$y(0) := \; punto\ d'innesco.$

Non essendo in grado di trovare analitcamente la soluzione esatta, cerco una soluzione approssimata ad un nodo x_i

$y_i := \; valore\ della\ soluzione\ approssimato\ al\ nodo\ x_i.$

10.3.1 - Metodo di Eulero

$y(x_{i+1}) - y(x_i) = y'(\xi)(x_{i+1} - \; x_i)$ dove $x_i < \; \xi \; < x_{i+1}$

non conosco l'esatta posizione di ξ => introduco un'approssimazione:

$$y'(\xi) \cong y'(x_i)$$

Se la funzione cresce rapidamente nel passo d'integrazione, l'approssimazione risulta essere molto grossolana.

ATTENZIONE: dal momento che introduco l'approssimazione, dovrò utilizzare la notazione numerica!

$$y_{i+1} - \; y_i = y'(\xi)(x_{i+1} - \; x_i) = h\, y'_i \quad => y_{i+1} = \; y_i + \; h\, y'_i$$

ma $y'(x) = f(x\,, y(x))$ quindi $y'(x_i) = f(x_i\,, y(x_i))$ ovvero $y_i = f(x_i\,, y_i)$

$$\boldsymbol{y_{i+1} = \; y_i + h\, f(x_i\,, y_i)} \longleftarrow \quad \text{Metodo di Eulero}$$

per $i \; = \; 0$

$y_1 = \; y_0 + h\, f(x_0\,, y_0)$ è un valore noto (conosco y_0 dalla condizione per iniziale)

$i = 1$

$y_2 = y_1 + h\, f(x_1, y_1)$ è noto conoscendo il risultato precedente (affetto 2 volte dall'errore)

per $i = 2$

$y_3 = y_2 + h\, f(x_2, y_2)$ è noto conoscendo il risultato precedente (affetto 3 volte dall'errore)

Propagazione (o Accumulo) dell'Errore

Vantaggi: semplicità.

Svantaggi: rapida propagazione dell'errore (approssimazione scadente).

10.3.2 - Metodo di Eulero – Cauchy

$$\frac{y(x_{i+1}) - y(x_{i-1})}{2\,h} \cong y'(x_i)$$

$$\frac{y_{i+1} - y_{i-1}}{2\,h} = y'_i \quad => y_{i+1} - y_{i-1} = 2\,h\,y'_i$$

ma $y'(x) = f(x, y(x))$ quindi $y_{i+1} - y_{i-1} = 2\,h\,f(x, y_i)$

$$\boldsymbol{y_{i+1} = y_{i-1} + 2\,h\,f(x_i, y_i)}$$ Metodo di Eulero-Cauchy

per $i = 1$

$y_2 = y_0 + 2\,h\,f(x_1, y_1)$ y_1 non è noto!

METODO NON AUTOPARTENTE: ha bisogno di un valore d'innesco aggiuntivo.

ATTENZIONE: se utilizzo Eulero per calcolare y_1 , immetterei un errore più grande, quindi non è utile!

Esistono tecniche basate sullo sviluppo in serie di Taylor che consentono di calcolare y_1 , ma richiedo l'utilizzo di derivate parziali (spesso onerose da calcolare).

Svantaggi: non auto-partente.

Vantaggi: semplice ed accurato.

10.3.3 - Metodi Multi-Step

$$\int_{x_i}^{x_{i+k}} y'(x)\,dx = \int_{x_i}^{x_{i+k}} f(x, y(x))\,dx$$

$$y_{i+k} - y_i = \sum_{j=0}^{k-1} \{a_j y_j + h b_j f(x_{i+j}, y_{i+j})\} + h b_k f(x_{i+k}, y_{j+k})$$

Metodi Predictor (Esplicito): conoscendo i valori nel caso precedente, posso conoscerli nel caso successivo. ($b_k = 0$) (e.g., metodo di Eulero ed Eulero-Cauchy)

Metodi Corrector (Implicito): il valore ignoto è legato ai valori precedenti e ad un valore di correzione.($b_k \neq 0$)

$$\begin{cases} y_{i+1} = y_{i-3} + \frac{4}{3} h\,(2y'_{i-2} - y'_{i-1} + 2y'_i) & \sim \text{esplicito} \\ y_{i-1} = y_{i-1} + \frac{4}{3}(y'_{i-1} + 4y'_{i+1} + y'_{i+1}) & \sim \text{implicito} \end{cases}$$

Metodo di Milne

La prima formula mi fornisce un valore approssimato dai valori precedenti, mentre la seconda raffina il risultato della prima, sostituendo$y'_{i+1} = f(x_{i+1}, y_{i+1})$.

Il Metodo di Milne si dimostra fornire risultati accurati.

10.3.4 - Metodi di Runge-Kutta

$$y_{i+1} = y_i + \frac{1}{2}(k_1 + k_2)$$

$$k_1 = h\,f(x_i, y_i)$$

$$k_2 = h\,f(x_i + h, y_i + k_1)$$

Vantaggi: auto-partenti, non richiedono derivate parziali.

Capitolo 11 – Codifica digitale

In breve:
La codifica digitale tratta i metodi di rappresentazione dell'immagine, in particolare verrà analizzata la trasformata di Fourier, che ha permesso di abbatterne i costi computazionali.

11.1 - Introduzione

Negli ultimi anni è stato riscontrato un grande aumento dell'utilizzo di macchine fotografiche di ultima generazione, le cosiddette macchine fotografiche digitali.
Si è passati, dunque, dalla fotografia analogica alla fotografia digitale, la nostra domanda è "Come può una macchina digitale (un minicomputer) rappresentare un'immagine?"
Analizziamo il funzionamento delle 2 macchine.
La macchina fotografica analogica è caratterizzata da 2 elementi fondamentali: L'otturatore e la pellicola.
La fonte di luce emette un segnale continuo, che passa attraverso l'otturatore e impressiona la pellicola.
In questo caso le operazioni sull'immagine sono facilmente eseguibili.
La macchina fotografica digitale è caratterizzata da sensori CMOS in grado di distinguere i 3 colori RGB (Rosso, Verde e Blu), per ogni colore è possibile rappresentare l'intensità della luce che ha colpito il sensore in una specifica area.
L'immagine viene rappresentata da una matrice discreta di punti, più grande è la matrice maggiore sarà la risoluzione dell'immagine.
L'obbiettivo è quello di gestire in maniera efficiente e corretta la matrice, quindi analizzeremo tecniche e algoritmi per la compressione delle immagini.

11.2 – Algoritmi di compressione

In generale gli algoritmi di compressione si dividono in 2 categorie: Lost e Lostless.
Gli algoritmi Lost sono caratterizzati da una perdita di informazioni (esempi: Jpeg, Mp3), gli algoritmi Lostless invece rappresentano l'informazione senza alcuna perdita (esempio: compressione di testo).
JPEG: Trasforma l'immagine naturale (circa 60 MB) in una immagine di pochi MB, perdendo informazione.
COMPRESSIONE TESTO: Analizza la frequenza dei caratteri in un testo e destina pochi bit per la rappresentazione dei caratteri piu frequenti e tanti bit per la rappresentazione di caratteri meno frequenti.
Analizziamo, come esempio, una macchina moderna da 18000 pixel, il che vuol dire avere una matrice 3000x6000, dove ogni bit rappresenta un pixel.
In realtà le matrici sono 3, una per ogni colore fondamentale.

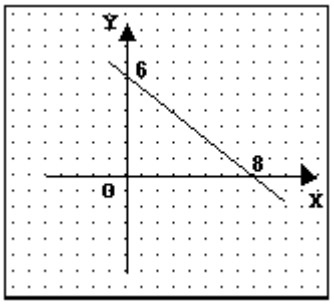
11.2.1 – Metodo Accetta

E' la tecnica piu semplice da implementare, consiste nel considerare le colonne pari o le colonne dispari, e scartare le altre, e operare la stessa operazione per le righe.
I pixel che fanno parte delle colonne o delle righe scartate vengono uguagliati ai loro adiacenti, oppure approssimati ad una media dei valori dei loro adiacenti.

Questa tecnica ha un enorme collo di bottiglia: la perdita di informazioni, infatti oltre a ridurre di un fattore 4 la matrice che rappresenta l'immagine (da 3000x6000 si passa a 1500x3000), si perde molta informazione che potrebbe essere cruciale per l'immagine.
In questo modo, una macchina avanzata da 18000 pixel diventa equivalente ad una macchina fotografica piu scadente da 4500 pixel.

11.2.2 – Rappresentazione parametrica

Il seguente esperimento ci aiuta a capire come possono essere rappresentate le immagini nella nostra macchina.
Spegnendo tutte le luci in una stanza e accendendo una lampadina da un'estremità della stanza, man mano che ci allontaniamo dalla sorgente luminosa, l'intensità luminosa diminuisce.
Questo fenomeno possiamo rappresentarlo in un grafico con una retta:



La sua equazione è $I = a + bi$
i = indice pixel
a,b = parametri retta.

Utilizziamo 2 coefficienti per disegnare ciò che viene rappresentato da 18000 pixel.
Il trucco è non considerare tutti i pixel, cioè non dare una rappresentazione pixel x pixel, ma trasformare la visualizzazione in un dominio funzionale, utilizzando funzioni matematiche.
In realtà l'intensità luminosa varia come $\frac{1}{R^2}$, se consideriamo R come la distanza.
Quindi avremo funzioni del tipo:

In questo caso ci vogliono più coefficienti per rappresentare la funzione..
Ad esempio $I = a + bi + ci^2 - di^3$ → 4 coefficienti.
In realtà, se vogliamo un'approssimazione più accurata possiamo utilizzare lo sviluppo in serie di Taylor, con cui possiamo rappresentare tutte le funzioni derivabili:

$$f(x) = f(x_0) + f^I(x_0)(x - x_0) + \frac{1}{2!}f^{II}(x_0)(x - x_0)^2 + \frac{1}{3!}f^{III}(x_0)(x - x_0)^3 \ldots\ldots$$

Anche se ci vogliono centinaia di coefficienti per rappresentare una funzione del genere, ciò rappresenta un notevole vantaggio rispetto ai 18000 pixel della macchina.

Generalizzando, possiamo dire che l'intensità di un pixel x può essere rappresentato con una successione di funzioni:

$$I(x) = a_0\varphi_0(x) + a_1\varphi_1(x) + a_2\varphi_2(x) + \cdots + a_{n-1}\varphi_{n-1}(x)$$

$a_0, a_1, a_2, \ldots, a_{n-1}$ = coefficienti

Nel caso di utilizzo di sviluppo in serie di Taylor:
($\varphi_0(x) = 1$, $\varphi_1(x) = x$, $\varphi_2(x) = x^2$,... $\varphi_k(x)=x^k$)

Conosciamo x (posizione del pixel), { φ_0, φ_1, φ_2, ..., φ_{n-1} } sono funzioni scelte arbitrariamente (possono essere polinomi, funzioni trigonometriche, etc).
L'intensità dei 3 colori è conosciuta, non conosciamo i coefficienti.
Ci è garantita però la linearità dei coefficienti, che ci evita equazioni del genere: $I(x) = (a_j + a_k)^2\varphi_1(x)$.
Conoscendo l'intensità luminosa per ogni pixel della riga, per trovare i coefficienti possiamo risolvere il seguente sistema lineare:

- $I(x_o) = a_0\varphi_0(x_o) + a_1\varphi_1(x_o) + a_2\varphi_2(x_o) + \cdots + a_{n-1}\varphi_{n-1}(x_o)$
- $I(x_1) = a_0\varphi_0(x_1) + a_1\varphi_1(x_1) + a_2\varphi_2(x_1) + \cdots + a_{n-1}\varphi_{n-1}(x_1)$
  .
  .
  .
- $I(x_{n-1}) = a_0\varphi_0(x_{n-1}) + a_1\varphi_1(x_{n-1}) + a_2\varphi_2(x_{n-1}) + \cdots + a_{n-1}\varphi_{n-1}(x_{n-1})$

La matrice che rappresenta il sistema ha sicuramente rango pieno (determinante diverso da 0) quindi è risolvibile, questo ci è garantito perché i pixel sono tutti diversi e le funzioni $\varphi_i(x)$ sono tutte linearmente indipendenti tra loro, cioè nessuna è esprimibile come funzione di un'altra.
Utilizzando il metodo di gauss, il costo computazionale per risolvere un sistema lineare è O(N^3), N = numero equazioni.
Nel nostro caso avremo O($(18\ x10^3)^3$), cioè 5832000000000 operazioni, nessun calcolatore è in grado di effettuare 5832000000000 operazioni in un tempo accettabile, per cui anche questa strategia è da eliminare.

11.3 - Sistemi Ortogonali

Per ridurre il costo computazionale si utilizzano insiemi di funzioni chiamate Sistemi Ortogonali.
Un Sistema Ortogonale è un sistema con la seguente proprietà:

$$\int f_i(x)f_j(x)dx \qquad \begin{cases} = 0 \rightarrow i \neq j \\ \neq 0 \rightarrow i = j \end{cases} \qquad x\epsilon[0,1]$$

I polinomi non sono un Sistema Ortogonale, infatti:

$$\int x^2\, x^3 \neq 0 \ \ in\ [0,1]$$

Le funzioni trigonometriche rappresentano un Sistema Ortogonale, infatti:
$f_0(x) = \cos(2\pi 0x) = 1$
$f_1(x) = \cos(2\pi x)$
$f_2(x) = \cos(2\pi 2x) = \cos(4\pi x)$
$f_3(x) = \cos(2\pi 3x) = \cos(6\pi x)$
.....
$f_k(x) = \cos(2k\Pi x)$

Segue che:

$$\int_0^1 \cos(2k\pi x)\cos(2j\pi x)dx = \int_0^1 \sin(2k\pi x)\cos(2j\pi x)dx = \int_0^1 \cos(2k\pi x)\sin(2j\pi x)dx$$
$$= \int_0^1 \sin(2k\pi x)\sin(2j\pi x)dx = 0$$

11.4 - Funzione Trigonometrica

Definiamo la funzione $T_n(x)$ come:

$$T_n(x) = a_0 \cos(2\pi 0x) + a_1 \cos(2\pi x) + a_2 \cos(4\pi x) + a_3 \cos(6\pi x) + \dots . a_n \cos(2n\pi x) + \\ + b_0 \sin(2\pi 0x) + b_1 \sin(2\pi x) + b_2 \sin(4\pi x) + b_3 \sin(6\pi x) + \cdots b_n \sin(2n\pi x)$$

$T_n(x)$ è un funzione periodica, di periodo 1, infatti:

$T_n(0) = a_0 + a_1 + a_2 + \dots . a_n$
$T_n(1) = a_0 + a_1 + a_2 + \dots . a_n$

Teorema di Parceval: La funzione $T_n(x)$ è una funzione periodica, di periodo 1.

Dimostrazione

$$T_n(x+1) = a_0 \cos(2\pi 0(x+1)) + a_1 \cos(2\pi(x+1)) + a_2 \cos(4\pi(x+1)) \\ + \dots . a_n \cos(2n\pi(x+1)) + b_0 \sin(2\pi 0(x+1)) + b_1 \sin(2\pi(x+1)) \\ + b_2 \sin(4\pi(x+1)) + \cdots b_n \sin(2n\pi(x+1))$$

Moltiplicando, abbiamo che:

$$T_n(x+1) = a_0 + a_1 \cos(2\pi x + 2\pi) + a_2 \cos(4\pi x + 4\pi) + \dots . a_n \cos(2n\pi x + 2n\pi) + b_0 \\ + b_1 \sin(2\pi x + 2\pi) + b_2 \sin(4\pi x + 4\pi) + \cdots b_n \sin(2n\pi x + 2n\pi)$$

Ricordando che:
$\cos(2\pi x + 2\pi) = \cos(2\pi x)$
$\cos(4\pi x + 4\pi) = \cos(4\pi x)$
$\sin(2\pi x + 2\pi) = \sin(2\pi x)$
$\sin(4\pi x + 4\pi) = \sin(4\pi x)$
E così via, possiamo scrivere $T_n(x+1)$ come:

$$T_n(x+1) = a_0 + a_1 \cos(2\pi x) + a_2 \cos(4\pi x) + a_3 \cos(6\pi x) + \dots . a_n \cos(2n\pi x) + b_0 \\ + b_1 \sin(2\pi x) + b_2 \sin(4\pi x) + b_3 \sin(6\pi x) + \cdots b_n \sin(2n\pi x) = T_n(x)$$

Il periodo della funzione è 1, ciò significa che l'immagine si ripete sia a destra che a sinistra dell'intervallo di riferimento [0,1].
La conseguenza è che approssimiamo i pixel più a destra della matrice che rappresenta l'immagine uguali ai pixel più a sinistra, ciò chiaramente non è quasi mai vero, per cui introduciamo un piccolo errore.
Il teorema di Parceval rende possibile la compressione di un segnale, per ogni componente del segnale si calcola la sua energia come l'integrale della funzione elevata al quadrato, ovvero:

$$\int I^2(x)dx = \sum a^2 k + b^2 k$$

Nel nostro caso il concetto di energia rappresenta la quantità di radiazione luminosa che impatta il pixel.
Possiamo calcolare indifferentemente l'integrale, oppure il valore nel dominio dei coefficienti (dominio di Fourier).
Questo teorema ci fa notare che ogni pixel è importante, e non va scartato, per questo motivo il metodo Accetta non va assolutamente utilizzato.

11.5 - Trasformata Di Fourier

$$T_N(x_k) = \sum_{j=0}^{N} (a_j \cos(2j\pi x_k) + b_j \sin(2j\pi x_k)$$

Questo polinomio gode delle seguenti proprietà:

-Periodica di periodo 1 → $T_n(x+1) = T_n(x)$

-Energia totale finita ed esprimibile come sommatoria di coefficienti → $\sum a^2k + b^2\,k$

-Proprietà di ortogonalità →

$\int_0^1 \cos(2k\pi x)\cos(2j\pi x)dx = \int_0^1 \sin(2k\pi x)\sin(2j\pi x)dx \begin{cases} = 0 \to k \neq j \\ \neq 0 \to k = j \end{cases}$

Questo è un polinomio compatto di grado N, e per ogni pixel j, noti i coefficienti a_j e b_j, ci permette di esprimere la sintesi di Fourier, il nostro obbiettivo è quindi quello di calcolare a_j e b_j.
I pixel sono tutti equidistanti e considerato l'intervallo [0,1]:
$x_k = \frac{k}{2N}$ $\quad 0 \leq k \leq 2N$
Il valore del polinomio nel punto finale è uguale al valore del polinomio nel punto iniziale per la periodicità della funzione, calcoliamo quindi il numero di equazioni:

$T_n(x_0) = f_0 = \sum_{j=0}^{N}(a_j \cos\frac{2j\pi}{2N}0 + b_j \sin\frac{2j\pi}{2N}0)$

$f_1 = \sum_{j=0}^{N}(a_j \cos\frac{2j\pi}{2N}1 + b_j \sin\frac{2j\pi}{2N}1)$

……

$f_{2N-1} = \sum_{j=0}^{N}(a_j \cos\frac{2j\pi(2N-1)}{2N} + b_j \sin\frac{2j\pi(2N-1)}{2N})$

Quindi il numero di equazioni è 2N, il numero degli a_j è N+1, il numero di b_j è N+1, quindi abbiamo un sistema di 2N equazioni in 2N+2 incognite, quindi il sistema è incompatibile, ma analizzando f_0 e f_N abbiamo che:

$$f_0 = b_0 \frac{\sin 2\pi xo}{2N} = b_0 \sin(0) = 0$$

$$f_N = b_N \frac{\sin 2N\pi x}{2N} = b_N \sin(\pi k) = 0$$

Di conseguenza possiamo approssimare i coefficienti b_0 e b_N uguali a 0.

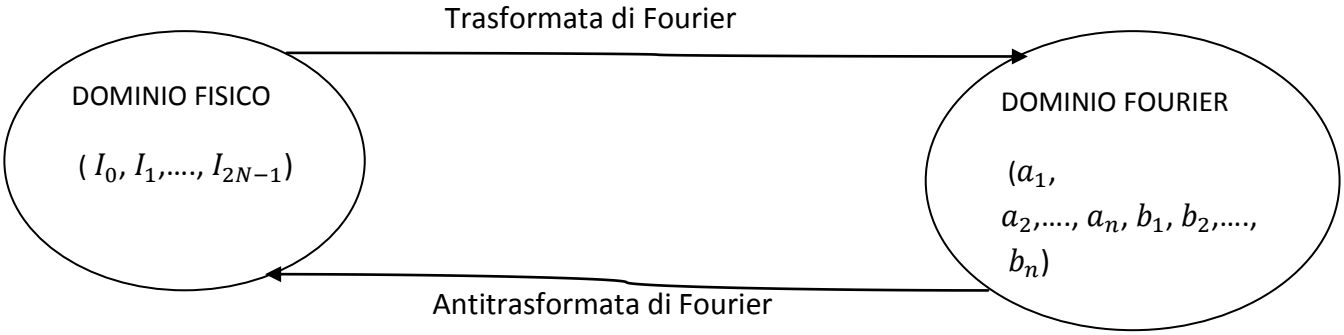
Il numero di incognite si riduce quindi a 2N, le 2N equazioni sono tutte diverse tra loro, il sistema è compatibile e risolvibile.

Le formule per trovare i coefficienti a_j e b_j sono:

$$a_j = \frac{1}{N}\sum_{k=0}^{2N-1} I_K \cos(2j\pi x_k)$$

$$b_j = \frac{1}{N}\sum_{k=0}^{2N-1} I_K \sin(2j\pi x_k)$$

Il numero di operazioni che occorre per calcolare a_j e b_j è dato dalla somma di 2N addizioni, 2N moltiplicazioni e 1, per una sola componente.
Moltiplicato per le 2N componenti si hanno $8N^2 + 2N$ operazioni, che nella notazione asintotica vuol dire che il metodo ha una complessità pari a O(N^2).
Dall'immagine rappresentata da una funzione, conosciuta nei punti: I_0, I_1,...., I_{2N-1}, si passa nel dominio di Fourier con i coefficienti: a_1, a_2,...., a_n, b_1, b_2,...., b_n.
Per visualizzare l'immagine bisogna ritornare nel dominio fisico.



Nel dominio di Fourier possiamo scartare dei coefficienti, poiché l'informazione che riportano è marginale, al contrario nel dominio fisico nessun punto può essere scartato, altrimenti si ridurrebbe la qualità dell'immagine.

11.5.1 - Richiamo ai numeri complessi

Un numero complesso è caratterizzato da una parte reale e da una parte immaginaria: $a + ib$
$i = \sqrt{-1}$ = unità immaginaria
$i^2 = 1$
$\frac{1}{i} = -i$

La somma di 2 numeri complessi viene definita come:
$(a + ib) + (c + id) = a + ib + c + id = (a + c) + i(b + d)$

La moltiplicazione di 2 numeri complessi viene definita come:
$(a + ib)\,(c + id) = ac + iad + ibc + i^2bd = ac + iad + ibc - bd = (ac - bd) + i(ad + bc)$

L'esponenziale viene definito come:
$e^{ix} = \cos x + i \sin x$
$e^{-ix} = \cos(-x) + i \sin(-x) = \cos x - i \sin x$

Formule di Eulero:

$e^{ix} + e^{-ix} = 2\ \cos x$ $\qquad\qquad$ $\cos x = \frac{e^{ix}+e^{-ix}}{2}$

$e^{ix} - e^{-ix} = 2i \sin x$ $\qquad\qquad$ $\sin x = \frac{e^{ix}+e^{-ix}}{2i}$

11.5.2 - Forma compatta della trasformata

Adesso, conoscendo i numeri complessi e le formule di Eulero, possiamo riscrivere la formula per la trasformata di Fourier in maniera più compatta.

$$T_n(x) = \sum_{j=0}^{N} (a_j \cos(2j\pi x) + b_j \sin(2j\pi x)$$

Sostituendo le formule di Eulero otteniamo:

$$T_n(x) = \sum_{j=0}^{N} (a_j \frac{e^{i2j\pi x} + e^{-i2j\pi x}}{2} + b_j \frac{e^{i2j\pi x} - e^{-i2j\pi x}}{2i})$$

Raccogliendo:

$$T_n(x) = \sum_{j=0}^{N} e^{i2j\pi x} \left(\frac{a_j}{2} + \frac{b_j}{2i}\right) + e^{-i2j\pi x} \left(\frac{a_j}{2} - \frac{b_j}{2i}\right)$$

Razionalizzando:

$$T_n(x) = \sum_{j=0}^{N} e^{i2j\pi x} \left(\frac{a_j}{2} - \frac{ib_j}{2}\right) + e^{-i2j\pi x} \left(\frac{a_j}{2} + \frac{ib_j}{2}\right)$$

Cioè:

$$T_n(x) = \sum_{j=0}^{N} e^{i2j\pi x} \left(\frac{a_j - ib_j}{2}\right) + e^{-i2j\pi x} \left(\frac{a_j + ib_j}{2}\right)$$

Adesso possiamo ridurre la sommatoria ad un unico termine, operando una modifica sugli indici, infatti se facciamo partire la sommatoria da –N, possiamo ottenere il secondo termine dal primo, quindi:

$$T_n(x) = \sum_{-N}^{N} c_j e^{i2j\pi x}$$

$$c_j = \begin{cases} \frac{a_j - ib_j}{2} & 0 < j < N \\ \frac{a_j + ib_j}{2} & -N < J < O \end{cases}$$

$$c_j = \frac{a_N}{2} \quad \text{per } j = N \text{ o } j = -N$$

$$c_j = a_0 \quad per\ j = 0$$

Possiamo quindi definire, la trasformata di Fourier e l'anti trasformata di Fourier.

Trasformata di Fourier $c_j = \frac{1}{2N}\sum_{k=0}^{2N-1} I_K e^{-i2j\pi x}$

Antitrasformata di Fourier $I_k = \sum_{j=-N}^{N} c_j e^{i2j\pi x}$

Il processo per la visualizzazione di un immagine in formato JPEG è il seguente:

Immagine → Trasformata → Elaborazione → Antitrasformata → JPEG

L'antitrasformata di Fourier di un vettore x è uguale al complesso coniugato della trasformata di Fourier del complesso coniugato del vettore x.
Se chiamiamo $F(x)$ la trasformata di Fourier per il vettore x, e con $IF(x)$ l'antitrasformata possiamo enunciare e dimostrare la seguente proprietà:

$$IF(x) = F(x^*)^*.$$

Dimostrazione

$IF(x) = \sum(cos + isin)(a_k + ib_k)$ → antitrasformata di x

$F(x^*) = \sum(cos - isin)(a_k - ib_k)$ → trasformata del complesso coniugato di x
$F(x^*) = \sum(a_k cos - ib_k cos - ia_k sin - b_k \sin)$
$F(x^*) = \sum(a_k cos - b_k \sin - i(b_k cos + a_k \sin)$

$F(x^*)^* = \sum(a_k cos - b_k \sin) + i(b_k cos + a_k \sin)$ →complesso coniugato della trasformata del complesso coniugato di x

$IF(x) = \sum(a_k cos + ib_k cos + ia_k sin - b_k \sin)$
$IF(x) = \sum(a_k cos - b_k \sin) + i(b_k cos + a_k \sin) = F(x^*)^*$

L'obbiettivo rimane sempre calcolare c_j, ma è possibile lavorare su N punti alla volta anziché 2N:

$$C_J = \frac{1}{N}\sum_{k=0}^{\frac{N}{2}-1} f_k e^{\frac{-i2j\pi k}{N}} + f_{(k+\frac{N}{2})} e^{\frac{-i2j\pi(k+\frac{N}{2})}{N}}$$

Poniamo $w = e^{\frac{-i2\pi}{N}}$

$$C_J = \frac{1}{N}\sum_{k=0}^{\frac{N}{2}-1} f_k w^{jk} + f_{(k+\frac{N}{2})} w^{jk} w^{j\frac{N}{2}}$$

Ma $w^{j\frac{N}{2}}$ può essere scritto come:

$$w^{j\frac{N}{2}} = \left(e^{\frac{-i2\pi}{N}}\right)^{j\frac{N}{2}} = e^{\frac{-i2\pi j\frac{N}{2}}{N}} = e^{\frac{-i2\pi J}{2}} = e^{-i\pi j}$$

Ricordiamo che $e^{-ix} = \cos x + i \sin x$, quindi:

$$e^{-i\pi j} = \cos(-\pi j) + i \sin(-\pi j) = \cos(\pi j) - i \sin(\pi j) = \cos(\pi j)$$

Ma sappiamo che:

$$\cos(\pi j) = \begin{cases} 1 & per\ j\ pari \\ -1 & per\ j\ dispari \end{cases}$$

Quindi possiamo riscrivere C_J come:

$$C_J = \frac{1}{N}\sum_{k=0}^{\frac{N}{2}-1} f_k w^{jk} + f_{\left(k+\frac{N}{2}\right)} w^{jk}(-1)^j$$

A questo punto abbiamo 2 diverse componenti, per j pari e per j dispari

$$\begin{cases} c_{2m1} = \frac{1}{N}\sum_{k=0}^{\frac{N}{2}-1}(f_k + f_{\left(k+\frac{N}{2}\right)})w^{2m1k} \\ c_{2m1+1} = \frac{1}{N}\sum_{k=0}^{\frac{N}{2}-1}(f_k - f_{\left(k+\frac{N}{2}\right)})w^{(2m1+1)k} \end{cases}$$

Poniamo $\frac{1}{N} = \frac{1}{2}\frac{1}{\frac{N}{2}}$

$$\begin{cases} c_{2m1} = \frac{1}{2}\left[\frac{1}{\frac{N}{2}}\sum_{k=0}^{\frac{N}{2}-1} F_k^{(i)} w1^{m1k}\right] \\ c_{2m1+\frac{N}{2}} = \frac{1}{2}\left[\frac{1}{\frac{N}{2}}\sum_{k=0}^{\frac{N}{2}-1} F_{k+\frac{N}{2}}^{(i)} w1^{m1k}\right] \end{cases} \qquad \text{con m1=0,…., } \frac{N}{2}$$

In sostanza avremo un risparmio di operazioni.

$$\begin{cases} f_0 \\ f_1 \\ \dots. \\ f_N \end{cases} \text{ N termini } \rightarrow \begin{cases} {f_0}^I = f_0 + f_{\frac{N}{2}+1} \\ {f_1}^I = f_1 + f_{\frac{N}{2}+2} \\ \dots. \\ {f_{\frac{N}{2}}}^I = f_{\frac{N}{2}} + f_N \end{cases} \quad \frac{N}{2} \text{ termini}$$

La formula di partenza aveva una complessità asintotica pari a $O(N^2)$, utilizzando questa rappresentazione possiamo abbassare questo limite.
Infatti, osservando la struttura delle somme, possiamo suddividerla in:

N somme → N termini

$\frac{N}{2} + \frac{N}{2}$ somme → $\frac{N}{2}$ termini

$\frac{N}{4} + \frac{N}{4} + \frac{N}{4} + \frac{N}{4}$ somme → $\frac{N}{4}$ termini

.
.
.

$\frac{N}{2^{N-1}} + \frac{N}{2^{N-1}}$ termini → $\frac{N}{N} = 1$termine

In questa maniera facciamo $\log N$ passaggi, un passaggio è dato dalla somma di $\frac{N}{2}$ termini + N volte la somma di $\frac{N}{2}$ termini, quindi: $\frac{N}{2} + \frac{N}{2}N = \frac{3}{2}N$.
Il costo computazionale quindi è di $O(\frac{3}{2}N \log N)$

Ad ogni nuovo passaggio perdo il concetto di pari/dispari, però possiamo sfruttare le caratteristiche del sistema binario, ed effettuare una riflessione a 180° dei bit, ad esempio:

$$\begin{pmatrix} 000 \\ 001 \\ 010 \\ 011 \\ 100 \\ 101 \\ 110 \\ 111 \end{pmatrix} \rightarrow \begin{pmatrix} 000 \\ 100 \\ 010 \\ 110 \\ 001 \\ 101 \\ 011 \\ 111 \end{pmatrix}$$

In questo caso l'ordine dei termini è $(c_0, c_4, c_2, c_6, c_1, c_5, c_3, c_7)$

La trasformata si presta ad essere implementata con circuiti integrati dedicati.
Nei processori le istruzioni per la trasformata sono direttamente implementate in macroistruzioni all'interno del processore.
La trasformata viene applicata per ogni riga e per ogni colonna, non importa l'ordine, ottenendo una nuova matrice con i coefficienti calcolati.

11.5.3 - Trasformata inversa di Fourier

Per effettuare la trasformata inversa di Fourier è possibile adottare 2 strategie:

- **Algoritmo ad hoc**

  $F(x) = \sum(cos + isin)(a_k + ib_k) = \sum(a_k cos + ib_k cos + ia_k sin - b_k \sin)$
  Bisogna, quindi, progettare un algoritmo che risolva questa serie.

- **Complesso coniugato della trasformata diretta**

  Possiamo descriverla mediante pochi passi:
  - Si utilizza la stessa parte reale, ma si cambia il segno alla parte immaginaria.
  - Applichiamo la trasformata di Fourier al nuovo numero ottenuto.
  - Riapplichiamo il punto 1, cioè al risultato ottenuto cambiamo il segno alla parte immaginaria.

11.5.4 - Trasformazione per potenze diverse da 2

La trasformata si può fare per dati che sono potenza di qualsiasi numero, invece di ricondurre i termini a 2 a 2, possiamo ricondurli a 3 a 3 (potenze di 3), a 4 a 4 (potenze di 4), e così via, ottenendo comunque una complessità logaritmica.
I circuiti dedicati implementano solo trasformate per potenze di 2, ma il JPEG utilizza un trucco, cioè divide la matrice in blocchi 8x8, trasformando ogni blocco.
Esempio: una matrice di 1920x1080 diventa una matrice composta da 240x135 blocchi.
Questo causa una discontinuità che, in caso di alta compressione, è ben visibile.

11.5.5 - Problema della periodicità

Come abbiamo già visto, la periodicità della funzione causa problemi ai bordi di ogni blocco.
Per risolvere il problema, riflettiamo la funzione F, quindi F diventerà pari.
Essendo la F pari, la possiamo scrivere come somma di soli coseni (funzione pari)

$$F(x) = \sum_{j=0}^{N}(a_j \cos(2j\pi x_k))$$
$$e^{ix} = \cos x$$

Con questo metodo non ho problemi di condizioni al contorno.
La trasformata coseno sarà completamente reale.
La funzione è continua all'esterno, ma la sua derivata non lo è.

11.5.6 - Problema di memorizzazione

Riflettiamo sulla memorizzazione dei coefficienti:
1 PIXEL → 0 – 255 RGB (intero) → 1 byte
1 coefficiente di Fourier → numero decimale → 4 byte
Inoltre nel secondo caso bisogna memorizzare anche la posizione (una coppia di interi) per ogni coefficiente.
Rischiamo di perdere più di $\frac{1}{4}$ del risparmio.
Per risolvere il problema possiamo trattare dei simboli che rappresentano più coefficienti.
Per memorizzare questa simbologia serve più spazio dei 0-255 (1 byte) del RGB.
Ma con un algoritmo di codifica possiamo riservare un numero diverso di bit per ogni simbolo (pochi bit per i simboli più frequenti, molti bit per i simboli meno frequenti, anche più di 8 bit)

11.6 – Funzione Wavelet (Jpeg 2000)

Il metodo che utilizza funzioni Wavelet è ancora più efficiente della trasformata di Fourier.
Utilizza funzioni frattali molto complesse, ma che possono essere calcolate numericamente in ogni punto.
Questo metodo ha complessità asintotica media pari ad O(N) ed è migliore della trasformata di Fourier nel 90% dei casi.

Capitolo 12 - Introduzione alla simulazione

12.1 - Automi Cellulari

Gli automi cellulari sono una soluzione intermedia tra i sistemi basati su agente e quelli

su variabili macroscopiche.

12.1.1 – Nascita degli automi cellulari

Il gioco “Life”

Negli anni ‘70 sulla rivista “Le Scienze” (versione italiana de “Scientific American”) Martin Gardner pubblicò un gioco chiama “Life”. Il gioco era composto da una scacchiera che ospitava dei vermi, i quali agivano in base a regole precise.

Questo gioco utilizzava una rappresentazione discreta invece che continua, creando un nuovo modo per modellare della realtà.

Vediamo in cosa consiste un sistema basato su automi cellulari sfruttando come esempio il gioco “Life”.

Si tratta di una semplice scacchiera (*Lattice*) e di alcune ipotesi semplificative:

- all’interno della scacchiera vi sono degli attori;
- un attore occupa una ed una sola posizione della scacchiera;
- tutti gli attori obbediscono a regole determinate.

	X			
	X	X		
		X		

L’obiettivo della nostra analisi è osservare come gli attori si evolvono nel tempo. Nel gioco Life l’unica cosa che un attore può fare è nascere o morire (da cui il nome “Life”): le regole che seguiranno determineranno quindi la sopravvivenza degli attori. Le regole saranno esclusivamente locali, ovvero il comportamento di un attore sarà basato esclusivamente sulle celle che gli stanno intorno (nel gioco Life, un intorno saranno le 8 celle adiacenti ad un attore).

Per ogni casella nella scacchiera:

se vi è un attore:

- **sopravvive** se sono presenti 2 o 3 attori nell'intorno
- **muore** negli altri casi

se non vi è nulla:

- **nascita** di un attore se sono presenti 3 attori nell'intorno
- rimane **vuota** negli altri casi

Le regole vengono sempre applicate alla scacchiera attuale per produrre la successiva (non tengo dinamicamente conto dell'evoluzione in corso).

Andamento di una partita

Scacchiera iniziale.

	1			
	2		4	
		3		

Applicando le regole sopra elencate a questa scacchiera iniziale: l'attore 1 soltanto un attore nel suo intorno (muore), l'attore 2 ha due attori (sopravvive), 3 ha due attori (sopravvive), 4 ha un attore (muore); mentre la casella a sinistra di 1 è vuota ed ha tre attori nel suo intorno, quindi nascerà un nuovo attore.

		5		
	2			
		3		

Applicando nuovamente le regole: l'attore 5 ha soltanto un attore nel suo intorno (muore), l'attore 2 ha due attori (sopravvive), 3 ha un solo attore (muore); mentre la casella a sinistra di 2 è vuota ed ha 3 attori nel suo intorno, quindi nascerà un nuovo attore.

	2	6		

Applicando nuovamente le regole: l'attore 2 ha soltanto un attore nel suo intorno (muore), l'attore 6 ha un solo attore (muore); mentre non c'è nessuna casella vuota con tre attori nel suo intorno, quindi non nasceranno nuovi attori.

Il gioco è terminato con tutti gli attori morti.

Questo gioco si chiama Life proprio perché cerca di modellare la sopravvivenza degli esseri viventi: se ad esempio un uomo è circondato da 2 o 3 altri individui, riesce a sopravvivere poiché in qualche modo gli altri individui lo aiutano a difendersi, sfamarsi, etc… Se ci sono troppi individui, abbiamo una situazione di sovraffollamento che non consente la sopravvivenza dell'individuo, mentre se ce ne sono troppo pochi, non sono sufficienti per consentirla. Allo stesso modo, se non ci sono individui, o sono in numero insufficiente, non può esserci riproduzione; ugualmente se ce ne sono troppi, il sovraffollamento ne ostacola la riproduzione. L'unica situazione favorevole alla nascita è la presenza di un numero intermedio di individui.

12.1.2 - Esempi

Scacchiera iniziale.

	1	2	
	3	4	

Applicando le regole: l'attore 1 ha tre attori nel suo intorno (sopravvive), l'attore 2 ha tre attori (sopravvive), 3 ha tre attori (sopravvive), 4 ha tre attori (sopravvive); mentre non ci sono celle vuote con 3 attori nell'intorno, quindi non nasceranno nuovi attori.

	1	2	
	3	4	

Il sistema è completamente stabile: evolve nel tempo sempre nello stesso modo.

Scacchiera iniziale.

		1		
	2		3	
		4		

Applicando le regole: l'attore 1 ha due attori nel suo intorno (sopravvive), l'attore 2 ha due attori (sopravvive), 3 ha due attori (sopravvive), 4 ha due attori (sopravvive); mentre non ci sono celle vuote con 3 attori nell'intorno, quindi non nasceranno nuovi attori.

		1		
	2		3	
		4		

Questo è un altro esempio di sistema stabile.

Scacchiera iniziale.

				1				
				2				
				3				
4	5	6				7	8	9
				10				
				11				
				12				

Applicando le regole: 1 ha un solo attore nel suo intorno (muore), 2 ha due attori (sopravvive), 3 ha un solo attore (muore), 4 ha un solo attore (muore), 5 ha due attori (sopravvive), 6 ha un solo attore (muore), simmetricamente per gli altri due gruppi di attori; mentre le caselle vuote subito a sinistra e destra di 2 hanno tre attori nell'intorno, quindi nasceranno due nuovi attori in quelle caselle, mentre le caselle vuote subito a sopra e sotto di 5 hanno tre attori nell'intorno, quindi nasceranno due nuovi attori in quelle caselle, per simmetria accadrà lo stesso negli altri due gruppi di attori.

			13	2	14			
	15						17	
	5						8	
	16						18	
			19	11	20			

Applicando le regole è facile verificare che otteniamo nuovamente la scacchiera iniziale.

				21				
				2				
				22				
23	5	24				7	8	9
				27				
				11				
				28				

Il sistema mostra un andamento ciclico.

Abbiamo andamenti molto vari tra loro: abbiamo soluzioni stabili, soluzioni cicliche (con periodi di varia lunghezza), soluzioni che portano all'estinzione, etc…

12.1.3 – Potenzialità e limitazioni

Le regole di "Life" sono note come 23/3, in altre parole un attore sopravvive se ci sono 2 o 3 attori nel suo intorno e nasce se che ne sono 3. È possibile utilizzare una grande varietà di regole, che possono essere anche molto complesse. Inoltre "Life" utilizza un sistema binario (esistenza / non esistenza), ma è possibile utilizzare sistemi ternari o più (ad esempio con colori diversi) e regole diversificate in base al contenuto specifico di una casella. Non solo, il sistema non deve essere necessariamente bidimensionale: può essere anche monodimensionale, tridimensionale, etc.. In questo modo la complessità aumenta di pari passo al numero di possibili stati che il sistema può assumere.

Gli automi cellulari sono quindi basati sulla riduzione di un sistema anche molto complesso a delle regole (più o meno numerose) semplici e valutabili localmente. È possibile così modellare una vasta varietà di realtà diverse.

Gli automi cellulari hanno però pesanti limitazioni:

- dominio spaziale molto rigido (un attore è vincolato su una singola casella e ogni casella può ospitare solo un attore), gli attori non sono quindi liberi di muoversi a piacere;
- sistema assolutamente deterministico (definita la condizione iniziale, è completamente definita anche l'evoluzione del sistema), mentre in un sistema reale l'evoluzione dipende anche da variabili casuali non prevedibili nelle condizioni iniziali.

Capitolo 13 – Minimo di una funzione

In breve:
Un problema comune a molte discipline consiste nell' individuare il minimo globale di una funzione. Questo compito, nel caso di funzioni non banali, quali funzioni complesse (difficilmente derivabili, a molte dimensioni etc…), non può essere affrontato analiticamente. Inoltre il costo computazionale per calcolare il valore della funzione in ogni punto - dunque trovare il minimo - può essere proibitivo. In questo capitolo vediamo come alcune procedure numeriche siano in grado di fornire una risposta approssimata ma soddisfacente.

13.1 – Metodi matematici

Rientrano in questa categoria quei metodi che fanno uso delle proprietà matematiche della funzione da minimizzare.

13.1.1 – Sezione aurea

Un primo metodo che illustriamo consiste nell'individuare un intervallo in cui la funzione contiene il minimo globale e restringerlo progressivamente.
Questo metodo non è indicato in presenza di minimi multipli perché tende ad individuare minimi locali, ma con gli opportuni accorgimenti possiamo ridurre al minimo questo rischio.
E' nota una procedura analoga per trovare le radici di una funzione, il metodo della bisezione. Dato un intervallo (a,b) tale che $f(a)$ e $f(b)$ hanno segno opposto, abbiamo almeno una radice nell'intervallo. Preso un punto x in (a,b) si ripete la procedura sul nuovo intervallo (x,b) o su (a,x), in base al segno di $f(x)$. La scelta ottimale della collocazione di x è a metà dell'intervallo (a,b), così da minimizzare la lunghezza dell'intervallo allo step successivo dell'algoritmo.
Applichiamo questa stessa filosofia al problema del minimo di funzione notando prima quanto segue: **un intervallo che contiene un minimo è caratterizzato non da 2 punti bensì da 3**, (a,b,c) tali che $f(b)<f(a),f(c)$.
Per ridurre l'intervallo sceglieremo un punto x in (a,c) e supponiamo che cada nella metà (b,c).
Avremo due possibilità. Se $f(x)<f(b)$ sceglieremo come nuova terna di punti (b,x,c).

Se $f(x) > f(b)$ sceglieremo (a, b, x) .

Ma come scegliere x ? Come nel caso della bisezione vogliamo che il costo computazionale sia minimizzato e al tempo stesso vogliamo evitare che l'algoritmo scelga due due intervalli quello che non contiene il minimo globale, cosa che è tanto più probabile quanto più l'intervallo scelto è piccolo. Dunque sceglierò x in modo tale che allo step successivo la lunghezza dei due possibili intervalli sia la medesima.

[disegno]

Consideriamo le distanze normalizzate sull'intervallo (a, c)

distanza ab: $w = (b-a)/(c-a)$

distanza bc: $1-w$

distanza bx: $z = (x-b)/(c-a)$ *(notare che* z *può essere negativo per* $x < b$ *)*

Le lunghezze normalizzate su (a, c) di (a, b, x) e (b, x, c) sono rispettivamente $w+z$ e $1-w$. Imponiamo dunque l'uguaglianza delle due lunghezze:

$$w + z = 1 - w$$
$$z = 1 - 2\mathrm{w}$$

Ipotizziamo che la distanza w sia stata scelta con lo stesso criterio con cui abbiamo scelto z , dunque che la distanza $(b-a)$ normalizzata su $(c-a)$ sia la stessa di $(x-b)$ normalizzata su $(c-b)$.

$$w = \frac{z}{1-w}$$
$$z = w - w^2$$

Mettendo a sistema le due espressioni di z otteniamo

$$1 - 2\mathrm{w} = w - w^2$$
$$w^2 - 3\mathrm{w} + 1 = 0$$
$$w = 3 \pm \frac{\sqrt{9-4}}{2}$$
$$w = 3 \pm \frac{\sqrt{5}}{2}$$

La cui soluzione accettabile è solo quella minore di 1 ($w < 1$ in quanto distanza normalizzata) e vale 0.38197 .

In pratica la collocazione ottimale dei punti prevede che il punto centrale disti 0.38197 da un estremo e 0.61803 dall'altro. Tale rapporto prende il nome

di sezione aurea.

13.1.2 – Metodo di Newton

Noto anche come metodo di Newton-Raphson è il metodo numerico più famoso per calcolare i minimi di funzione. Ha tale successo per la rapidità con cui converge ad una soluzione.

Il metodo di Newton ha radici nell'analisi e sfrutta la proprietà della derivata per la quale quand'è calcolata in un suo punto di minimo vale 0.

chiamo $g(x)=f\,'(x)$

Com'è ben noto, le radici di $g(x)$ corrispondono a punti di minimo o di massimo della funzione.

La funzione è $g(x)$ solitamente complessa da analizzare, così il metodo di Newton procede per approssimazione.

Dato un punto di innesco x_0 , scelto opportunamente, calcoliamo la tangente a $g(x)$ nel punto $(x_{0,}g(x_0))$. Ottenuta la radice della tangente $(x_{1,}o)$, verifico se $g(x_1)$ soddisfa le condizioni di terinazione dell'algoritmo, altrimenti ripeto la prodecura dal punto $(x_{1,}g(x_1))$.

Ricordiamo l'espressione della tangente ad un generico punto $(x_n, g(x_n))$

$$r(x)=g(x_n)+g\,'(x_n)(x-x_n)$$

Ed i passaggi per calcolare la radice di tale retta

$$0=g(x_n)+g\,'(x_n)(x-x_n)$$

$$\Rightarrow g\,'(x_n)(x-x_n)=-g(x_n)$$

$$\Rightarrow x-x_n=-\frac{g(x_n)}{g\,'(x_n)}$$

$$\Rightarrow x=x_n-\frac{g(x_n)}{g\,'(x_n)}$$

Scrittura dell'algoritmo:

- scelta di un punto d'innesco $p_n=(x_n, g(x_n))$
- fino a che in s non è presente una soluzione accettabile itera:
  - $\Rightarrow x_{n+1}=x_n-\frac{g(x_n)}{g\,'(x_n)}$
  - s ← $g(x_{n+1})$

Notiamo come il metodo di Newton determini una successione di punti

$$(x_0, g(x_0)) \Rightarrow x_1 = x_0 - \frac{g(x_0)}{g'(x_0)}$$

$$(x_1, g(x_1)) \Rightarrow x_2 = x_1 - \frac{g(x_1)}{g'(x_1)}$$

$$(x_2, g(x_2)) \Rightarrow x_3 = x_2 - \frac{g(x_2)}{g'(x_2)}$$

Ottengo a questo modo una successione di punti a partire dal punto di innesco x_0 :

$x_0 \Rightarrow x_1 = x_0 - g(x_0)/g'(x_0)$

$x_1 \Rightarrow x_2 = x_1 - g(x_1)/g'(x_1)$

$x_2 \Rightarrow x_3 = x_2 - g(x_2)/g'(x_2)$ etc...

Si dimostra che il metodo ha convergenza quadratica, dunque è molto veloce.

$$|x_{n+1} - \alpha| = c|x_n - \alpha|^2$$

Il metodo di Newton presenta comunque dei problemi

Convergenza.

Il metodo converge ad una soluzione solo se la funzione è convessa all'interno dell'intervallo. Per tentare di aggirare il problema possiamo scegliere un punto di innesco sicuro; ad esempio eseguendo una scansione discreta dell'intervallo e scegliendo il punto d'innesco in base a questa osservazione.

Inoltre Newton, poiché cerca punti per i quali si annulla la derivata, non distingue un minimo locale da uno globale.

13.2 - Metodi Euristici

Abbiamo visto come i metodi precedenti convergono ad una soluzione solo se determinati requisiti sono soddisfatti. Questi possono risultare stretti o insolvibili, come nel caso del calcolo della derivata prima nel metodo di Newton; inoltre questi metodi non distinguono i minimi locali dai minimi globali.

A questi problemi risponde un'altra categoria di procedure che si allontanano del tutto da una visione analitica della funzione per abbracciarne una del tutto numerica. Le strategie che vedremo prendono ispirazione da fenomeni naturali: il movimento di uno sciame, la formazione di un cristallo ed il meccanismo di selezione naturale. Nella loro diversità sono tutte caratterizzate da un qualche meccanismo per sfuggire al rischio di convergere verso un minimo locale.

13.2.1 – Particle Swarm Optimization

Il modello di ottimizzazione prende spunto dal comportamento delle particelle in uno spazio fisico. Ogni particella rappresenta un candidato ad essere il

minimo della funzione obiettivo, chiameremo così la funzione per la quale vogliamo trovare il minimo globale.
Di seguito viene riportato l'algoritmo utilizzato da tale metodo:

- Seleziona un'insieme di particelle di cardinalità S con i criteri più opportuni
- Per ogni particella i inizializza:
  - La posizione della particella: x_i
  - La velocità della particella: v_i
  - La migliore posizione raggiunta: p_i
- Individua tra le particelle la particella j per la quale $f(x_j) \leqslant f(x_i) \ \forall i = 1 \dots S$
- Inizializza il minimo globale con la posizione della particella j: $g \leftarrow x_j$
- Fino a che le condizioni di terminazione non sono raggiunte:
  - Per ogni particella $i = 1 \dots S$:
    - Scegli due numeri casuali: $r_p, r_g \in U(0,1)$
    - Calcola la velocità della particella: $v_i \leftarrow \omega v_i + \varphi_p r_p (p_i - x_i) + \varphi_g r_g (g - x_i)$ dove φ_p, φ_g sono parametri scelti in base all'esperienza dell'utente per ottimizzare il risultato della funzione
    - Aggiorna la posizione della particella: $x_i \leftarrow x_i + v_i$
    - Se é vero $f(x_i) \leqslant f(p_i)$:
      - Aggiorna la migliore posizione raggiunta: $p_i \leftarrow x_i$
      - Se $f(p_i) \leqslant f(g)$ Aggiorna la migliore posizione assoluta: $g \leftarrow p_i$
- Una volta terminato **g** contiene il minimo globale della funzione.

Notare che la velocità della particella viene aggiornata in base sia del valore minimo che ha raggiunto sia del valore minimo raggiunto dallo sciame. Questi fattori determinano la convergenza verso una soluzione mentre la casualità imposta da r_p ed r_g dà al sistema la polssibiltà di uscire da un minimo locale.

13.2.2 – Simulated Annealing

Il modello è ispirato alla formazione dei cristalli che, raffreddandosi lentamente a partire da altissime temperature, formano la propria struttura solida assecondando il principio di minima energia. Nel nostro algoritmo ogni punto s dello spazio di ricerca rappresenta uno stato un sistema fisico. In tale rappresentazione la funzione obiettivo E(s) è analoga all'energia interna del sistema.

Ad ogni passo della computazione l'algoritmo decide per ogni s se passare in uno stato s' o rimanere nello stato s basandosi su una funzione di probabilità.

Riportiamo di seguito una Versewione dell'algoritmo:

- Seleziona un'insieme di particelle di cardinalità S con i criteri più opportuni
- Per ogni particella i inizializza:
  - La posizione della particella: x_i
  - La tempertura iniziale: t
  - Un valore arbitrario di step size per le fluttauazioni di energia : Δ
  - Due funzioni di probabilità:
    - La prima che calcola l'energia di uno stato atomico, per

esempio: $\varepsilon(x_i)=e^{-\frac{Kx_i}{t}}$ dove K è una costante arbitraria.

- La seconda valuta se cambiare stato anche se l'energia ad esso associata è maggiore di quella attuale. Per esempio: $p(x_i,\tilde{}x_i)=\frac{1}{1+e^{f(\tilde{}x_i)-f(x_i)}}$

- Individua tra le particelle la particella j per la quale $f(x_j)\leqslant f(x_i)\ \forall i=1\ldots S$
- Inizializza il minimo globale con la posizione della particella j: $g\leftarrow x_j$
- Fino a che la temperatura t non ha raggiunto il valore di equilibrio:
  - Per ogni particella $i=1\ldots S$:
    - Calcola la possibile variazione di posizione della particella: $\tilde{}x_i\leftarrow x_i+\varepsilon(x_i)\Delta$
    - Se é vero $f(\tilde{}x_i)\leqslant f(x_i)$ o se randomicamente si rientra nella probabilità $p(x_i,\tilde{}x_i)$:
      - Aggiorna la posizione : $x_i\leftarrow\tilde{}x_i$
    - Se $f(x_i)\leqslant f(g)$ Aggiorna la migliore posizione assoluta: $g\leftarrow x_i$
  - Diminuisco la temperatura globale t e vario lo step size Δ a piacere
- Una volta terminato **g** contiene il minimo globale della funzione.

13.2.3 – Genetic Analysis

Il modello si basa sui due meccanismi di variazione del genoma nella riproduzione: ricombinazione e mutazione. A partire da un certo numero di individui (possibili soluzioni), si genera una nuova popolazione, e poi se ne variano le caratteristiche con gli analoghi matematici dei due metodi sopracitati.
Una versione dell'algoritmo per uno spazio discreto è il seguente :

- Seleziona un'insieme di indivudui di cardinalità S con i criteri più opportuni
- Per ogni individuo $i=1\ldots S$ inizializza:
  - Il genoma dell'individuo: x_i
  - Calcola il suo valore di adattabilità: $a_i\leftarrow f(x_i)$
- Fino a che le condizioni di terminazione non sono state raggiunte:
  - Per ogni individuo $i=1\ldots S$:
    - Se $f(x_i)\leqslant f(g)$ Aggiorna il minimo globale: $g\leftarrow x_i$
    - Assegna a ciascun individuo la probabilità di essere selezionato per la generazione successiva: $p_i\leftarrow x_i$
  - Genero una nuova popolazione di cardinalità S selezionando randomicamente gli individui in base alla loro probabilità p_i
  - Seleziono un insieme $S'\subseteq S\times S$ di coppie alle quali applicare la ricombinazione.
  - Seleziono un sottinsieme $S''\subseteq S$ di individuo ai quali applicare la mutazione
- Una volta terminato **g** contiene il minimo globale della funzione.

Le procedure di ricombinazione e di mutazione consistono rispettivamente in:

- Data una coppia di invidui selzionare casualmente un valore di offset sulla

stringa binaria che ne rappresenta il genoma e scambiare la parte di essa a sinistra dell'offset

- Dato un'individuo selezionare casualmente un bit ndella stringa binaria che ne definisce il genoma e sostituirlo con il suo complemento

Capitolo 14 - Curve e superfici di Bézier

14.1 - Introduzione alla Computer Graphics

La Computer Graphics costituisce una delle più diffuse applicazioni dell'informatica, è una disciplina che basandosi su metodi ed algoritmi di analisi numerica, insieme con particolari tecnologie e strumenti hardware, consente di visualizzare oggetti mediante il calcolatore.
Senza dubbio si può affermare che non esiste applicazione in cui la grafica non possa essere di valido aiuto o quanto meno di complemento.
La grafica trova ampie applicazioni anche in settori non tecnici, ad esempio nelle indagini demografiche nell'analisi statistica e nelle applicazioni economiche in genere.
A seconda del tipo di applicazione la Computer Graphics può dividersi in tre categorie principali

- Grafica per applicazioni economiche (Business Graphics)
- Animazione
- Progettazione

L'obiettivo della Business Graphics è quello di presentare determinati risultati con l'ausilio di grafici, rendendone più immediata la comprensione, utilizzando applicazioni molto semplici e risorse hardware non sofisticate.
Nel settore dell'animazione si cerca di sfruttare l'elevata velocità dei calcolatori attuali codificando programmi in grado di animare disegni con risoluzione elevata in modo tale da dare la sensazione del movimento continuo. I possibili campi applicativi sono la preparazione dei videogiochi o di pellicole cinematografiche e l'analisi cinematica di un fenomeno; Esistono programmi che, simulando gli spostamenti del corpo di un guidatore in caso d'incidente automobilistico, consentono di decidere quale debba essere la disposizione migliore all'interno della vettura.

Uno dei primi campi applicativi della Computer Graphics è stato quello della progettazione.
A seconda dei settori di applicazione essa si divide in quattro categorie

- Progettazione elettronica
- Progettazione architettonica
- Progettazione industriale
- Progettazione meccanica

Nel settore elettronico le principali applicazioni vanno dalla progettazione dei circuiti integrati a quella di intere schede, con produzione automatica dei disegni esecutivi e della lista dei componenti necessari alla produzione. I vantaggi vanno oltre il semplice risparmio di tempo associato all'esecuzione della parte grafica, infatti, man mano che il progetto procede, l'uso del computer consente di controllare le scelte del progettista in termini di compatibilità e di interfacciabilità tra i componenti usati e di operare una selezione tra i diversi componenti similari sulla base delle caratteristiche richieste.

Nella progettazione architettonica le principali applicazioni riguardano l'abitabilità degli ambienti, l'arredamento, l'analisi strutturale e il disegno di viste prospettiche.
Il campo applicativo dell'industrial design copre tutti i settori nei quali al progetto tecnico esecutivo di un determinato prodotto è necessario associare uno studio estetico approfondito del risultato che si desidera ottenere.

La progettazione meccanica è spesso identificata con il termine C.A.D. (Computer Aided Design), cioè la progettazione automatica nel suo complesso ed è, in realtà, il capostipite di tutte le altre applicazioni.
Sia il C.A.D. che il C.A.M. (Computer Aided Manufacturing), ovvero il controllo automatico delle macchine utensili, hanno portato al raggiungimento di un'elevata rapidità di calcolo ad elevata precisione numerica, abbassando tempi e costi di produzione.
Un sistema CAD può essere visto come uno strumento di progettazione assistita.
La caratteristica fondamentale di un sistema CAD è la capacità di esprimere come insiemi di numeri la forma di un oggetto e di operare su di essa utilizzando il calcolatore.
A questa rappresentazione numerica della forma si richiede che

- sia applicabile ad una gran varietà di forme geometriche
- permetta di operare su queste forme per mezzo di calcoli semplici
- sia indipendente dal sistema di coordinate
- sia facilmente utilizzabile anche da non matematici.

Un tipo di rappresentazione di curve e superfici che risponde a queste esigenze è quella parametrica. Dal punto di vista matematico, tale rappresentazione offre molti vantaggi rispetto alla forma analitica, quali

- una rappresentazione di tipo esplicito
- l'uso della la forma vettoriale che consente di esprimere la curva come combinazione lineare a coefficienti vettoriali di funzioni scalari
- indipendenza dal sistema di riferimento

- calcolo separato delle componenti e l'uso della stessa routine per il calcolo di ciascuna componente con riduzione della complessità dei programmi

Lo studio delle curve e delle superfici parametriche da un punto di vista computazionale ha costituito il nucleo base di una nuova disciplina, il C.A.G.D (Computer Aided Geometric Design), dove l'aggiunta dell'aggettivo Geometric limita l'attenzione agli aspetti matematici del CAD.

14.2 - Curve di Bézier

Le curve di Bézier sono curve parametriche polinomiali, tali forme parametriche non possono rappresentare forme semplici come il cerchio.
Della stessa specie sono anche le curve B-splines.
Introducendo coordinate omogenee queste curve si generalizzano in curve razionali di Bézier e **N**on-**U**niform **R**ational **B**-splines, in breve NURBS.
Le curve razionali sono più potenti delle curve originali dal momento che possono rappresentare forme come cerchi ed elissi.

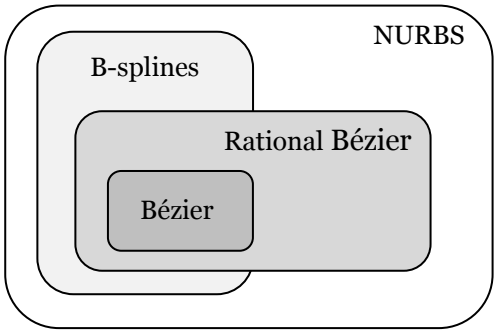


Fig. 1

In Fig. 1 è riportata la relazione tra questi quattro tipi di rappresentazione di curve.

14.3 - Storia

Le curve di Bézier furono scoperte contemporaneamente da Paul de Casteljau e Pierre E. Bézier. Furono pubblicizzate nel 1962 da Pierre E. Bézier, un ingegnere meccanico francese della fabbrica Renault, che si occupava della gestione delle macchine a controllo numerico, impiegate per il taglio delle lamiere delle carrozzerie per automobili.
Le curve furono realizzate da Paul de Casteljau nel 1959 tramite l'algoritmo di de Casteljau.

14.4 - Definizione

Dati $n+1$ punti P_i, denominati punti di controllo, chiamiamo legs i segmenti di linea P_i-P_{i+1} per $i = 0 \dots n-1$ e polilinea di controllo o poligono di controllo l'unione dei legs.
La curva di Bézier definita dai punti P_i è descritta dalla formula parametrica

$$C(u) = \sum_{i=0}^{n} B_{n,i}(u)\, P_i \quad u \in [0,1]$$

$$B_{n,i}(u) = \frac{n!}{i!\ (n-i)!} u^i\ (1-u)^{n-i} = \binom{n}{i} u^i (1-u)^{n-i} \quad (0 \le i \le n)$$

dove:
$B_{n,i}(u)$ sono i polinomi di Bernstein o funzioni di base di Bézier.
P_i sono i punti di controllo ($P_i \in R^d, d \ge 2$).

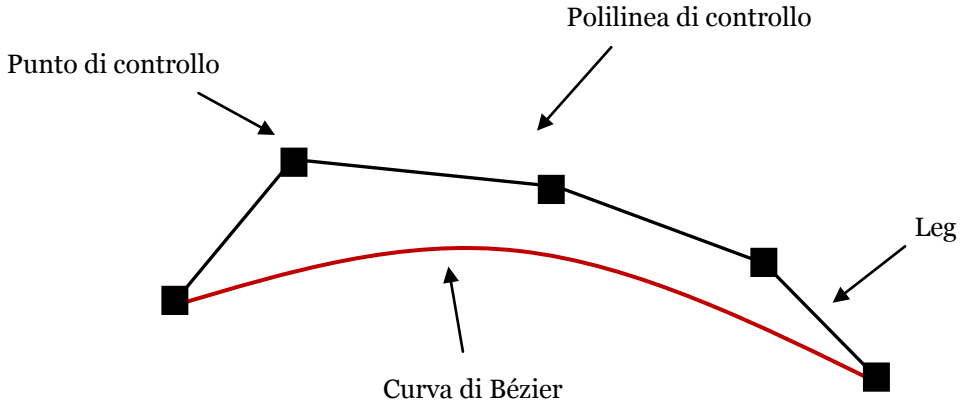


Fig. 2

Il punto che corrisponde ad u sulla curva è la media pesata di tutti i punti di controllo con i pesi pari ai coefficienti $B_{n,i}(u)$.
Il dominio di u è [0,1], quindi le funzioni di base sono non negative.

Nella formula che definisce i polinomi di Bernstein sia u che i, cosi come $1-u$ e $n-i$, possono essere entrambi zero poiché $0 \leq i \leq n$, si adotta dunque la convezione che $0^0 = 1$.

14.5 - Proprietà

Dipendenza grado-punti di controllo (proprietà negativa)

Il grado di una curva di Bézier definita in $n+1$ punti di controllo è n.
In ogni funzione di base l'esponente di u è $i + (n-i) = n$ pertanto il grado della curva è n.
La proprietà comporta un grande svantaggio poiché polinomi di grado elevato sono più difficili da gestire per motivi di instabilità numerica.

Interpolazione agli estremi

La curva di Bézier passa sempre attraverso il primo e l'ultimo punto del poligono di controllo.
La proprietà è molto utile per raccordare bracci di diverse curve.
Quindi

$$B_{n,i}(0) = \begin{cases} 0 & i \neq 0 \\ 1 & i = 0 \end{cases} \quad B_{n,i}(1) = \begin{cases} 0 & i \neq n \\ 1 & i = n \end{cases}$$

$$C(0) = P_0 \quad C(1) = P_n$$

Partizione dell'unità (curva ben definita)

La somma delle funzioni di base per un fissato u è sempre pari a 1.
La proprietà assicura che la curva è ben definita ovvero indipendente dalla scelta dell'origine delle coordinate e dipendente solo dalla scelta dei punti P_i.
Traslando i punti di un vettore $\bar{v}$ anche la curva risulterà traslata rispetto all'origine di una quantità $\bar{v}$.
Occorre dunque verificare

$$\sum_{i=0}^{n} B_{n,i}(u) = 1$$

ovvero

$$\sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i} = 1$$

ricordando che

$$(a+b)^n = \sum_{i=0}^{n} \binom{n}{i} a^i b^{n-i}$$

Sostituendo a = u e b = 1 − u, otteniamo

$$(u+(1-u))^n = \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i} = 1$$

Le funzioni di base sono quindi i coefficienti nell'espansione binomiale $1 = (u+(1-u))^n$.
Inoltre essendo le funzioni di base non negative concludiamo che il valore di qualunque funzione di base è compreso tra 0 e 1.
Da notare che essendo le funzioni di base comprese tra 0 e 1 e la loro somma pari a 1, tali funzioni possono essere considerate come pesi nel calcolo di una media pesata.
Per calcolare $C(u)$ si prende il peso $B_{n,i}(u)$ per ciascun punto di controllo P_i e si sommano insieme.

Hull convesso

La curva di Bézier definita da n+1 punti di controllo assegnati giace completamente nel dominio convesso (hull convesso) formato dai punti di controllo.
Il dominio convesso di un insieme di punti è il più piccolo insieme convesso che contiene tutti punti.
Un insieme di punti è definito convesso se per ogni coppia di punti (a,b) dell'insieme il segmento a-b che li congiunge è interamente contenuto nell'insieme.

La proprietà di Hull convesso è sempre verificata quando

$$B_{n,i}(u) \geq 0$$

Ma ricordando la definizione dei polinomi di Bernstein

$$B_{n,i}(u) = \binom{n}{i} u^i (1-u)^{n-i} \quad (0 \leq i \leq n)$$

La disuguaglianza è sempre verificata per

$$u \in [0,1]$$

La proprietà garantisce che la curva generata giacerà in una regione definita e calcolabile vicina al poligono di controllo e non fuoriuscirà da essa.

In Fig. 3 il dominio convesso dei 5 punti di controllo è mostrato tratteggiato e la curva corrispondente giace completamente in questa regione di piano.

In Fig. 4 la curva non rispetta la proprietà in quanto fuoriesce dal dominio convesso.

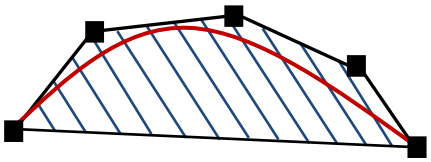


Fig. 3

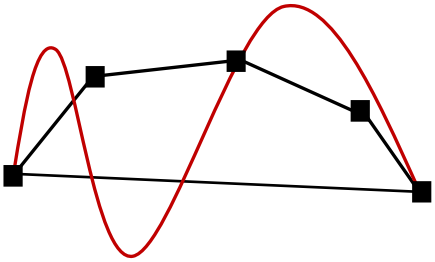


Fig. 4

Variazione di diminishing (riduzione delle oscillazioni)

Il numero di intersezioni di una qualsiasi retta con la curva è sempre minore o uguale al numero di intersezioni della retta con il poligono di controllo.
La proprietà è valida anche nello spazio con i piani.

La proprietà garantisce che la complessità della curva (TURNING e TWISTING) sarà non maggiore della complessità del poligono di controllo e che la curva non oscillerà troppo rispetto al poligono.

In Fig. 5 si può notare che la curva in rosso non rispetta la proprietà in quanto la retta interseca la curva un numero di volte maggiore rispetto al numero di intersezioni della stessa retta con il poligono di controllo.
Diversamente, la curva in blu rispetta la proprietà in quanto il numero di volte che la retta interseca la curva è pari al numero di volte che la stessa retta interseca il poligono di controllo.

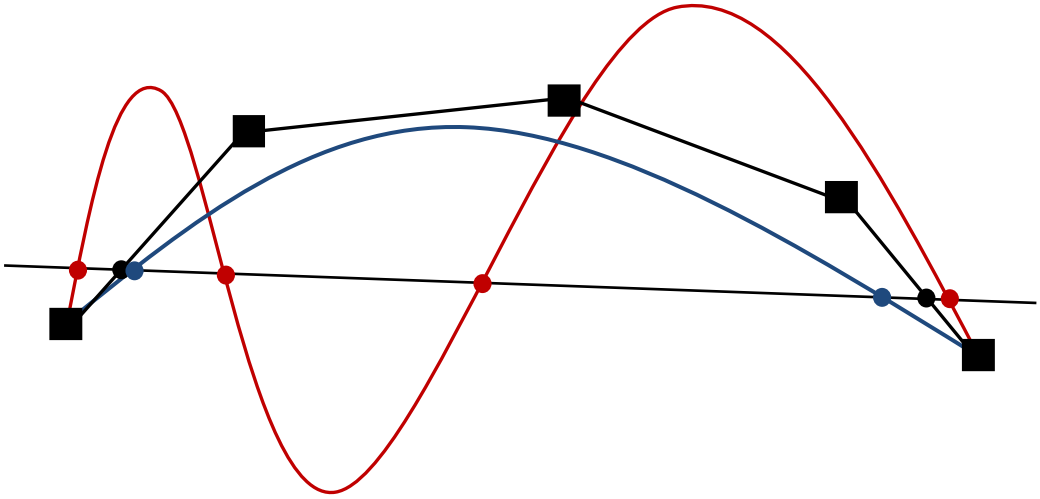


Fig. 5

Invariante affine

Quando occorre una trasformazione geometrica o affine (rotazione, traslazione ecc..) su una curva di Bézier si opera solo sui punti di controllo. Ottenuti i punti trasformati la curva di Bézier trasformata è definita dai nuovi punti. La trasformazione della curva risulta più semplice.
Non occorre dunque lavorare sui punti della curva.

Regolarità della curva (smoothness)

La curva di Bézier deve essere dolce e regolare e non deve presentare picchi o discontinuità in quanto la funzione $C(u)$ è derivabile più volte poiché somma di polinomi.

Simmetria della curva

Invertendo l'ordine dei punti del poligono di controllo non cambia la forma della curva poiché

$$B_{n,i}(u) = B_{n,n-i}(1-u)$$

infatti

$$B_{n,n-i}(1-u) = \frac{n!}{(n-i)!\,(n-(n-i))!}(1-u)^{n-i}(1-(1-u))^{n-(n-i)}$$

$$= \frac{n!}{i!\,(n-i)!}u^{i}(1-u)^{n-i} \;= B_{n,i}(u) \quad u \in [0,1]$$

quindi

$$C(P_n, \dots, P_0, u) = C(P_0, \dots, P_n, (1-u)) \quad u \in [0,1]$$

Conservazione di punti e rette

Se tutti i punti P_i del poligono di controllo collassano nel punto P_0 anche la curva coinciderà con quel punto.

$$P_i = P_0 \; \forall i = 1 \dots n$$

allora

$$C(u) = \sum_{i=0}^{n} B_{n,i}(u)\, P_i = P_0 \sum_{i=0}^{n} B_{n,i}(u)$$

ricordando la proprietà "partizione dell'unità"

$$\sum_{i=0}^{n} B_{n,i}(u) \; = 1$$

quindi

$$C(u) = \sum_{i=0}^{n} B_{n,i}(u)\, P_i = P_0 \sum_{i=0}^{n} B_{n,i}(u) \; = P_0 1 = P_0$$

Se tutti i punti sono allineati a formare una retta anche la curva seguirà l'andamento (Fig. 6). Infatti

$$\sum_{i=0}^{n} B_{n,i}(u) \; i = nu$$

Fig. 6

Controllo locale (proprietà negativa)

Cambiando la posizione di un punto di controllo la forma della curva di Bézier cambia in ogni punto (Fig. 7).
La proprietà verrà analizzata accuratamente nel paragrafo 14.7.

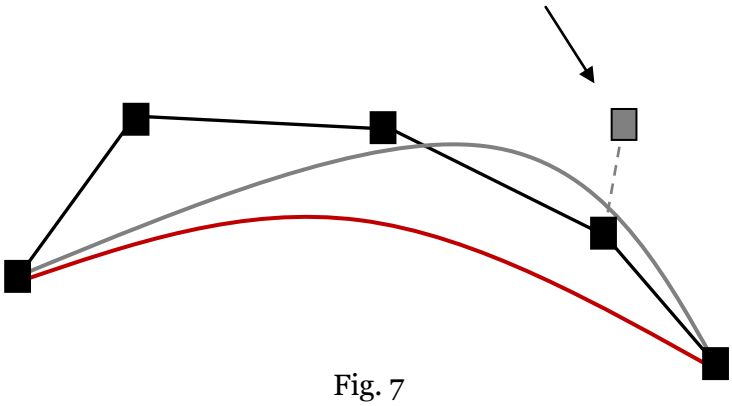


Fig. 7

14.6 - Dominio di u diverso da [0,1]

Talvolta il dominio di una curva di Bézier è $[a, b]$ invece che [0,1].
Si opera dunque un cambiamento di variabile ovvero si converte una u in $[a, b]$ in una nuova $\underline{u}$ in [0,1] e si usa la nuova $\underline{u}$ nelle funzioni di base.
La conversione può essere eseguita nel seguente modo

$$\underline{u} = \frac{u - a}{b - a}$$

Inserendo questa nuova $\underline{u}$ nelle funzioni di base $B_{n,i}(\underline{u})$ si produce la seguente formula

$$B_{n,i}(u) = \frac{n!}{i!\ (n-i)!} \left(\frac{u-a}{b-a}\right)^i \left(1 - \frac{u-a}{b-a}\right)^{n-i}$$

Queste nuove funzioni di base definiscono una curva di Bézier nel dominio $[a, b]$.

14.7 - Spostare i punti di controllo

Come visto nel paragrafo 14.5 con la proprietà “controllo locale”, variando la posizione di un punto di controllo la forma della curva di Bézier cambia in ogni punto.
Si supponga che il punto di controllo P_k venga spostato in una nuova posizione $P_k + \bar{v}$, dove $\bar{v}$ rappresenta il vettore spostamento che fornisce sia la direzione che la lunghezza dello spostamento (Fig. 8).

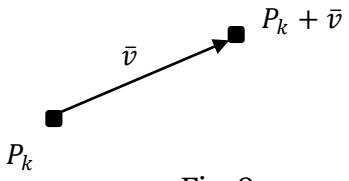


Fig. 8

La curva di Bézier originale è data da

$$C(u) = \sum_{i=0}^{n} B_{n,i}(u)\, P_i$$

La nuova curva di Bézier è definita dai punti di controllo $P_0, P_1, \dots, P_{k+\bar{v}}, \dots, P_n$ e la sua equazione $D(u)$ è definita come

$$\begin{aligned} D(u) &= \sum_{i=0}^{k-1} B_{n,i}(u)\, P_i + B_{n,k}(u)(P_k + \bar{v}) + \sum_{i=k+1}^{n} B_{n,i}(u)\, P_i \\ &= \sum_{i=0}^{k-1} B_{n,i}(u)\, P_i + B_{n,k}(u)P_k + B_{n,k}(u)\bar{v} + \sum_{i=k+1}^{n} B_{n,i}(u)\, P_i \\ &= \sum_{i=0}^{k-1} B_{n,i}(u)\, P_i + B_{n,k}(u)P_k + \sum_{i=k+1}^{n} B_{n,i}(u)\, P_i + B_{n,k}(u)\bar{v} \\ &= \sum_{i=0}^{n} B_{n,i}(u)\, P_i + B_{n,k}(u)\bar{v} = C(u) + B_{n,k}(u)\bar{v} \end{aligned}$$

Quindi il corrispondente punto di u sulla nuova curva è ottenuto traslando il punto u originale nella direzione di $\bar{v}$ ad una distanza $\left|B_{n,k}(u)v\right|$.
La variazione da $C(u)$ a $D(u)$ è massima quando $B_{n,k}(u)v$ raggiunge il massimo.
Inoltre $B_{n,k}(u)$ è diverso da 0 in $(0,1)$ quindi $B_{n,k}(u)\bar{v}$ non è un vettore nullo in $(0,1)$ e questo vuol dire che tutti i punti sulla curva originaria vengono spostati in nuove posizioni eccetto i due punti $C(0)$ e $C(1)$ poiché $B_{n,k}(0) = B_{n,k}(1) = 0$.

In Fig. 9 la curva di Bézier è definita mediante 9 punti di controllo (grado 8).
Il punto di controllo 3 viene spostato in una nuova direzione indicata dal vettore $\bar{v}$ e la curva rossa si modifica in quella grigia.
Su ognuna delle due curve vi è il punto corrispondente a $u = 0.5$. Il punto $C(0.5)$ si sposta nella stessa direzione di $\bar{v}$ verso $D(0.5)$.

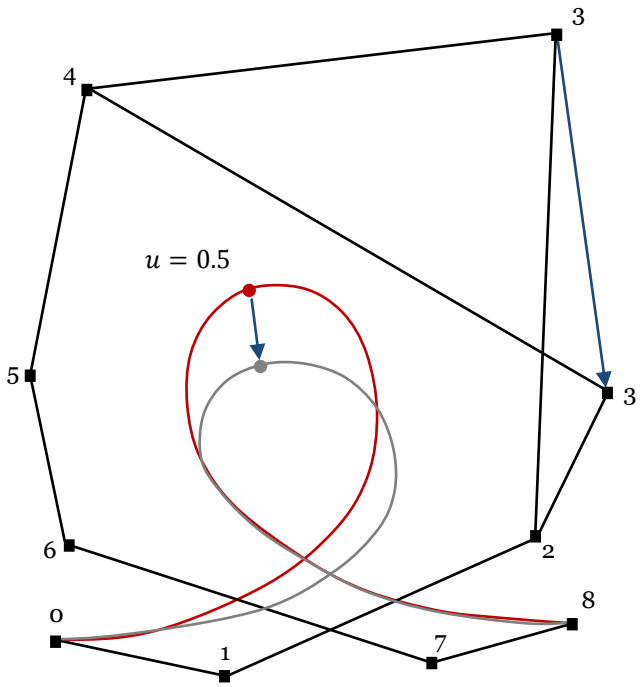


Fig. 9

La distanza tra $C(0.5)$ e $D(0.5)$ è la lunghezza del vettore $B_{8,3}(0.5)v$

$$B_{8,3}(0.5)v = \frac{8!}{3!\,(8-3)!}0.5^3(1-0.5)^{8-3}v = 0.22v$$

pertanto la distanza è circa il 22% della distanza tra il punto di controllo originale 3 e il nuovo punto di controllo 3.

14.8 - Tracciamento della curva

Un modo semplice per calcolare il valore di $C(u)$ e costruire una curva di Bézier è quello di assegnare un valore a u in ogni funzione di base, calcolare il prodotto di ciascuna funzione di base e del suo corrispondente punto di controllo e infine sommare i termini.
Pur essendo questa procedura corretta, tuttavia non è numericamente stabile, vale a dire potrebbe introdurre errori numerici durante le operazioni necessarie per calcolare i polinomi di Bernstein.

14.8.1 - Algoritmo di De Casteljau

L'algoritmo di De Casteljau permette di costruire la curva di Bézier associata al vettore di punti di controllo assegnato, lavorando su combinazioni lineari.
Il concetto fondamentale dell'algoritmo di de Casteljau è scegliere un punto C nel segmento di linea A-B tale che C divide il segmento in un rapporto di u: $1-u$, vale a dire il rapporto della distanza tra A e C e quella tra A e B è u (Fig. 10).

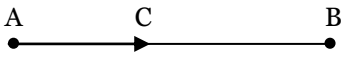


Fig. 10

Per determinare il punto C si considera il vettore da A a B, $B-A$.
Poiché u è un rapporto compreso tra 0 e 1, il punto C è situato nella posizione

$$u(B-A)$$

Considerando la posizione di A, il punto C sarà

$$A+u(B-A)=(1-u)A+uB$$

Pertanto, dato u

$$C=(1-u)A+uB$$

è il punto tra A e B che divide il segmento AB nel rapporto u: $1-u$.

14.8.2 - Interpretazione geometrica

Considerando la prima polilinea P_0-P_1-P_2- … -P_n si utilizza la formula

$$C=(1-u)A+uB$$

per trovare un punto P_i^1 su ogni segmento P_i-P_{i+1} che divide lo stesso nel rapporto u: $1-u$.
Applicando tale metodo si ottengono n nuovi punti P_0^1 (sul segmento P_0-P_1), P_1^1 (sul segmento P_1-P_2), …, P_{n-1}^1 (sul segmento P_{n-1}-P_n), che definiscono una nuova polilinea di $n-1$ legs in n punti.
Applicando una seconda volta il metodo si ottiene una nuova polilinea di $n-2$ legs in $n-1$ punti P_0^2, P_1^2,…,P_{n-2}^2.
Ripetendo la procedura n volte si ottiene un singolo punto P_0^n.
De Casteljau ha dimostrato che tale punto corrisponde al punto C(u) sulla curva.

14.8.3 - Calcolo effettivo

In Fig. 11 è mostrato lo schema a triangolo di De Casteljau

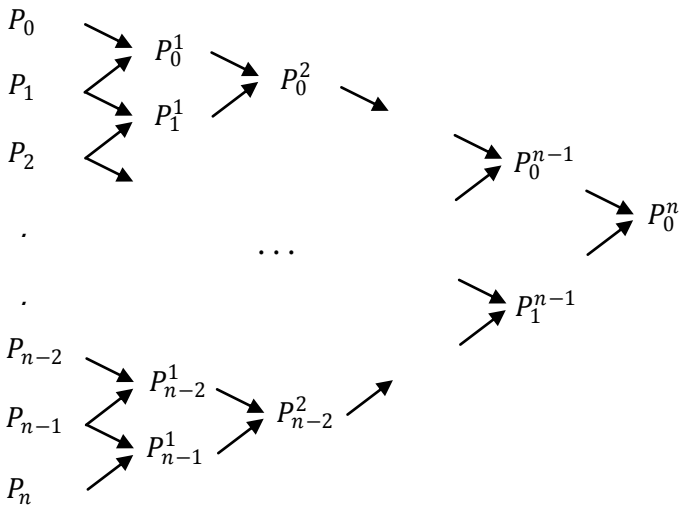


Fig. 11

Come mostrato in figura si dispongono tutti i punti di controllo nella colonna di sinistra.
Per ogni coppia di punti adiacenti si traccia una freccia ↘ che rappresenta il prodotto di $1-u$ per il punto all'origine della freccia, ed una freccia ↗ che rappresenta il prodotto di u sempre per il punto all'origine. Infine si scrive il nuovo punto nell'intersezione delle due frecce.
Se i due punti adiacenti sono P_i^j e P_{i+1}^j il nuovo punto sarà P_i^{j+1} e rappresenterà la somma dei due prodotti.

14.8.4 - Relazione di ricorrenza

Dati P_i punti di controllo si costruiscono i punti successivi con la seguente formula

$$P_i^j = (1-\mathrm{u})\, P_i^{j-1} + u\, P_{i+1}^{j-1} \quad \begin{cases} j = 1, \dots, n \\ i = 0, \dots, n-j \end{cases} \quad u \in R$$

Il punto sulla curva di Bézier sarà calcolato in P_0^n.

14.8.5 - Esempi

Curva di grado 2 (arco di parabola)

Il poligono di controllo è dato da 3 punti P_0, P_1, P_2 (Fig. 12).

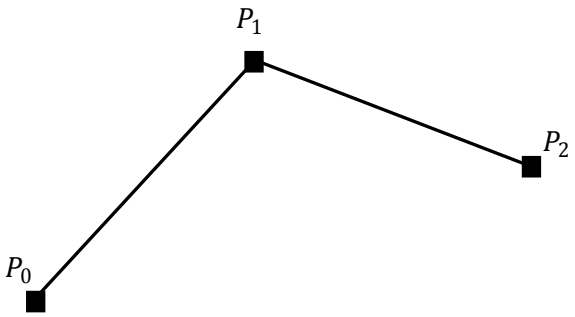


Fig. 12

$$\boldsymbol{u = \frac{1}{2}}$$

In funzione di u si definiscono due nuovi punti sui segmenti P_0-P_1 e P_1-P_2 (Fig. 13).

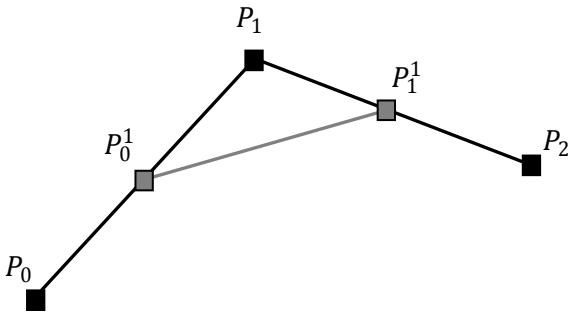


Fig. 13

Sul segmento congiungente questi due punti si definisce un terzo punto, sempre in funzione di u.

Il terzo punto è quello corrispondente alla curva di Bézier di grado 2 per il valore di u (Fig. 14).

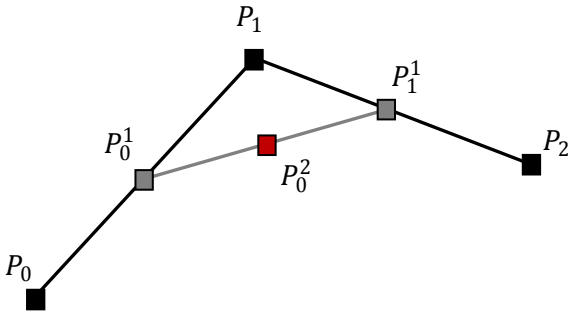


Fig. 14

u = $\frac{1}{3}$

Ripetendo l'algoritmo per $u = \frac{1}{3}$ si ottiene un nuovo punto per la curva di Bézier di grado 2 (Fig. 15).

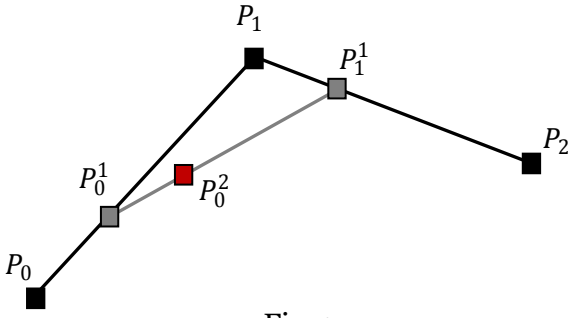


Fig. 15

Ripetendo l'algoritmo per ogni u si ottiene l'intera curva (Fig. 16)

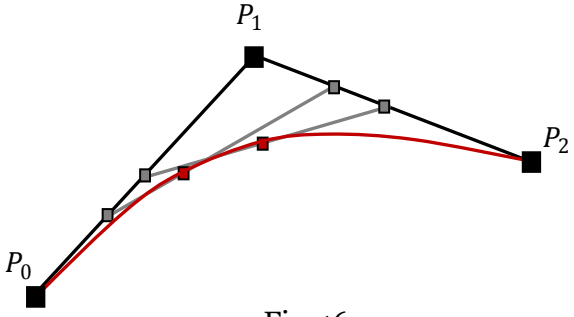


Fig. 16

Seguendo lo schema a triangolo di Fig. 11 la formula generale è ottenuta per sostituzione

$$P_0^1 = (1-u)\,P_0^0 + u\,P_1^0$$

$$P_1^1 = (1-u)\,P_1^0 + u\,P_2^0$$

$$P_0^2 = (1-u)\,P_0^1 + u\,P_1^1 = (1-u)^2\,P_0^0 + 2(1-u)u\,P_1^0 + u^2 P_2^0$$

Curva di grado 3

Il poligono di controllo è dato da 4 punti (Fig. 17).

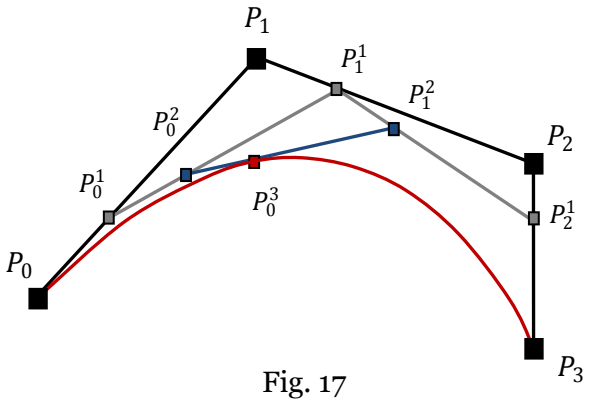


Fig. 17

La formula generale è ottenuta di nuovo per sostituzione

$$P_0^3 = (1-u)^3\, P_0^0 + 3(1-u)^2 u\, P_1^0 + 3(1-u)u^2 P_2^0 + u^3 P_3^0$$

14.8.6 - Implementazioni

Versione iterativa

Input: array P di $n+1$ punti e un numero reale u nell'intervallo [0,1]

Output: $C(u)$, un punto sulla curva

For $i = 0$ to n

    $Q[i] = P[i]$

    For $k = 1$ to n

        For $i = 0$ to $n - k$

            $Q[i] = (1-u) * Q[i] + u * Q[i+1]$

    Return $Q[0]$

Complessità $\theta(n^2)$.

Versione ricorsiva

Sfruttando la relazione di ricorrenza del paragrafo 14.8.4 si può scrivere la seguente procedura ricorsiva

Function deCasteljau (i, j)

Begin

If $(j = 0)$

Return P_i^0

Else

Return $(1-u) * \text{deCasteljau}(i, j-1) + u * \text{deCasteljau}(i+1, j-1)$

End

Questa procedura è semplice e breve, tuttavia risulta estremamente inefficiente poiché ridondante.
La maggior parte delle chiamate vengono ripetute e la complessità è esponenziale $2^{\Omega(n)}$ (lo schema è identico a quello usato per il calcolo dell'n-mo numero di Fibonacci).

Per entrambe le implementazioni l'algoritmo risulta stabile in quanto i coefficienti u e $1-u$ sono compresi tra 0 e 1 e quindi non c'è amplificazione dell'errore.

14.9 - Derivate di una curva di Bézier

Il calcolo delle derivate di una curva di Bézier in un punto è molto semplice.
La curva di Bézier definita da $n + 1$ punti di controllo P_0, P_1,..., P_n è rappresentata dalla seguente equazione

$$C(u) = \sum_{i=0}^{n} B_{n,i}(u)\, P_i \quad u \in [0,1]$$

dove

$$B_{n,i}(u) = \frac{n!}{i!\ (n-i)!} u^i\ (1-u)^{n-i} = \binom{n}{i} u^i (1-u)^{n-i} \quad (0 \le i \le n)$$

Poiché i punti di controllo sono costanti e indipendenti dalla variabile u, il calcolo della derivata $C'(u)$ si riconduce al calcolo delle derivate dei $B_{n,i}(u)$.

$$\frac{d}{du} B_{n,i}(u) = B'_{n,i}(u) = n(B_{n-1,i-1}(u) - B_{n-1,i}(u))$$

Quindi la derivata della curva C(u) risulta

$$\frac{d}{du} C(u) = C'(u) = \sum_{i=0}^{n-1} B_{n-1,i}(u)\, \{\, n(P_{i+1} - P_i)\, \}$$

Definendo

$$Q_0 = n(P_1 - P_0), Q_1 = n(P_2 - P_1), Q_2 = n(P_3 - P_2), \dots, Q_{n-1} = n(P_n - P_{n-1})$$

la derivata della curva si riduce a

$$C'(u) = \sum_{i=0}^{n-1} B_{n-1,i}(u) Q_i$$

La derivata di C(u) è dunque una curva di Bézier di grado $n - 1$ definita per mezzo degli n punti $Q_0, Q_1, Q_2, \dots, Q_{n-1}$.

Un caso particolare è la derivata di una curva di Bézier nel primo e nell'ultimo punto

$$\frac{d}{du} C(0) = C'(0) = n(P_1 - P_0)$$

$$\frac{d}{du} C(1) = C'(1) = n(P_n - P_{n-1})$$

Dal punto di vista fisico la derivata è un odografo, ovvero il vettore normale alla curva in un punto particolare.

14.10 - Continuità

Spesso in grafica per riprodurre forme particolari occorre raccordare più curve.
Un metodo matematico utilizzato per fare ciò sono le derivate in quanto permettono di far congiungere due tratti di curva conservando una certa continuità.
Esistono due tipi di continuità, quella parametrica e quella geometrica.
In generale due curve continue parametricamente saranno continue anche geometricamente ma non viceversa.

14.10.1 - Continuità parametrica

Si parla di continuità parametrica quando due curve adiacenti hanno uguali le derivate parametriche di grado *i* nei punti di giuntura.
Per avere che due derivate siano uguali con continuità parametrica bisogna imporre che i due vettori abbiano la stessa direzione e lo stesso modulo.
La continuità parametrica viene indicata con C_i dove i ne indica il grado.

In particolare C_0 indica che due curve hanno un punto in comune e C_1 indica che due curve hanno sia un punto in comune, sia la derivata prima uguale in direzione e modulo (Fig. 18).

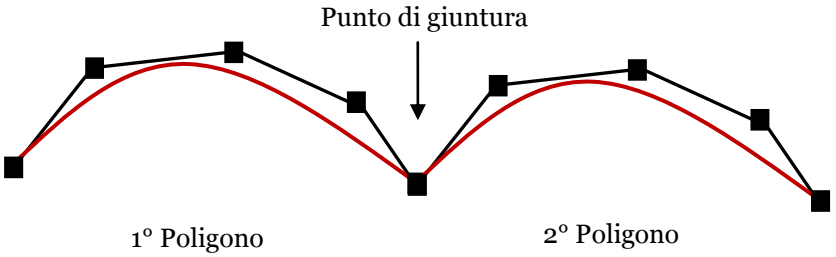


Fig. 18

E' importante osservare che per avere un punto di giuntura tra due curve è richiesta la continuità C_0 , mentre saranno necessarie altre continuità (C_1, C_2 ...) per avere due tratti di curva raccordati con una certa dolcezza nel profilo.

14.10.2 - Continuità geometrica

Questo tipo di continuità è espressa con G ed è una continuità debole.
Richiede che le derivate nei punti di giuntura abbiano la stessa direzione ma non necessariamente lo stesso modulo.

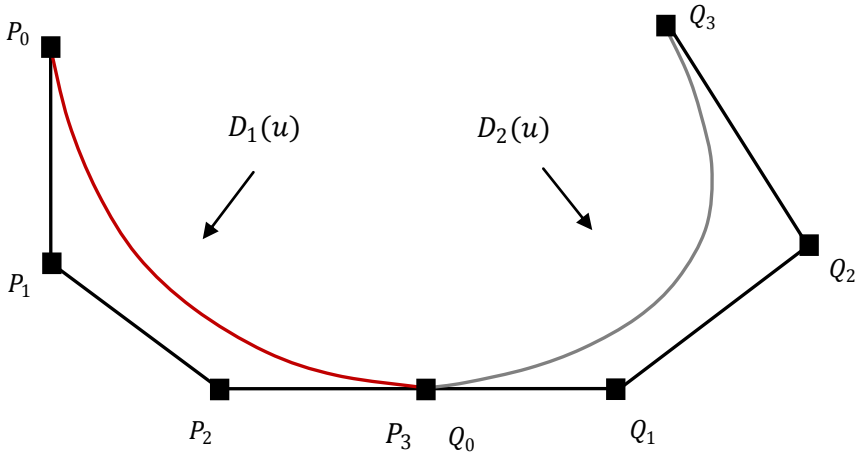


Fig. 19

In Fig. 19 si può constatare cha la continuità C_0 è garantita dall'uguaglianza $P_3 = Q_0$.

Utilizzando le formule presenti nel paragrafo 14.9 e considerando che le curve sono definite tramite 4 punti di controllo, le derivate della prima curva nel punto 1 e della seconda curva nel punto 0 sono pari a

$$\frac{d}{du}D_1(1) = n(P_n - P_{n-1}) = 3(P_3 - P_2)$$

$$\frac{d}{du}D_2(0) = n(P_1 - P_0) = 3(Q_1 - Q_0)$$

Sostituendo $Q_0 = P_3$ nella derivata della seconda curva si ottiene

$$\frac{d}{du}D_2(0) = 3(Q_1 - P_3)$$

Per avere continuità parametrica le due quantità devono essere uguali in modulo e direzione, mentre per avere continuità geometrica i due bracci P_2- P_3 e Q_0- Q_1 devono avere stessa direzione ma possono avere lunghezza diversa.

14.11 - Suddivisione di una curva di Bézier

Il significato di suddivisione è tagliare un'assegnata curva di Bézier in due segmenti di curva, ciascuno dei quali è ancora una curva di Bézier di grado n definita da nuovi punti di controllo. Assegnati $n + 1$ punti di controllo P_0, P_1,..., P_n ed il valore del parametro u compreso tra 0 e 1, si cercano due insiemi di $n + 1$ punti di controllo Q_0, Q_1,..., Q_n e R_0, R_1,..., R_n tali che la curva di Bézier definita dai Q_i (rispettivamente dai R_i) sia la parte di curva originaria su $[0, u]$ (rispettivamente su $[u, 1]$).
La suddivisione di una curva ha diverse applicazioni come il calcolo dell'intersezione di due curve, il rendering e la semplificazione del disegno.
Una curva non soddisfacente può essere suddivisa in due parti, una soddisfacente e l'altra no, in un punto appropriato. Si può, dunque, continuare a lavorare solo sulla parte di curva che non soddisfa.

14.11.1 - Algoritmo di suddivisione

Per ottenere la suddivisione si utilizza l'algoritmo e lo schema a triangolo di De Casteljau. Durante l'esecuzione dell'algoritmo di De Casteljau si raccolgono il primo e l'ultimo punto di ogni colonna.
Tutti i primi punti formeranno la suddivisione che corrisponde alla parte di curva originale in $[0, u]$, mentre tutti gli ultimi punti formeranno la suddivisione che corrisponde alla parte di curva originale in $[u, 1]$.

Un esempio è fornito in Fig. 20

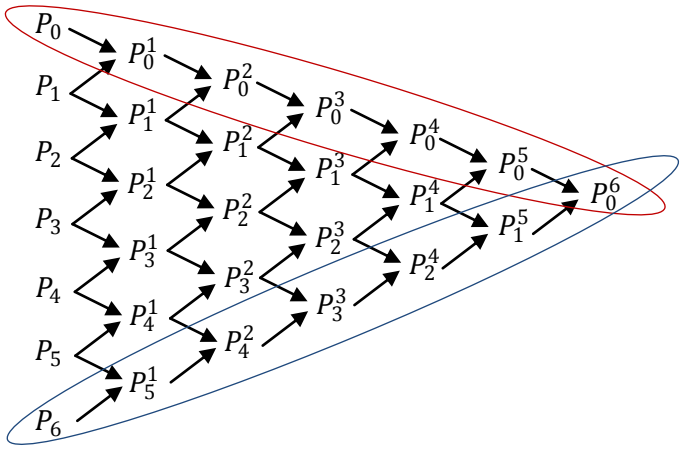


Fig. 20

Nello schema triangolare il bordo superiore in direzione delle frecce fornisce i punti di controllo del primo segmento di curva, mentre il bordo inferiore nella direzione opposta delle frecce fornisce i punti di controllo del secondo segmento di curva.

14.12 - Elevazione grado di una curva di Bézier

Molte applicazioni che vedono coinvolte due o più curve di Bézier richiedono che tutte le curve abbiano lo stesso grado. Inoltre curve di Bézier di grado più elevato risultano maggiormente flessibili nel disegnare forme anche se richiedono un maggiore tempo di calcolo.
Il punto fondamentale dell'elevazione di grado di una curva di Bézier è non variarne la forma.

14.12.1 - Algoritmo del Degree Elevation

Dati $n+1$ punti di controllo P_0,P_1,...,P_n e la rispettiva curva di Bézier si definisce un nuovo insieme di $n+2$ punti di controllo in modo che la curva generata da questi nuovi punti abbia grado $n+2-1=n+1$.
I punti P_0e P_n devono far parte di questo nuovo insieme poiché la curva deve passare per gli estremi. Rimangono dunque da definire n punti di controllo.
Siano Q_0, Q_1,..., Q_{n+1} i nuovi punti di controllo con $Q_0 = P_0$e $Q_{n+1} = P_n$, si definiscono gli altri punti come

$$Q_i = \frac{i}{n+1}P_{i-1} + (1 - \frac{i}{n+1})P_i \quad 1 \le i \le n$$

Ogni leg della polilinea di controllo conterrà un nuovo punto di controllo (il segmento P_{i-1}- P_i conterrà il punto Q_i).
Ricordando l'algoritmo di De Casteljau si osserva che ogni nuovo punto Q_i dividerà il segmento P_{i-1}- P_i nel rapporto

$$1 - \frac{i}{n+1} : \frac{i}{n+1}$$

Contrariamente all'algoritmo di De Casteljau il rapporto non sarà costante ma dipenderà da i.
Poiché ogni gamba della polilinea di controllo originaria conterrà un nuovo punto, la procedura di sostituzione del vecchio insieme di punti con quello nuovo può essere interpretata come un'eliminazione degli angoli dei punti di controllo originari.

In Fig. 21 viene mostrata l'elevazione di grado di un curva di Bézier definita mediante 5 punti di controllo e l'effetto di rimozione degli angoli.
Nella tabella vengono forniti i rapporti su ogni leg della polilinea di controllo originale.

i	$1 - \frac{i}{n+1}$
1	0.8
2	0.6
3	0.4
4	0.2

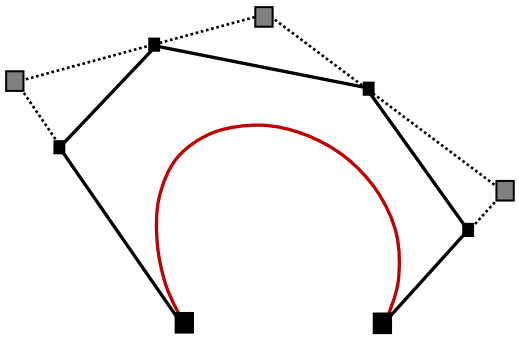


Fig. 21

L'algoritmo può essere applicato più volte.
Ad ogni passo aumenta il numero di punti di controllo mentre la nuova polilinea si sposta verso la curva che non cambia forma (Fig. 22.1, 22.2, 22.3, 22.4, 22.5, 22.6).

$Degree = 6$

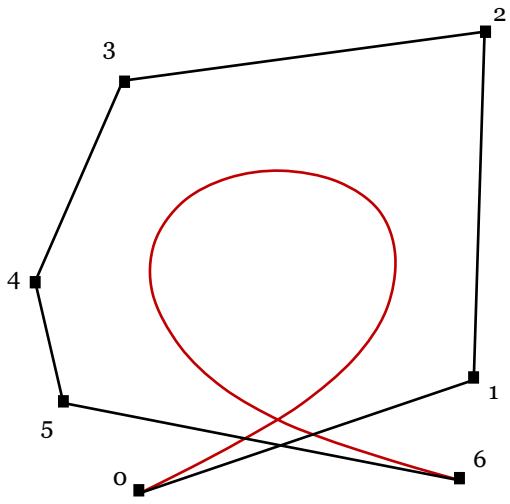


Fig. 22.1

$Degree = 7$

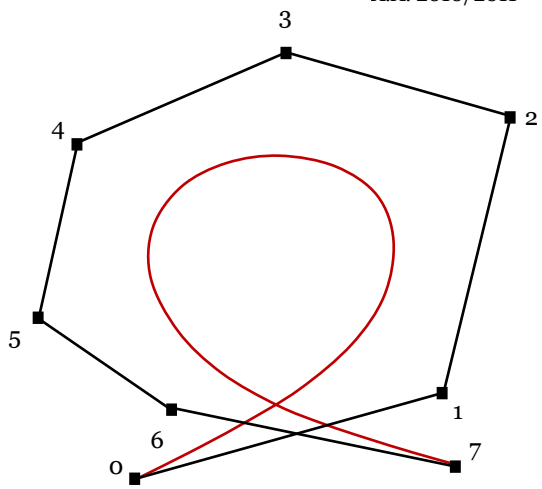


Fig. 22.2

$Degree = 8$

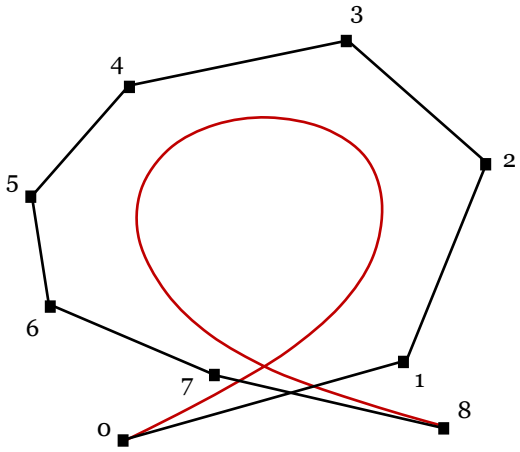


Fig. 22.3

$Degree = 10$

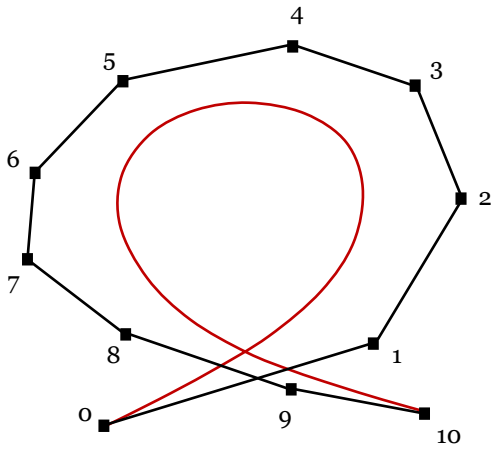


Fig. 22.4

$Degree = 15$

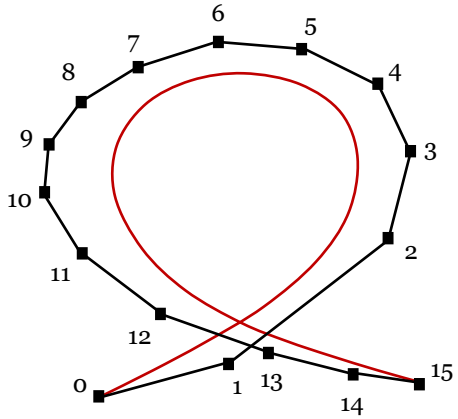


Fig. 22.5

$Degree = 29$

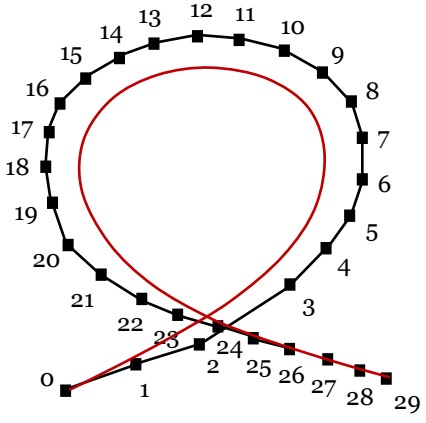


Fig. 22.6

Al limite, quando il grado tende a ∞, la polilinea di controllo si avvicina alla curva che ne risulta la posizione limite.
L'algoritmo si presenta stabile e ricorsivo ma lento in quanto per ottenere un'ottima approssimazione della curva di Bézier è necessario iterare la procedura più volte. Per tale motivo l'algoritmo è poco usato in grafica.

14.13 - Curve di Bézier razionali

Alcune curve che sembrano semplici, come il cerchio, non possono essere descritte da una curva di Bézier.
Le curve di Bézier Razionali aggiungono dei pesi che vengono utilizzati per poter modellare gli oggetti scegliendo un peso per un determinato punto.
Dati $n + 1$ punti di controllo P_i la curva di Bézier Razionale è data da

$$C(u) = \frac{\sum_{i=0}^{n} B_{n,i}(u)\, P_i\, w_i}{\sum_{i=0}^{n} B_{n,i}(u)\, w_i} \qquad u \in [0,1]$$

Il numeratore rappresenta una curva di Bézier pesata mentre il denominatore rappresenta la somma pesata dei polinomi di Bernstein.

E' importante osservare che all'aumentare del peso w_i la curva $C(u)$ si spinge verso il punto P_i , viceversa al diminuire di w_i la curva si allontana da tale punto (Fig. 23).

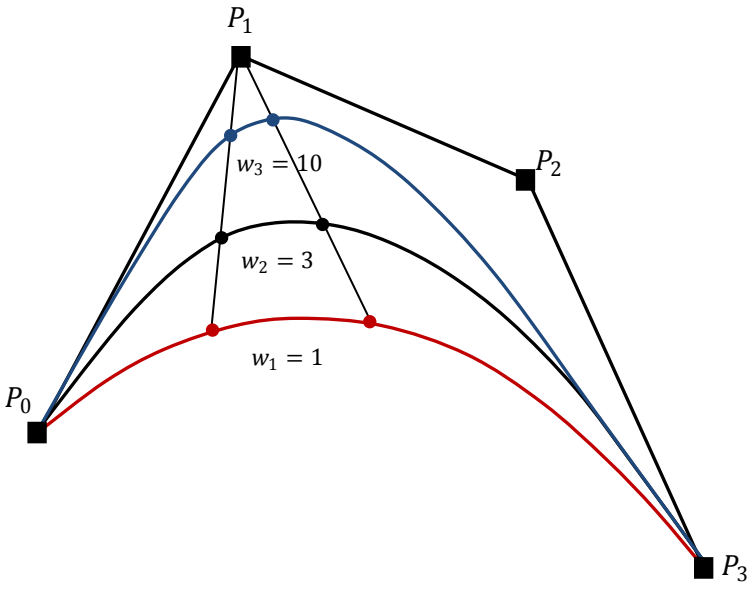


Fig. 23

14.14 - Superfici di Bézier

Le superfici di Bézier sono uno strumento molto potente per la progettazione di superfici e vengono ampiamente utilizzate nella computer graphics assieme alle curve di Bézier.
L'idea alla base è di definire una superficie a partire da un numero finito di punti P_0,P_1..., P_n nello spazio, in modo che il suo supporto segua l'andamento del poliedro, detto poliedro di controllo, individuato da questi.
La costruzione delle superfici di Bézier avviene mediante i polinomi di Bernstein.

14.15 - Superfici parametriche

Le superfici parametriche sono definite da un insieme di tre funzioni, una per ogni coordinata.

$$f(u,v) = [x(u,v); y(u,v); z(u,v)]$$

Si assume che $u, v \in [0,1]$ quindi (u,v) sarà un punto nel quadrato definito dai punti $(0,0),(1,0),(0,1),(1,1)$.
Se $x(u,v)$, $y(u,v)$, $z(u,v)$ sono polinomi, le superfici parametriche non sono utili per rappresentare la maggior parte delle superfici (spire, sfere, ecc..), si utilizzano allora le superfici implicite.

14.16 - Definizione

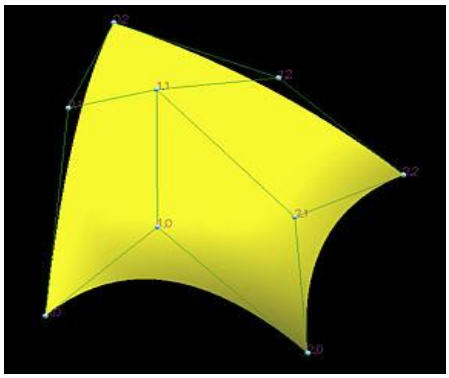
Le superfici di Bézier vengono definite considerando un insieme di punti $P_{i,j}$ bidimensionali con $i = 0, \dots, m$ e $j = 0, \dots, n$, distribuiti su $m+1$ righe e $n+1$ colonne per un totale di $(m+1)(n+1)$ punti di controllo. Questi punti sono distribuiti su di un reticolo che prende il nome di poliedro di controllo o control net.
La superficie di Bézier è descritta dalla formula

$$P(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} B_{m,i}(u) B_{n,j}(v) P_{i,j} \qquad u, v \in [0,1]$$

$$B_{m,i}(u) = \frac{m!}{i!\,(m-i)!} u^i (1-u)^{m-i} \qquad (0 \le i \le m)$$

$$B_{n,j}(v) = \frac{n!}{j!\,(n-j)!} v^j (1-v)^{n-j} \qquad (0 \le j \le n)$$

dove $B_{m,i}(u)$ e $B_{n,j}(v)$ sono l'i-simo e il j-simo polinomio di Bernstein in u, rispettivamente in v.
Dal momento che il grado di $B_{m,j}(v)$ é m e il grado di $B_{n,i}(u)$ è n, il grado di una superficie di Bézier è definito come (m, n).
La figura mostra una superficie di Bézier definita da 9 punti di controllo con grado (2, 2).



14.17 - Superfici prodotto tensoriale

Il prodotto tensoriale è una tecnica che permette di costruire superfici moltiplicando due curve.
Date due curve di Bézier il prodotto tensoriale costruisce una superficie moltiplicando le funzioni base della prima curva con le funzioni base della seconda e utilizza i risultati ottenuti come funzioni base per l'insieme di punti di controllo bidimensionali.
La superficie generata viene chiamata superficie prodotto tensoriale.
Le superfici di Bézier sono quindi superfici prodotto tensoriale.

14.18 - Proprietà

Le superfici di Bézier godono di alcune proprietà che derivano dalle proprietà dei polinomi di Bernstein e dalle proprietà delle curve di Bézier.

Interpolazione degli estremi

La superficie di Bézier interpola i 4 angoli del control net $P_{0,0}$, $P_{m,0}$, $P_{m,n}$, $P_{0,n}$.

$$P(0,0) = P_{0,0}$$

$$P(1,0) = P_{m,0}$$

$$P(0,1) = P_{0,n}$$

$$P(1,1) = P_{m,n}$$

Non negatività

Il prodotto $B_{m,i}(u)B_{n,j}(v)$ è non negativo $\forall\ m, n, i, j \quad u, v \in [0,1]$.

Partizione dell'unità

$$\sum_{i=0}^{m}\sum_{j=0}^{n} B_{m,i}(u)B_{n,j}(v) = 1 \quad \forall u, v \in [0,1]$$

Quindi la superficie di Bézier dipende unicamente dalla scelta dei punti di controllo e non dalla scelta delle origini delle coordinate.

Convex Hull

Dalle due proprietà precedenti si osserva che $P(u,v)$ è una combinazione lineare di punti di controllo e coefficienti positivi la cui somma è pari a 1.
La superficie di Bézier sarà dunque contenuta nel dominio convesso del control net.

Invariante affine

Quando occorre una trasformazione affine su una superficie di Bézier si opera solo sui punti di controllo. Ottenuti i punti trasformati la superficie di Bézier trasformata è definita dai nuovi punti.

Variazione di diminishing

Le superfici di Bézier, differentemente dalle curve di Bézier, non godono della proprietà di riduzione delle oscillazioni.

14.19 - Limitazioni

Le superfici di Bézier hanno alcune limitazioni.

- Non si possono rappresentare sfere o calotte sferiche, né superfici di rotazione.
- Non ammettono controllo locale, pertanto la variazione di un solo punto del poliedro di controllo porta ad un rilevante cambiamento di tutta la superficie.
- Per rappresentare forme complesse, usando una sola superficie, sono necessari molti punti di controllo. Questo significa che il grado dei polinomi usati può diventare molto alto e quindi difficile da trattare al calcolatore.

14.20 - Algoritmo di De Casteljau

L'algoritmo di De Casteljau può essere esteso per calcolare anche le superfici di Bézier.
L'algoritmo può essere applicato diverse volte finché non si ottiene il punto $P(u,v)$ corrispondente sulla superficie di Bézier.
Dato (u,v), la superficie è descritta dalla formula

$$P(u,v) = \sum_{i=0}^{m}\sum_{j=0}^{n} B_{m,i}(u)B_{n,j}(v)P_{i,j} \qquad u,v \in [0,1]$$

che può essere riscritta come

$$P(u,v) = \sum_{i=0}^{m} B_{m,i}(u)\left(\sum_{j=0}^{n} B_{n,j}(v)P_{i,j}\right) \qquad u,v \in [0,1]$$

Per $i = 0 \ldots m$ si definisce

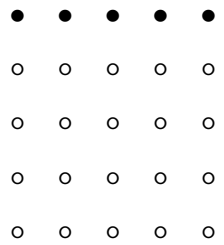
$$q_i(v) = \sum_{j=0}^{n} B_{n,j}(v) P_{i,j}$$

Dove q_i(v) rappresenta un punto sulla curva di Bézier definita dai punti di controllo $P_{i,0}$, $P_{i,1}$, …, $P_{i,\mathrm{n}}$

Per calcolare q_i(v) $\forall i = 0 \ldots m$ si utilizza l'algoritmo di De Casteljau.

Per $i = 0$

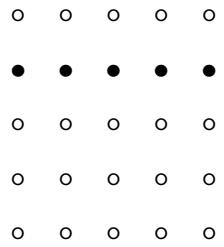
$$q_0(v) = \sum_{j=0}^{n} B_{n,j}(v) P_{0,j}$$



i punti di controllo rappresentano la prima riga del control net.

Per $i = 1$

$$q_1(v) = \sum_{j=0}^{n} B_{n,j}(v) P_{1,j}$$



i punti di controllo rappresentano la seconda prima del control net.

…

Si ottiene infine la formula per la superficie di Bézier

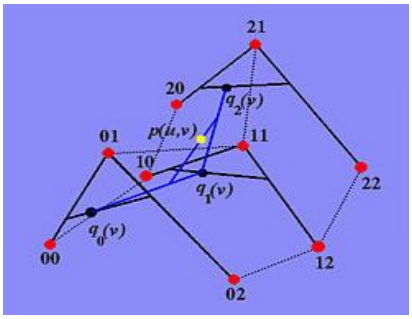
$$P(u,v) = \sum_{i=0}^{m} B_{m,i}(u)\, q_i(v)$$

che rappresenta una curva di Bézier definita da $m + 1$ punti di controllo q_0(v), q_1(v), …, q_{m}(v).

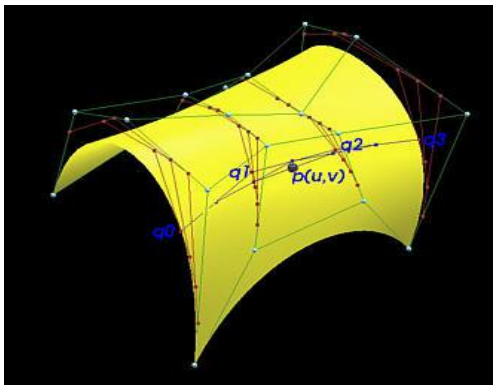
In conclusione per calcolare il punto $P(u, v)$ sulla superficie di Bézier, si cercano $m + 1$ punti $q_0(\mathrm{v})$, $q_1(\mathrm{v})$, ..., $q_\mathrm{m}(\mathrm{v})$ e da questi si ottiene il punto cercato.
Ogni $q_\mathrm{i}(\mathrm{v})$ rappresenta un punto sulla curva di Bézier definita dalla i-sima riga dei punti di controllo $P_{i,0}$, $P_{i,1}$, ..., $P_{i,\mathrm{n}}$, e per calcolarne il valore si applica l'algoritmo di De Casteljau.
Dopo $m + 1$ applicazioni dell'algoritmo si ottengono i valori di $q_0(\mathrm{v})$, $q_1(\mathrm{v})$, ..., $q_\mathrm{m}(\mathrm{v})$ e applicandovi l'algoritmo di De Casteljau con la variabile u si perviene al punto finale della superficie $P(u, v)$.

14.21 - Esempi

Nel diagramma che segue la superficie di Bézier è definita da 9 punti di controllo e ha grado (2,2).
Il valore di u è pari $\frac{2}{3}$ mentre il valore di v è $\frac{1}{3}$.
Per determinare il valore di $q_0\left(\frac{1}{3}\right)$ si prende la prima riga del control net formata dai punti $P_{0,0}$, $P_{0,1}$, $P_{0,2}$ e si applica l'algoritmo di De Casteljau alla curva di Bézier in $v = \frac{1}{3}$.
Si ripete la procedura per la seconda e la terza riga del control net sempre in $v = \frac{1}{3}$, ottenendo i tre punti di controllo $q_0\left(\frac{1}{3}\right)$, $q_1\left(\frac{1}{3}\right)$, $q_2\left(\frac{1}{3}\right)$.
Infine si applica l'algoritmo di De Casteljau ai tre nuovi punti di controllo calcolati pervenendo al punto $P(\frac{2}{3}, \frac{1}{3})$, rappresentato in giallo nella figura.



Nella figura che segue viene mostrata una superficie di Bézier.
I punti di controllo sono mostrati in bianco e la control net ha 4 righe e 5 colonne per un grado pari a (3,4). Per ogni riga, le polilinee intermedie utilizzate dall'algoritmo di De Casteljau sono mostrate in rosso, mentre le polilinee intermedie per il calcolo dei punti sulla superficie sono mostrate in blu. Vengono inoltre riportati i quattro punti di controllo intermedi q_0, q_1, q_2, q_3.



14.22 - Implementazione

Input: $m + 1$ righe e $n + 1$ colonne di punti di controllo e la coppia (u,v).

Output: un punto sulla superficie di Bézier $P(u, v)$

For $i = 0$ to m do

begin

Applica l'algoritmo di De Casteljau alla i-sima riga del controllo net in v

Assegna a $q_i(v)$ il punto ottenuto

end

Applica l'algoritmo di De Casteljau ai punti di controllo $q_0(v)$, $q_1(v)$, …, $q_m(v)$ in u

Il punto ottenuto è $P(u, v)$