# Capitolo 1

# Approssimazione di autovalori e autovettori

## 1.1 Richiami di algebra lineare

In questo capitolo descriveremo il metodo delle potenze per il calcolo dell'autovalore di massimo modulo di una matrice reale e una sua importante applicazione, cioè il cosiddetto algoritmo per il PageRank utilizzato dal motore di ricerca Google per assegnare un ranking alle pagine web. Prima di affrontare questo problema richiamiamo alcuni concetti e definizioni di algebra lineare.
D'ora in poi saranno considerati vettori appartenenti all'insieme $\mathbb{R}^n$.

**Definizione 1.1.1** *Si definisce Prodotto scalare ogni funzione*

$$(\cdot,\cdot) : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$$

*che verifica le seguenti proprietà:*

1. $(\mathbf{x},\mathbf{x}) \geq 0$ *per ogni* $\mathbf{x} \in \mathbb{R}^n$ *e* $(\mathbf{x},\mathbf{x}) = 0$ *se e solo se* $\mathbf{x} = 0$*;*
2. $(\mathbf{x},\mathbf{y}) = (\mathbf{y},\mathbf{x})$*;*
3. $(\alpha\mathbf{x} + \beta\mathbf{y},\mathbf{z}) = \alpha(\mathbf{x},\mathbf{z}) + \beta(\mathbf{y},\mathbf{z})$ *per ogni* $\alpha,\beta \in \mathbb{R}$ *e per ogni* $\mathbf{x},\mathbf{y},\mathbf{z} \in \mathbb{R}^n$.

Il classico prodotto scalare su $\mathbb{R}^n$ è definito come

$$(\mathbf{x},\mathbf{y}) = \mathbf{y}^T\mathbf{x} = \sum_{i=1}^{n} x_i y_i.$$

Se il prodotto scalare tra due vettori è uguale a zero allora i due vettori si dicono ortogonali.

**Definizione 1.1.2** *La funzione* $\|\cdot\| : \mathbb{R}^n \to \mathbb{R}_+$ *si dice norma vettoriale se per ogni* $\mathbf{x} \in \mathbb{R}^n$ *soddisfa le seguenti proprietà:*

1. $\|\mathbf{x}\| \geq 0$ *e* $\|\mathbf{x}\| = 0$ *se e solo se* $\mathbf{x} = 0$*;*
2. $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|$ *per ogni* $\alpha \in \mathbb{R}$*;*
3. $\|\mathbf{x}+\mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$ *per ogni* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ *(Disuguaglianza triangolare).*

Poichè il prodotto scalare tra un vettore e se stesso è un valore nonnegativo allora ogni prodotto scalare induce una norma, ed in particolare il prodotto scalare su $\mathbb{R}^n$ definisce la seguente norma:

$$\|\mathbf{x}\|_2 = \sqrt{(\mathbf{x}, \mathbf{x})} = \sqrt{\mathbf{x}^T\mathbf{x}} = \sqrt{\sum_{i=1}^{n} x_i^2}$$

che viene appunto norma 2 o norma euclidea (ed indica la lunghezza del vettore stesso). Altre norme molto utilizzate sono le seguenti:

$$\|\mathbf{x}\|_1 = \sum_{j=1}^{n} |x_j| \qquad \text{norma 1}$$

$$\|\mathbf{x}\|_\infty = \max_{1 \leq j \leq n} |x_j| \qquad \text{norma infinito.}$$

Un importante legame tra prodotto scalare e norma è la cosiddetta disuguaglianza di Schwarz:

$$|(\mathbf{x}, \mathbf{y})|^2 \leq \|\mathbf{x}\|\|\mathbf{y}\| \tag{1.1}$$

Se un vettore $\mathbf{x}$ ha norma diversa da zero allora il vettore $\mathbf{x}/\|\mathbf{x}\|$ ha norma unitaria. Se $\mathbf{x}$ è un vettore di lunghezza unitaria allora il prodotto $\mathbf{x}^T\mathbf{y}$ fornisce la proiezione sulla semiretta su cui giace il vettore $\mathbf{x}$.
In maniera simile è possibile definire anche norme su matrici quadrate.

**Definizione 1.1.3** *Una funzione* $\|\cdot\| : \mathbb{R}^{n\times n} \to \mathbb{R}$ *che per ogni matrice* $A \in \mathbb{R}^{n\times n}$ *soddisfa le seguenti proprietà:*

1. $\|A\| \geq 0$ *e* $\|A\| = 0$ *se e solo se* $A = 0$*;*

2. $\|\alpha A\| = |\alpha|\|A\|$ *per ogni* $\alpha \in \mathbb{R}$;

3. $\|A + B\| \leq \|A\| + \|B\|$ *per ogni* $A, B \in \mathbb{R}^{n\times n}$;

4. $\|A \cdot B\| \leq \|A\| \cdot \|B\|$ *per ogni* $A, B \in \mathbb{R}^{n\times n}$;

*si dice norma matriciale.*

**Definizione 1.1.4** *Si dice che una norma di matrice è compatibile con una norma di vettore se per ogni matrice* $A$ *e per ogni vettore* $\mathbf{x}$ *risulta*

$$\|A\mathbf{x}\| \leq \|A\|\|\mathbf{x}\|.$$

Un modo per definire le norme di matrici compatibili con norme di vettori è il seguente. Sia $\mathbf{x} \neq 0$ con norma $\|\mathbf{x}\|$. Considerata la norma del vettore $A\mathbf{x}$, $\|A\mathbf{x}\|$, definiamo come norma di $A$ il numero $\|A\|$ dato da

$$\|A\| = \max_{\|\mathbf{x}\|=1} \|A\mathbf{x}\| \tag{1.2}$$

detta norma indotta dalla norma vettoriale $\|\mathbf{x}\|$.
Le principali norme matriciali indotte sono date le seguenti:

$$\begin{aligned} \|A\|_1 &= \max_{1\leq j\leq n} \sum_{i=1}^{n} |a_{ij}| \\ \|A\|_2 &= \sqrt{\rho(A^T A)} \\ \|A\|_\infty &= \max_{1\leq i\leq n} \sum_{j=1}^{n} |a_{ij}| \end{aligned}$$

dove $\rho(A^T A)$ indica il *raggio spettrale* di $A^T A$, ovvero la radice quadrata del massimo modulo di un autovalore della matrice $A^T A$.

**Definizione 1.1.5** *Sia* $A \in \mathbb{C}^{n\times n}$ *si dice che* $\lambda \in \mathbb{C}$ *è un autovalore di* $A$ *se e solo se esiste un vettore* $\mathbf{x} \in \mathbb{C}^n$, $\mathbf{x} \neq 0$, *tale che:*

$$A\mathbf{x} = \lambda\mathbf{x}. \tag{1.3}$$

Se $\lambda \in \mathbb{C}$ è un autovalore di $A$, ogni vettore non nullo $\mathbf{x}$ che soddisfa la relazione (1.3) si dice autovettore associato a $\lambda$.
La relazione (1.3) può essere scritta come

$$(A - \lambda I)\mathbf{x} = 0, \tag{1.4}$$

dunque il vettore $\mathbf{x}$ deve essere una soluzione non banale del sistema omogeneo (1.4). Affinchè quest'ultimo ammetta soluzioni non banali deve essere

$$\det(A - \lambda I) = 0. \tag{1.5}$$

Sviluppando il primo membro in (1.5) con la regola di Laplace è facile vedere che rappresenta un polinomio di grado $n$ in $\lambda$,

$$p_n(\lambda) = \det(A - \lambda I),$$

detto polinomio caratteristico di $A$. Gli autovalori di $A$ sono le $n$ radici dell'equazione caratteristica

$$\det(A - \lambda I) \equiv p_n(\lambda) = 0.$$

Se la matrice $A$ ha elementi reali allora i coefficienti del polinomio caratteristico sono reali, tuttavia gli autovalori possono anche essere numeri complessi.

## 1.2 Il Metodo delle Potenze

In questo paragrafo sarà descritto un metodo per il calcolo di un particolare autovalore di una matrice reale. Sia $A$ una matrice di ordine $n$ ad elementi reali, tale che $\lambda_1, \lambda_2, \ldots, \lambda_n$ siano i suoi autovalori e $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ i corrispondenti autovettori. Quindi

$$A\mathbf{v}_i = \lambda_i \mathbf{v}_i, \qquad i = 1, \ldots, n. \tag{1.6}$$

Ricordiamo inoltre che, moltiplicando l'equazione (1.6) per $A$ risulta

$$A^2\mathbf{v}_i = \lambda_i A\mathbf{v}_i = \lambda_i^2 \mathbf{v}_i,$$

e, generalizzando, risulta

$$A^k\mathbf{v}_i = \lambda_i^k \mathbf{v}_i,$$

ovvero gli autovalori delle potenze di $A$ sono le potenze degli autovalori di $A$. Assumiamo le seguenti ipotesi semplificative:

1. $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$
2. $\{\mathbf{v}_i\}_{i=1}^n$ linearmente indipendenti.

Sotto la prima ipotesi $\lambda_1$ è detto autovalore dominante. Scopo del metodo delle potenze è quello di determinare $\lambda_1$ e il corrispondente autovettore. Una prima osservazione è che, sotto queste ipotesi, $\lambda_1$ deve essere necessariamente un numero reale. Infatti se la matrice $A$ è reale allora anche il polinomio caratteristico ha coefficienti reali. Se $\lambda_1$ fosse un numero complesso allora anche il suo coniugato, $\overline{\lambda}_1$ sarebbe autovalore di $A$. In questo caso sarebbe violata la prima condizione, in quanto $\lambda_1$ e $\overline{\lambda}_1$ hanno il medesimo modulo.
Supponiamo ora che sia $\mathbf{x}^{(0)} \in \mathbb{R}^n$ un vettore arbitrario. Definiamo il processo iterativo

$$\mathbf{x}^{(k+1)} = A\mathbf{x}^{(k)} \qquad k = 0, 1, 2, \ldots \tag{1.7}$$

Ogni vettore appartenente alla successione può essere espresso come

$$\mathbf{x}^{(k)} = A^k\mathbf{x}^{(0)} \qquad k = 0, 1, 2, \ldots \tag{1.8}$$

ovvero come prodotto tra una potenza della matrice $A$ ed il vettore iniziale (da qui il nome di Metodo delle potenze per indicare l'algoritmo in questione). Poichè i vettori $\{\mathbf{v}_i\}_{i=1}^n$ sono linearmente indipendenti formano una base quindi $\mathbf{x}^{(0)}$ può essere espresso nel seguente modo:

$$\mathbf{x}^{(0)} = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i, \tag{1.9}$$

ipotizzando, per semplicità, che sia $\alpha_1 \neq 0$. Sostituendo (1.9) in (1.8), abbiamo

$$\mathbf{x}^{(k)} = A^k \sum_{i=1}^{n} \alpha_i \mathbf{v}_i = \sum_{i=1}^{n} \alpha_i A^k \mathbf{v}_i = \sum_{i=1}^{n} \alpha_i \lambda_i^k \mathbf{v}_i.$$

Poichè $\alpha_1 \neq 0$ allora si può scrivere

$$\mathbf{x}^{(k)} = \lambda_1^k \left( \alpha_1 \mathbf{v}_1 + \sum_{i=2}^{n} \alpha_i \frac{\lambda_i^k}{\lambda_1^k} \mathbf{v}_i \right) = \lambda_1^k (\alpha_1 \mathbf{v}_1 + \varepsilon^{(k)}) \tag{1.10}$$

dove si è posto

$$\varepsilon^{(k)} = \sum_{i=2}^{n} \alpha_i \left( \frac{\lambda_i}{\lambda_1} \right)^k \mathbf{v}_i.$$

Scrivendo la (1.10) per $k+1$ segue che

$$\mathbf{x}^{(k+1)} = \lambda_1^{k+1}(\alpha_1 \mathbf{v}_1 + \varepsilon^{(k+1)}). \tag{1.11}$$

Detto $\mathbf{e}_j$ il $j$-esimo versore fondamentale di $\mathbb{R}^n$, e, tenendo conto di (1.10) e (1.11), definiamo come $(k+1)$-esimo elemento della successione $\lambda_1^{(k+1)}$ il rapporto tra le $j$-esime componenti dei vettori $\mathbf{x}^{(k+1)}$ e $\mathbf{x}^{(k)}$, cioè

$$\lambda_1^{(k+1)} = \frac{\mathbf{e}_j^T \mathbf{x}^{(k+1)}}{\mathbf{e}_j^T \mathbf{x}^{(k)}} \tag{1.12}$$

ovvero

$$\lambda_1^{(k+1)} = \frac{\lambda_1^{k+1}(\alpha_1 \mathbf{e}_j^T \mathbf{v}_1 + \mathbf{e}_j^T \varepsilon^{(k+1)})}{\lambda_1^k(\alpha_1 \mathbf{e}_j^T \mathbf{v}_1 + \mathbf{e}_j^T \varepsilon^{(k)})} = \frac{\lambda_1(\alpha_1 \mathbf{e}_j^T \mathbf{v}_1 + \mathbf{e}_j^T \varepsilon^{(k+1)})}{\alpha_1 \mathbf{e}_j^T \mathbf{v}_1 + \mathbf{e}_j^T \varepsilon^{(k)}}.$$

Poichè

$$|\mathbf{e}_j^T \varepsilon^{(k)}| \overset{k\to\infty}{\longrightarrow} 0$$

in quanto $|\lambda_i/\lambda_1| < 1$ per ogni $i$, segue

$$\lim_{k\to\infty} \frac{\mathbf{e}_j^T \mathbf{x}^{(k+1)}}{\mathbf{e}_j^T \mathbf{x}^{(k)}} = \lambda_1 \, \frac{\alpha_1 \mathbf{e}_j^T \mathbf{v}_1}{\alpha_1 \mathbf{e}_j^T \mathbf{v}_1} = \lambda_1.$$

Inoltre, dalla (1.10), sempre poichè $|\lambda_i/\lambda_1| < 1$ per ogni $i$, segue

$$\lim_{k\to\infty} \frac{1}{\lambda_1^k} \, \mathbf{x}^{(k)} = \alpha_1 \mathbf{v}_1.$$

Quest'ultima relazione ci informa che per $k$ sufficientemente grande, $\mathbf{x}^{(k)}$ tende all'autovettore $\mathbf{v}_1$.
Si deve considerare che, se $|\lambda_1| > 1$, si ha

$$\mathbf{x}^{(k)} \simeq |\lambda_1|^k \alpha_1 \mathbf{v}_1 \longrightarrow \infty$$

pertanto quando l'algoritmo viene implementato così come descritto possono nascere facilmente problemi di overflow (o analogamente problemi di underflow se risulta $|\lambda_1| < 1$).
È possibile evitare queste circostanze osservando che in realtà siamo interessati al rapporto $\mathbf{e}_j^T \mathbf{x}^{(k+1)} / \mathbf{e}_j^T \mathbf{x}^{(k)}$ e dunque è possibile normalizzare la successione $\mathbf{x}^{(k)}$ lasciando inalterata tale quantità. È possibile allora modificare

l'algoritmo descritto nel seguente modo:
1. Si sceglie un vettore $\mathbf{x}^{(0)}$ tale che

$$\|\mathbf{x}^{(0)}\|_\infty = 1;$$

2. Si calcola il vettore

$$\mathbf{y}^{(k+1)} = A\mathbf{x}^{(k)};$$

3. Si pone

$$\lambda_1^{(k+1)} = \frac{e_{i_k}^T \mathbf{y}^{(k+1)}}{e_{i_k}^T \mathbf{x}^{(k)}}$$

4. Si calcola il vettore

$$\mathbf{x}^{(k+1)} = \frac{\mathbf{y}^{(k+1)}}{\|\mathbf{y}^{(k+1)}\|_\infty}$$

Il processo iterativo viene fermato quando due elementi della successione sono sufficentemente vicini, cioè risulta

$$|\lambda_1^{(k+1)} - \lambda_1^{(k)}| \leq \varepsilon$$

dove $\varepsilon$ è la precisione fissata.
L'algoritmo codificato in MatLab è il seguente ($A$ è una matrice di ordine $n$ mentre `tol` è la precisione fissata):

```
function [lambda1,k] = potenze(A,tol)
%
% [lambda1,k] = potenze(A)
%
% Parametri di input
% A = matrice quadrata di ordine n
% tol = tolleranza voluta
%
% Parametri di output
% lambda1 = approssimazione dell'autocalore dominante
%         Posto uguale a inf se il metodo non converge
%         dopo 500 iterazioni
% Num_it = numero di iterazioni
%         Posto uguale a inf se il metodo non converge
%
n = size(A,1);
```

```
% Se la tolleranza non e' un parametro specificato
% in input allora tol = 1e-8 per default
if nargin == 1
     tol = 1e-8;
end
% Si fissa a 500 il numero massimo di iterazioni
Max_it = 500;
error = 2*tol;
x = ones(n,1);
lambda0 = 0;
Num_it = 0;
while error > tol
     y = A*x;
     [norma,k] = max(abs(y));
     lambda1 = y(k)/x(k);
     x = y/norma;
     error = abs(lambda1-lambda0);
     lambda0 = lambda1;
     Num_it = Num_it+1;
     If Num_it > Max_it
          Num_it = inf;
          lambda1 = inf;
          return
     end
end
```

Nell'algoritmo possiamo scegliere ad ogni passo l'indice del vettore $i_k$ tale che

$$|e_{i_k}^T \mathbf{x}^{(k)}| = \|\mathbf{x}^{(k)}\|_\infty = 1.$$

In questo caso $i_k$ può cambiare ad ogni passo, sarebbe improprio applicare la relazione (1.12), tuttavia è possibile dimostrare che la convergenza della successione $\lambda_1^{(k)}$ è garantita.
Se $\alpha_1$, componente di $\mathbf{x}^{(0)}$ su $\mathbf{v}_1$, è nulla e

$$|\lambda_2| > |\lambda_3|$$

allora il processo (1.12) dovrebbe, in teoria, convergere a $\lambda_2$. Tuttavia a causa degli errori di arrotondamento, accade sempre che $\alpha_1 \neq 0$ e dunque il

metodo converge di fatto a $\lambda_1$.
Se

$$|\lambda_2| \simeq |\lambda_1|$$

allora, avendo osservato che la convergenza della successione $\{\lambda_1^{(k)}\}$ dipende dal rapporto

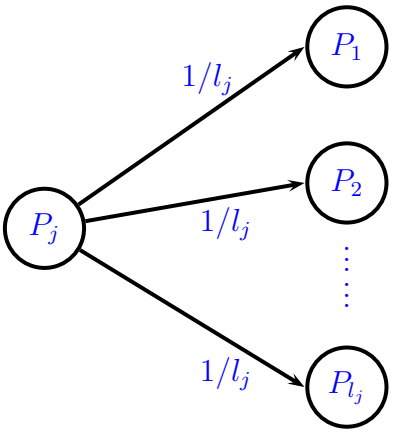
$$\frac{|\lambda_2|}{|\lambda_1|},$$

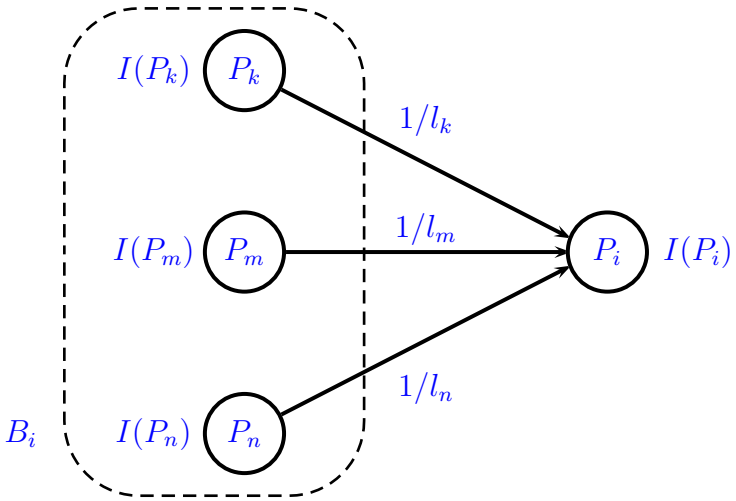
la velocità di convergenza risulterà particolarmente lenta.

### 1.2.1 L'algoritmo di PageRank di Google

Uno dei motivi per cui il metodo delle potenze è diventato particolarmente popolare è legato al fatto che costituisce il cuore di Google, il motore di ricerca più utilizzato nel web, ed in particolare al cosiddetto PageRank. Descriveremo brevemente in questo paragrafo le caratteristiche principali del PageRank ed il suo legame con il metodo delle potenze. Come è noto i motori di ricerca svolgono un'intensa, benchè invisibile, attività di monitoraggio delle pagine web, provvedendo ad indicizzare le parole più importanti che si trovano su un determinato documento e immagazzinando opportunamente tale informazione. Una volta che nel motore di ricerca viene inserita una determinata frase esso provvede a trovare tutte le pagine che riportano tale frase. Poichè il numero di tali pagine risulta essere particolarmente elevato è necessario che il motore di ricerca fornisca una misura dell'importanza di tali pagine in modo tale che queste siano visualizzate all'utente in base al valore di tale parametro. Lo scopo principale dell'algoritmo di PageRank è proprio quello di misurare il valore di tale parametro.
È chiaro che ogni pagina web contiene una serie di link ad altre pagine web, quindi l'idea alla base di PageRank è che l'importanza di una pagina dipende:
1) dalle pagine che hanno un link a tale pagina;
2) dall'importanza delle suddette pagine.
Il valore di PageRank viene valutato settimanalmente applicando l'algoritmo che ci apprestiamo a descrivere (aggiornamenti più frequenti sono impossibili a causa dell'enorme quantità di pagine pubblicate sul web). Consideriamo quindi una pagina web, che indichiamo con $P_j$, e che supponiamo abbia un

numero di link pari ad $l_j$. Se uno di questi link è diretto verso la pagina $P_i$ allora si può affermare che $P_i$ ha un'importanza rispetto a $P_j$ pari a $1/l_j$.



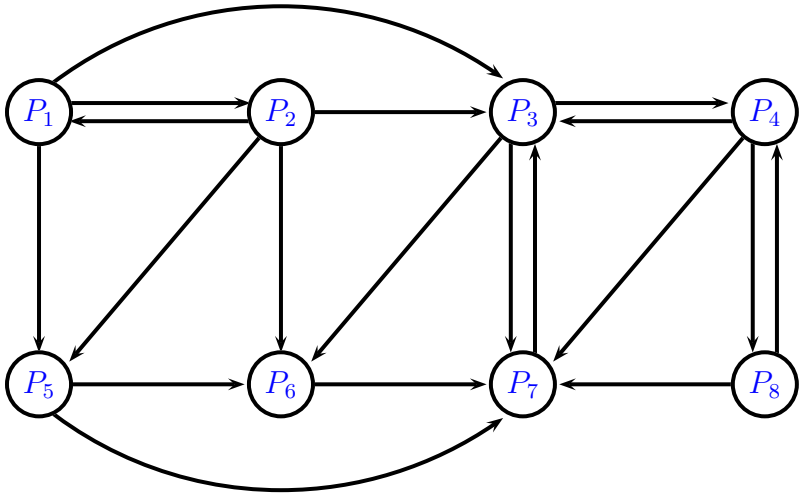

L'importanza di una pagina $P_i$, che indichiamo con $I(P_i)$, può essere calcolato come la somma di tutti i contributi delle pagine che hanno un link verso $P_i$. Se indichiamo con $B_i$ l'insieme delle pagine che hanno un link alla pagina $P_i$ allora possiamo definire la seguente formula:

$$I(P_i) = \sum_{P_j \in B_i} \frac{I(P_j)}{l_j}. \tag{1.13}$$

Chiaramente l'importanza di una pagina dipende dall'importanza delle pagine che consentono di arrivarci. Per vedere quale relazione lega tali quantità con il metodo delle potenze dobbiamo riformulare la definizione appena vista utilizzando una matrice $H$, detta Matrice di Hyperlink (Hyperlink Matrix), con elementi:

$$h_{ij} = \begin{cases} 1/l_j & \text{se } P_j \in B_i, \\ 0 & \text{altrimenti.} \end{cases}$$

La matrice $H$ ha tutti elementi nonnegativi ed inoltre la somma degli elementi che si trovano sulla stessa colonna è uguale a uno. Vediamo un esempio, considerando otto pagine web connesse nel seguente modo:




La Hyperlink matrix è la seguente:

$$H = \begin{bmatrix} 0 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/4 & 0 & 1/3 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1/3 & 0 & 0 & 0 & 0 & 1/2 \\ 1/3 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/4 & 1/3 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/2 & 1 & 0 & 1/2 \\ 0 & 0 & 0 & 1/3 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Per esempio risulta

$$I(P_1) = I(P_2)/4$$

oppure

$$I(P_3) = I(P_1)/3 + I(P_2)/4 + I(P_4)/3 + I(P_7).$$

In base alla relazione (1.13), detto $\mathbf{x}$ il vettore

$$\mathbf{x} = [I(P_1), I(P_2), \ldots, I(P_8)]^T$$

è verificata la seguente uguaglianza

$$\mathbf{x} = H\mathbf{x}$$

e quindi $\mathbf{x}$ è un autovettore della matrice di Hyperlink associato all'autovalore 1. Per calcolare il vettore $\mathbf{x}$ si può applicare il metodo delle potenze, in quanto è possibile verificare che 1 è proprio l'autovalore dominante di $H$, definendo il processo iterativo

$$\mathbf{x}^{(k+1)} = H\mathbf{x}^{(k)}.$$

Nella pratica, per ottenere la convergenza certa del metodo, si applica il metodo delle potenze non alla matrice di Hyperlink, ma alla cosiddetta Matrice di Google:

$$G = \alpha H + (1 - \alpha)\frac{U}{n}$$

dove $n$ è il numero delle pagine web, $U$ è una matrice quadrata di ordine $n$ i cui elementi sono tutti uguali a uno ed $\alpha$ è un numero casuale compreso tra 0 e 1.

# Capitolo 2

# Approssimazione ai minimi quadrati

## 2.1 Introduzione

Uno dei problemi più frequenti che si affronta nel campo ingegneristico è il seguente. Supponiamo di dover effettuare una misura di una grandezza incognita utilizzando uno strumento di misura e facendo variare una determinata grandezza (oppure al variare del tempo). Se è nota la legge che lega le due grandezze in gioco (per esempio esse potrebbero essere l'intensità di corrente e la differenza di potenziale di un circuito, oppure la pressione e la temperatura di un materiale durante un determinato esperimento) si può notare che raramente le misure da noi effettuate obbediscono a tale legge matematica. Questo perchè le misure effettuate con strumenti sono sempre soggette ad errori dovuti alle più varie cause (errore di parallasse dovuto ad un'errata posizione nella lettura dello strumento oppure a problemi interni allo strumento come l'effetto di campi elettromagnetici oppure ad una cattiva taratura del fondo scala). Sostanzialmente il problema è simile a quello dell'interpolazione cioè bisogna trovare una funzione $\Psi(x)$ che però approssimi le coppie di dati $(x_i, y_i)$ ma che non passi per questi poichè non sono dati precisi:

$$\Psi(x_i) \simeq y_i.$$

Il problema dell'approssimazione polinomiale prevede che la funzione $\Psi(x)$ sia un polinomio di grado opportuno (generalmente inferiore di molto rispetto al numero di dati disponibili e questa è un'ulteriore differenza rispetto

al problema dell'interpolazione). Un ulteriore grado di libertà sta nel fatto che si deve decidere come tradurre matematicamente il concetto di approssimazione, cioè come misurare la differenza tra i valori noti $y_i$ e quelli che li approssimano, cioè $\Psi(x_i)$. Vedremo che una delle tecniche più usate nel campo dell'approssimazione è la cosiddetta approssimazione polinomiale ai minimi quadrati. Prima di vedere di cosa si tratta introduciamo un importante algoritmo di algebra lineare che è la fattorizzazione $QR$ di matrici rettangolari reali e le proprietà principali delle matrici ortogonali.

## 2.2 Matrici Ortogonali

**Definizione 2.2.1** *Una matrice $Q \in \mathbb{R}^{n\times n}$ si dice ortogonale se:*

$$QQ^T = Q^TQ = I_n$$

*dove $Q^T$ è la matrice trasposta ed $I_n$ è la matrice identità di ordine $n$.*

Dalla definizione di matrice ortogonale segue come conseguenza immediata che l'inversa coincide con la matrice trasposta:

$$Q^{-1} = Q^T.$$

Se riscriviamo la matrice $Q$ per colonne

$$Q = [\mathbf{q}_1\ \mathbf{q}_2\ \mathbf{q}_3\ \ldots\ \mathbf{q}_n]$$

cioè in modo tale che $\mathbf{q}_i$ sia la $i-$esima colonna di $Q$, allora poichè $Q^TQ = I_n$ risulta

$$\begin{bmatrix} \mathbf{q}_1^T \\ \mathbf{q}_2^T \\ \vdots \\ \mathbf{q}_n^T \end{bmatrix} \begin{bmatrix} \mathbf{q}_1 & \mathbf{q}_2 & \mathbf{q}_3 & \ldots & \mathbf{q}_n \end{bmatrix} = \begin{bmatrix} \mathbf{q}_1^T\mathbf{q}_1 & \mathbf{q}_1^T\mathbf{q}_2 & \ldots & \mathbf{q}_1^T\mathbf{q}_n \\ \mathbf{q}_2^T\mathbf{q}_1 & \mathbf{q}_2^T\mathbf{q}_2 & \ldots & \mathbf{q}_2^T\mathbf{q}_n \\ \vdots & \vdots & & \vdots \\ \mathbf{q}_n^T\mathbf{q}_1 & \mathbf{q}_n^T\mathbf{q}_2 & \ldots & \mathbf{q}_n^T\mathbf{q}_n \end{bmatrix} = I_n$$

Quindi deve essere necessariamente

$$\mathbf{q}_i^T\mathbf{q}_j = (\mathbf{q}_j, \mathbf{q}_i) = \begin{cases} 1 & i = j \\ 0 & i \neq j. \end{cases}$$

Quindi una matrice ortogonale è caratterizzata dalla proprietà che tutti i suoi vettori colonna sono a due a due ortogonali ed ogni colonna ha norma euclidea uguale a 1. Una conseguenza di questa proprietà è che tutti gli elementi di una matrice ortogonale sono, in modulo, minori di 1. Infatti

$$\mathbf{q}_i^T \mathbf{q}_i = \sum_{j=1}^{n} q_{ji}^2 = 1.$$

Il discorso potrebbe essere ripetuto anche per le righe che risultano essere a due a due ortogonali e aventi norma euclidea uguale a 1.
Se $Q_1$ e $Q_2$ sono matrici ortogonali di ordine $n$ allora anche la matrice

$$Q = Q_1 Q_2$$

è ortogonale. Infatti

$$QQ^T = Q_1 Q_2 (Q_1 Q_2)^T = Q_1 Q_2 Q_2^T Q_1^T = Q_1 Q_1^T = I_n.$$

Si dice che l'insieme delle matrici è chiuso rispetto all'operazione di moltiplicazione, cioè forma un Gruppo. Se $Q_1$ è una matrice ortogonale di ordine $n$ e $Q_2$ è una matrice ortogonale di ordine $m$ allora anche le seguenti matrici di ordine $n+m$ sono ortogonali.

$$\begin{bmatrix} O & Q_1 \\ Q_2 & O \end{bmatrix} \qquad \begin{bmatrix} Q_1 & O \\ O & Q_2 \end{bmatrix}.$$

Se $Q$ è una matrice ortogonale di ordine $n$ ed $\mathbf{x} \in \mathbb{R}^n$ allora

$$\|Q\mathbf{x}\|_2 = \|\mathbf{x}\|_2.$$

Infatti

$$\|Q\mathbf{x}\|_2^2 = (Q\mathbf{x}, Q\mathbf{x}) = (Q\mathbf{x})^T Q\mathbf{x} = \mathbf{x}^T Q^T Q\mathbf{x} = \mathbf{x}^T\mathbf{x} = \|\mathbf{x}\|_2^2.$$

Tra le matrici ortogonali quelle che hanno una maggiore importanza nelle applicazioni sono le Matrici di Householder e le Matrici di Givens.

### Le Matrici di Householder

Una matrice di Householder $P$ di ordine $n$ è definita nel seguente modo

$$P = I - \beta \mathbf{u}\mathbf{u}^T, \qquad \beta \in \mathbb{R}, \mathbf{u} \in \mathbb{R}^n.$$

Appare evidente che una matrice di questo tipo è simmetrica ($P^T = P$), e imponiamo ora che sia anche ortogonale, cioè che

$$PP^T = (I - \beta \mathbf{u}\mathbf{u}^T)(I - \beta \mathbf{u}\mathbf{u}^T) = I$$

e quindi

$$\begin{aligned} PP^T &= I - 2\beta \mathbf{u}\mathbf{u}^T + \beta^2 \mathbf{u}(\mathbf{u}^T\mathbf{u})\mathbf{u}^T = I - 2\beta \mathbf{u}\mathbf{u}^T + \beta^2 \|\mathbf{u}\|_2^2 \mathbf{u}\mathbf{u}^T = \\ &= I + (\beta^2 \|\mathbf{u}\|_2^2 - 2\beta)\mathbf{u}\mathbf{u}^T = I + \beta(\beta \|\mathbf{u}\|_2^2 - 2)\mathbf{u}\mathbf{u}^T. \end{aligned}$$

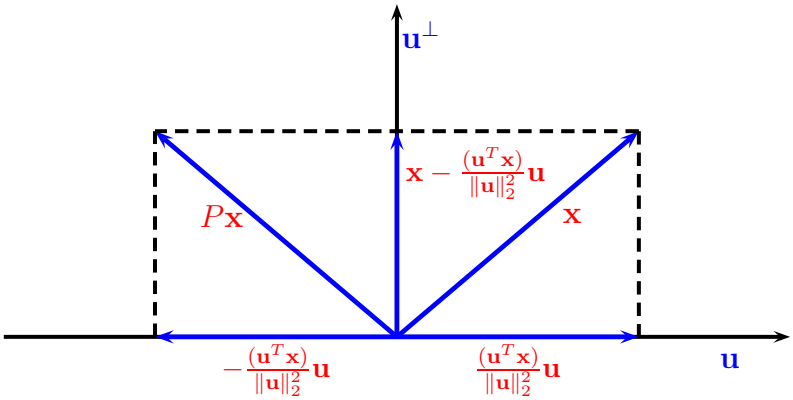
I valori di $\beta$ che rendono $P$ ortogonale sono due ma la soluzione $\beta = 0$ rende $P$ coincidente con la matrice identità, e quindi di scarso interesse pratico. Il valore accettabile per $\beta$ è

$$\beta = \frac{2}{\|\mathbf{u}\|_2^2}. \tag{2.1}$$

Le matrici di Householder sono dette anche matrici di riflessione poichè il vettore $P\mathbf{x}$ risulta essere il vettore riflesso di $\mathbf{x}$ rispetto allo spazio vettoriale perpendicolare al vettore $\mathbf{u}$, infatti

$$P\mathbf{x} = \mathbf{x} - \frac{\mathbf{u}^T\mathbf{x}}{\|\mathbf{u}\|_2^2}\mathbf{u} - \frac{\mathbf{u}^T\mathbf{x}}{\|\mathbf{u}\|_2^2}\mathbf{u}$$

e, come rappresentando tali vettori nella seguente figura, risulta chiara la relazione tra $\mathbf{x}$ e $P\mathbf{x}$.



### Le Matrici di Givens

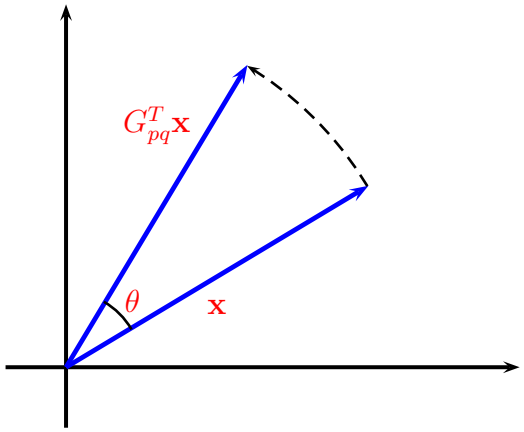
Fissati due numeri interi $p, q$ compresi tra 1 ed $n$ e tali che $p < q$ ed un angolo $\theta$, poniamo

$$c = \cos\theta, \qquad s = \sin\theta$$

e definiamo la matrice di Givens $G_{pq}$ nel seguente modo:

$$G_{pq} = \begin{array}{|ccccccc|l} \multicolumn{7}{l}{\left[\begin{array}{ccccccc} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ \hline & & c & & & s & \\ \hline & & & 1 & & & \\ & & & & \ddots & & \\ & & & & & 1 & \\ \hline & & -s & & & c & \\ \hline & & & & & & 1 \\ & & & & & & & \ddots \\ & & & & & & & & 1 \end{array}\right.} & \begin{array}{l} \\ \\ \\ \text{riga } p \\ \\ \\ \\ \text{riga } q \\ \\ \\ \\ \end{array} \end{array} \tag{2.2}$$

Solo le righe (e le colonne) di indice $p$ e $q$ differiscono dalla matrice identità. La verifica dell'ortogonalità di $G_{pq}$ è piuttosto immediata.
Le matrici di Givens sono dette anche matrici di rotazione poichè il vettore $G_{pq}^T\mathbf{x}$ risulta essere il risultato della rotazione di $\mathbf{x}$ in senso antiorario di un angolo pari a $\theta$.



## 2.3 Fattorizzazione QR

Assegnata una matrice $A \in \mathbb{R}^{m\times n}$, $m \geq n$, ci poniamo il problema di determinare una matrice $Q \in \mathbb{R}^{m\times m}$ ed una matrice $R \in \mathbb{R}^{m\times n}$ tali che:

$$A = QR \tag{2.3}$$

con $Q$ matrice ortogonale, cioè $Q^TQ = I$, mentre $R$ ha la seguente struttura

$$R = \begin{bmatrix} R_1 \\ \mathrm{O} \end{bmatrix},$$

dove $R_1 \in \mathbb{R}^{n\times n}$ è una matrice triangolare superiore. La fattorizzazione (2.3) prende il nome di fattorizzazione $QR$ (oppure fattorizzazione di Householder) della matrice $A$.
Se $m = n$ e $\det A \neq 0$ allora si può risolvere il sistema $A\mathbf{x} = \mathbf{b}$ nel modo seguente:

$$A\mathbf{x} = \mathbf{b} \quad \Leftrightarrow \quad QR\mathbf{x} = \mathbf{b} \quad \Leftrightarrow \quad R\mathbf{x} = \mathbf{c} \qquad \text{con } \mathbf{c} = Q^T\mathbf{b}$$

e dunque per calcolare il vettore $\mathbf{x}$ basta risolvere un sistema triangolare superiore. Vedremo successivamente che tale fattorizzazione risolve brillantemente il problema della risoluzione del sistema $A\mathbf{x} = \mathbf{b}$, $A \in \mathbb{R}^{m\times n}$, $m > n$.
Consideriamo preliminarmente il seguente problema: assegnato un vettore $\mathbf{x} \in \mathbb{R}^m$, $\mathbf{x} \neq 0$, determinare una matrice di Householder $P$ tale che

$$P\mathbf{x} = k\mathbf{e}_1 \qquad k \in \mathbb{R},$$

dove $\mathbf{e}_1$ è il primo versore fondamentale di $\mathbb{R}^m$. Quindi

$$P\mathbf{x} = (I - \beta\mathbf{u}\mathbf{u}^T)\mathbf{x} = \mathbf{x} - \beta\mathbf{u}\mathbf{u}^T\mathbf{x} = \mathbf{x} - \beta(\mathbf{u}^T\mathbf{x})\mathbf{u}.$$

Poichè il vettore $\mathbf{u}$, è arbitrario possiamo imporre

$$\beta\mathbf{u}^T\mathbf{x} = 1 \tag{2.4}$$

cosicchè

$$\mathbf{x} - \mathbf{u} = k\mathbf{e}_1$$

quindi deve essere necessariamente $u_i = x_i$, $i = 2, \ldots, m$, e $k = x_1 - u_1$. Determiniamo ora l'espressione dell'elemento $u_1$. Da (2.1) e (2.4) abbiamo:

$$\frac{2}{\|\mathbf{u}\|_2^2}\mathbf{u}^T\mathbf{x} = 1$$

$$\|\mathbf{u}\|_2^2 - 2\mathbf{u}^T\mathbf{x} = 0$$

$$u_1^2 + \sum_{i=2}^{m} x_i^2 - 2u_1x_1 - 2\sum_{i=2}^{m} x_i^2 = 0$$

$$u_1^2 - 2x_1u_1 - \sum_{i=2}^{m} x_i^2 = 0.$$

Risolvendo l'equazione di secondo grado si ottengono due soluzioni:

$$u_1 = x_1 \pm \sqrt{x_1^2 + \sum_{i=2}^{m} x_i^2} = x_1 \pm \|\mathbf{x}\|_2.$$

Per evitare la cancellazione di cifre significative (vedere paragrafo 2.3.1) si sceglie il segno in modo tale da evitare l'operazione di sottrazione, cioè

$$u_1 = x_1 + segn(x_1)\|\mathbf{x}\|_2.$$

La (2.1) può anche scriversi:

$$\begin{aligned}\beta &= \frac{2}{\|\mathbf{u}\|_2^2} = \frac{2}{u_1^2 + \sum_{i=2}^{m} x_i^2} = \frac{2}{(x_1 + segn(x_1)\|\mathbf{x}\|_2)^2 + \sum_{i=2}^{m} x_i^2} \\ &= \frac{2}{x_1^2 + \|\mathbf{x}\|_2^2 + 2|x_1|\|\mathbf{x}\|_2 + \sum_{i=2}^{m} x_i^2} \\ &= \frac{2}{2\|\mathbf{x}\|_2^2 + 2|x_1|\|\mathbf{x}\|_2} = \frac{1}{\|\mathbf{x}\|_2(|x_1| + \|\mathbf{x}\|_2)}.\end{aligned}$$

In definitiva scegliendo

$$\beta = \frac{1}{\|\mathbf{x}\|_2(|x_1| + \|\mathbf{x}\|_2)} \qquad \text{e} \qquad \mathbf{u} = \begin{bmatrix} x_1 + segn(x_1)\|\mathbf{x}\|_2 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

risulta

$$P\mathbf{x} = (I - \beta\mathbf{u}\mathbf{u}^T)\mathbf{x} = -segn(x_1)\|\mathbf{x}\|_2\mathbf{e}_1.$$

Sfruttiamo la proprieta appena trovata per calcolare la fattorizzazione $QR$ della matrice rettangolare $A$. Per semplicità consideriamo il caso in cui $A \in \mathbb{R}^{7\times 5}$ :

$$A = \begin{bmatrix} \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \\ \times & \times & \times & \times & \times \end{bmatrix}.$$

Si conseidera come vettore $\mathbf{x}$ la prima colonna della matrice (evidenziata in rosso) e si trova la matrice di Householder $P_1$ che rende nulli tutti gli elementi tranne il primo. Moliplicando $P_1$ per $A$ l'effetto è quello di aver reso nulli tutti gli elementi sottodiagonali della prima colonna di $A$:

$$P_1A = \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \otimes & \times & \times & \times \\ 0 & \otimes & \times & \times & \times \\ 0 & \otimes & \times & \times & \times \\ 0 & \otimes & \times & \times & \times \\ 0 & \otimes & \times & \times & \times \\ 0 & \otimes & \times & \times & \times \end{bmatrix}.$$

A questo punto si considera come vettore $\mathbf{x}$ quello composto da tutti gli elementi evidenziati con $\otimes$ e si determina la matrice di Householder $\widehat{P}_2$ che azzera tutti tali elementi, tranne il primo. Poichè tale matrice ha ordine 6 allora per applicarla a $P_1A$ si definisce la matrice

$$P_2 = \begin{bmatrix} 1 & \mathbf{o} \\ \mathbf{o}^T & \widehat{P}_2 \end{bmatrix}$$

cosicchè

$$P_2P_1A = \begin{bmatrix} \times & \times & \times & \times & \times \\ 0 & \times & \times & \times & \times \\ 0 & 0 & \otimes & \times & \times \\ 0 & 0 & \otimes & \times & \times \\ 0 & 0 & \otimes & \times & \times \\ 0 & 0 & \otimes & \times & \times \\ 0 & 0 & \otimes & \times & \times \end{bmatrix}.$$

Se $A \in \mathbb{R}^{m\times n}$ allora al $k-$esimo passo si definisce la matrice

$$P_k = \begin{bmatrix} I_{k-1} & O \\ O^T & \widehat{P}_k \end{bmatrix}$$

con $\widehat{P}_k$ matrice di Householder di ordine $m-k+1$ e $I_{k-1}$ matrice identità di ordine $k-1$. È facile osservare che le matrici $P_k$ sono ortogonali per ogni $k$, in conseguenza di una proprietà vista in precedenza. Se $m > n$ si devono applicare $n$ matrici $P_k$ cosicchè

$$R = P_nP_{n-1}\dots P_2P_1A$$

è triangolare superiore. La matrice $A$ può essere scritta come

$$A = (P_nP_{n-1}\dots P_1)^{-1}\,R = \left(P_1^{-1}P_2^{-1}\dots P_n^{-1}\right)R = P_1P_2\dots P_nR,$$

poichè tutte le matrici $P_k$ sono ortogonali e simmetriche (quindi $P_k^{-1} = P_k$). Posto

$$Q = P_1P_2\dots P_{n-1}P_n,$$

tale matrice è ortogonale in quanto prodotto di $n$ matrici ortogonali, quindi $A$ può essere fattorizzata come

$$A = QR.$$

Osserviamo che ad ogni passo la matrice $P_k$ moltiplica la matrice al passo precedente che è già stata parzialmente triangolarizzata. Osserviamo che il numero di operazioni algebriche necessarie a calcolare tali colonne non è molto elevato. Infatti se $P$ è una matrice di Householder ed $y$ è un vettore diverso da $\mathbf{u}$ allora

$$P\mathbf{y} = \mathbf{y} - (\beta\mathbf{u}^T\mathbf{y})\mathbf{u}.$$

Per calcolare $\mathbf{u}^T\mathbf{y}$ sono necessarie $2m$ operazioni aritmetiche ($\beta$ si suppone che sia già stato calcolato), poi sono necessari $m$ prodotti per calcolare $(\beta\mathbf{u}^T\mathbf{y})\mathbf{u}$ e poi $m$ differenze. In tutto sono richieste circa $4m$ operazioni mentre per calcolare il prodotto tra una matrice di ordine $m$ ed un vettore di ordine $m$ sono necessarie $2m^2$ operazioni aritmetiche. Ricordiamo che il costo computazionale di un algoritmo (cioè il numero di operazioni che richiede) è uno dei parametri da considerare quando si valuta l'efficienza di un algoritmo. L'algoritmo codificato in MatLab è il seguente ($A$ è una matrice di $m \times n$):

```
function [Q,R] = myqr(A)
%
% [Q,R] = myqr(A)
%
% Parametri di input
% A = matrice rettangolare di dimension m x n
%
% Parametri di output
% Q = matrice ortogonale m x m
% R = matrice triangolare superiore m x n
%
[m,n] = size(A)
Q = eye(m)
for k=1:n
     u = A(k:m,k);
     norma = norm(u,2);
     u(1) = u(1)+sign(u(1))*norma;
     beta = 2/norm(u,2)^2;
     U = eye(m-k+1)-beta*u*u';
     P = [eye(k-1) zeros(k-1,m-k+1); zeros(m-k+1,k-1) U]
     A = P*A;
     Q = Q*P;
end
return
```

### 2.3.1 La cancellazione di cifre significative

Ricordiamo che se $x$ è un numero reale, la sua rappresentazione macchina, cioè il numero che approssima $x$ nella memoria di un elaboratore, e che spesso viene indicata con $\mathrm{fl}(x)$, è un numero che è affetto da un certo errore (dipendente da $x$), tale che vale la seguente relazione:

$$\mathrm{fl}(x) = x(1 + \varepsilon_x)$$

dove $\varepsilon_x$ rappresenta l'errore relativo ed è maggiorato, in modulo, dalla precisione macchina $u$ (che dipende chiaramente dalla memoria dell'elaboratore e rappresenta il più piccolo numero reale rappresentabile). Chiaramente l'errore di rappresentazione di un numero reale può influire nella precisione dei risultati forniti dall'elaboratore poichè esso si propaga nei calcoli ed i calcoli stessi tendono ad introdurre ulteriori errori di rappresentazione. In questo paragrafo sarà affrontato un caso particolare legato all'operazione della differenza tra numeri reali, come effettuata all'interno dell'elaboratore. Supponiamo quindi di voler calcolare la differenza tra due numeri reali $a$ e $b$. Siano $fl(a)$ e $fl(b)$ rispettivamente le loro rappresentazioni di macchina. Vogliamo vedere quale è l'errore che viene commesso dall'elaboratore quando calcola $a - b$.

$$\begin{aligned} fl(a) \ominus fl(b) &= [fl(a) - fl(b)](1 + \varepsilon_1) = [a(1 + \varepsilon_2) - b(1 + \varepsilon_3)](1 + \varepsilon_1) \\ &= (a + a\varepsilon_2 - b - b\varepsilon_3)(1 + \varepsilon_1) \\ &= (a - b) + (a - b)\varepsilon_1 + a\varepsilon_2 - b\varepsilon_3 + a\varepsilon_1\varepsilon_2 - b\varepsilon_1\varepsilon_3. \end{aligned}$$

Una maggiorazione per l'errore relativo è la seguente

$$\begin{aligned} \frac{|(fl(a) \ominus fl(b)) - (a - b)|}{|a - b|} \leq \; & |\varepsilon_1| + \frac{|a|}{|a - b|}(|\varepsilon_2| + |\varepsilon_1||\varepsilon_2|) + \\ & + \frac{|b|}{|a - b|}(|\varepsilon_3| + |\varepsilon_1||\varepsilon_3|). \end{aligned} \tag{2.5}$$

Se $a$ e $b$ hanno segno opposto risulta

$$\max(|a|, |b|) \leq |a - b|$$

e dalla (2.5) segue la maggiorazione

$$\frac{|(fl(a) \ominus fl(b)) - (a-b)|}{|a-b|} \leq 3u + O(u^2)$$

dove $u$ è la precisione di macchina.
Se $a$ e $b$ hanno lo stesso segno allora l'errore relativo può essere molto grande quanto più $a$ e $b$ sono vicini. Questo fenomeno prende il nome di cancellazione di cifre significative. Questa analisi indica che, nell'effettuare operazioni aritmetiche sarebbe opportuno evitare la differenza tra due numeri reali aventi lo stesso segno e vicini tra loro.

### 2.3.2 Calcolo della fattorizzazione $QR$ con le matrici di Givens

Sia $\mathbf{x} \in \mathbb{R}^m$, ci poniamo il problema di determinare una matrice di Givens $G_{pq} \in \mathbb{R}^{m \times m}$, con $p, q$ numeri interi tali che $1 \leq p < q \leq m$, e tale che il vettore $\mathbf{y}$ definito come

$$\mathbf{y} = G_{pq}\mathbf{x}$$

abbia la componente di indice $q$ uguale a zero. Ovviamente supponiamo che $x_q \neq 0$ altrimenti potremmo prendere $G_{pq}$ uguale alla matrice identità di ordine $m$. Calcolando il prodotto $G_{pq}\mathbf{x}$ si ottiene

$$\begin{bmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & c & & & s & & \\ & & & & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & & \\ & & & -s & & & c & & \\ & & & & & & & 1 & \\ & & & & & & & & \ddots \\ & & & & & & & & & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{p-1} \\ x_p \\ x_{p+1} \\ \vdots \\ x_{q-1} \\ x_q \\ x_{q+1} \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} x_1 \\ \vdots \\ x_{p-1} \\ cx_p + sx_q \\ x_{p+1} \\ \vdots \\ x_{q-1} \\ -sx_p + cx_q \\ x_{q+1} \\ \vdots \\ x_m \end{bmatrix}.$$

Uguagliando le componenti al vettore $\mathbf{y}$ risulta:

$$\begin{aligned} y_i &= x_i, \qquad\qquad i \neq p, q \\ y_p &= cx_p + sx_q \\ y_q &= -sx_p + cx_q. \end{aligned}$$

Il problema da risolvere è quello di trovare i valori di $c, s$ $(c^2 + s^2 = 1)$ tali che

$$\begin{cases} cx_p + sx_q = y_p \\ -sx_p + cx_q = y_q = 0. \end{cases}$$

Prima di risolvere il sistema troviamo il valore di $y_p$. Ricordiamo che poichè la matrice $G_{pq}$ è ortogonale allora i vettori $\mathbf{x}$ e $\mathbf{y}$ hanno la stessa norma 2:

$$\|\mathbf{y}\|_2^2 = \|G_{pq}\mathbf{x}\|_2^2 = \|\mathbf{x}\|_2^2.$$

Quindi

$$\sum_{i=1}^{m} y_i^2 = \sum_{i=1, i\neq p,q}^{m} y_i^2 + y_p^2 + y_q^2 = \sum_{i=1}^{m} x_i^2,$$

e poichè $x_i = y_i$ se $i \neq p, q$, e $y_q = 0$ allora

$$\sum_{i=1, i\neq p,q}^{m} x_i^2 + y_p^2 = \sum_{i=1, i\neq p,q}^{m} x_i^2 + x_p^2 + x_q^2.$$

Da cui si ottiene

$$y_p^2 = x_p^2 + x_q^2 \qquad \Rightarrow \qquad y_p = \sqrt{x_p^2 + x_q^2}$$

Il sistema da risolvere diviene

$$\begin{cases} cx_p + sx_q = \sqrt{x_p^2 + x_q^2} \\ cx_q - sx_p = 0. \end{cases}$$

Applicando la regola di Cramer si trova

$$c = \frac{\begin{vmatrix} \sqrt{x_p^2 + x_q^2} & x_q \\ 0 & -x_p \end{vmatrix}}{\begin{vmatrix} x_p & x_q \\ x_q & -x_p \end{vmatrix}} = \frac{-x_p\sqrt{x_p^2 + x_q^2}}{-x_p^2 - x_q^2} = \frac{x_p\sqrt{x_p^2 + x_q^2}}{x_p^2 + x_q^2} = \frac{x_p}{\sqrt{x_p^2 + x_q^2}}.$$

$$s = \frac{\begin{vmatrix} x_p & \sqrt{x_p^2 + x_q^2} \\ x_q & 0 \end{vmatrix}}{\begin{vmatrix} x_p & x_q \\ x_q & -x_p \end{vmatrix}} = \frac{-x_q\sqrt{x_p^2 + x_q^2}}{-x_p^2 - x_q^2} = \frac{x_q\sqrt{x_p^2 + x_q^2}}{x_p^2 + x_q^2} = \frac{x_q}{\sqrt{x_p^2 + x_q^2}}.$$

Le matrici di Givens possono essere usate per calcolare la fattorizzazione $QR$ di una matrice $A$ rettagolare utilizzandole per azzerare gli elementi della parte triangolare inferiore. Se $A$ è una matrice $m \times n$ allora possiamo definire la matrice di Givens $G_{1m}$ che annulla l'elemento di $A$ di posto $(m, 1)$, quindi definire la matrice $G_{1,m-1}$ che annulla l'elemento della nuova matrice di posto $(m-1, 1)$ e così via fino alla matrice $G_{1,2}$. Per gli elementi delle successive colonne si procede allo stesso modo, finchè la matrice non è completamente triangolare. La matrice triangolare (se $m \geq n$) $R$ è uguale a:

$$R = G_{n,n+1}G_{n,n+2}\ldots G_{1,m-1}G_{1m}A$$

quindi

$$A = (G_{n,n+1}G_{n,n+2}\ldots G_{1,m-1}G_{1m})^{-1}\, R = QR$$

con

$$Q = (G_{n,m+1}G_{n,m+2}\ldots G_{1,m-1}G_{1m})^{-1}\,.$$

Come ultima osservazione si deve dire che, per calcolare la fattorizzazione $QR$, conviene utilizzare le matrici di Householder se la matrice $A$ è piena, cioè molti suoi elementi sono diversi da zero, in quanto consente di annullare, in un singolo passo, tutti gli elementi appartenenti ad una singola porzione di colonna. Conviene utilizzare le matrici di Givens quando $A$ è sparsa, cioè ha già una buona parte degli elementi uguali a zero, poichè consente di azzerare in modo selettivo tali elementi.

## 2.4 La Retta di Regressione

Come si è già accennato nell'introduzione di questo Capitolo quando i dati $(x_i, y_i)$, $i = 0, \ldots, n$, sono rilevati con scarsa precisione, non ha molto senso cercare un polinomio di grado $n$ (o, più in generale una funzione $\Psi(x)$) che interpoli i valori $y_i$ nei nodi $x_i$. Può invece risultare più proficuo cercare un polinomio di grado $m < n$ che si avvicini il più possibile ai dati rilevati. I

criteri che si possono scegliere per dare un senso alla frase si avvicini il più possibile possono essere diversi, per esempio si potrebbe cercare il polinomio $p_m(x)$ tale che

$$\max_i |p_m(x_i) - y_i| = \min_{p \in \Pi_m} \max_i |p(x_i) - y_i|$$

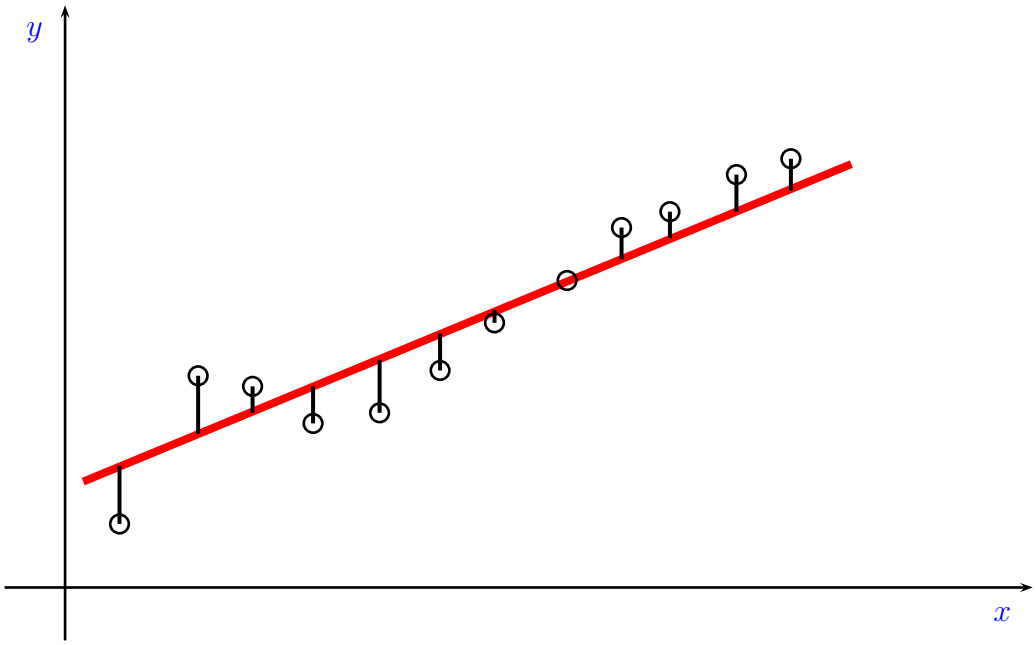
avendo indicato con $\Pi_m$ l'insieme dei polinomi di grado al più $m$. Il tipo di approssimazione più utilizzato (e anche quello più semplice da risolvere) è quello polinomiale ai minimi quadrati. Un caso particolare è quello in cui si cerca una funzione lineare che approssima nel modo migliore i dati sperimentali. Quindi si pone

$$\Phi(x) = \alpha x + \beta, \qquad \alpha, \beta \in \mathbb{R} \tag{2.6}$$

e si cercano, tra tutte le possibili rette, i coefficienti $\alpha$ e $\beta$ che globalmente minimizzano la differenza

$$\Phi(x_i) - y_i = \alpha x_i + \beta - y_i$$

La retta (2.6) che risolve tale problema viene detta Retta di regressione. Nella seguente figura sono evidenziate le quantità che devono essere globalmente minimizzate (i punti $(x_i, y_i)$ sono evidenziati con il simbolo $\circ$).



Un modo per minimizzare globalmente le distanze della retta dalle approssimazioni è quello di trovare i valori $\alpha, \beta$ che minimizzano la funzione:

$$\Psi(\alpha, \beta) = \sum_{i=0}^{n} (\alpha x_i + \beta - y_i)^2 .$$

Per questo si parla di problema ai minimi quadrati (si minimizza una somma di quantità elevate al quadrato).
Per determinare tali valori calcoliamo le derivate parziali rispetto alle incognite:

$$\frac{\partial \Psi}{\partial \alpha} = 2 \sum_{i=0}^{n} x_i (\alpha x_i + \beta - y_i)$$

$$\frac{\partial \Psi}{\partial \beta} = 2 \sum_{i=0}^{n} (\alpha x_i + \beta - y_i)$$

$$\begin{cases} \dfrac{\partial \Psi}{\partial \alpha} = 2 \displaystyle\sum_{i=0}^{n} x_i (\alpha x_i + \beta - y_i) = 0 \\ \dfrac{\partial \Psi}{\partial \beta} = 2 \displaystyle\sum_{i=0}^{n} (\alpha x_i + \beta - y_i) = 0 \end{cases}$$

$$\begin{cases} \displaystyle\sum_{i=0}^{n} x_i (\alpha x_i + \beta - y_i) = 0 \\ \displaystyle\sum_{i=0}^{n} (\alpha x_i + \beta - y_i) = 0 \end{cases}$$

$$\begin{cases} \alpha \displaystyle\sum_{i=0}^{n} x_i^2 + \beta \sum_{i=0}^{n} x_i - \sum_{i=0}^{n} x_i y_i = 0 \\ \alpha \displaystyle\sum_{i=0}^{n} x_i + (n+1)\beta - \sum_{i=0}^{n} y_i = 0. \end{cases}$$

Poniamo per semplicità

$$S_{xx} = \sum_{i=0}^{n} x_i^2 \qquad S_x = \sum_{i=0}^{n} x_i$$

$$S_{xy} = \sum_{i=0}^{n} x_i y_i \qquad S_y = \sum_{i=0}^{n} y_i.$$

Il sistema diventa

$$\begin{cases} S_{xx}\alpha + S_x\beta = S_{xy} \\ \\ S_x\alpha + (n+1)\beta = S_y \end{cases}$$

la cui soluzione è

$$\alpha = \frac{(n+1)S_{xy} - S_x S_y}{(n+1)S_{xx} - S_x^2}$$

$$\beta = \frac{S_y S_{xx} - S_x S_{xy}}{(n+1)S_{xx} - S_x^2}.$$

## 2.5 Approssimazione Polinomiale ai Minimi Quadrati

Esattamente nello stesso modo descritto nel precedente paragrafo si può cercare il polinomio di grado $m$ che si avvicina ai dati sperimentali nel senso dei minimi quadrati, cioè si cerca il polinomio

$$p_m(x) = \sum_{j=0}^{m} a_j x^j,$$

di grado $m < n$, tale che:

$$\sum_{i=0}^{n} (p_m(x_i) - y_i)^2$$

sia minima. Cioè si vuole risolvere il problema di minimo:

$$\min_{a_0, a_1, \ldots, a_m} \sum_{i=0}^{n} \left( \sum_{j=0}^{m} a_j x_i^j - y_i \right)^2. \tag{2.7}$$

Inseriamo in un vettore i valori del polinomio $p_m(x)$ calcolati nelle ascisse $x_i$:

$$\begin{bmatrix} p_m(x_0) \\ p_m(x_1) \\ \vdots \\ p_m(x_n) \end{bmatrix} = \begin{bmatrix} a_0 + a_1x_0 + a_2x_0^2 + \cdots + a_{m-1}x_0^{m-1} + a_mx_0^m \\ a_0 + a_1x_1 + a_2x_1^2 + \cdots + a_{m-1}x_1^{m-1} + a_mx_1^m \\ \vdots \\ a_0 + a_1x_n + a_2x_n^2 + \cdots + a_{m-1}x_n^{m-1} + a_mx_n^m \end{bmatrix}$$

Definendo la matrice $B \in \mathbb{R}^{(n+1)\times(m+1)}$ nel seguente modo:

$$B = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{m-1} & x_0^m \\ 1 & x_1 & x_1^2 & \dots & x_1^{m-1} & x_1^m \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^{m-1} & x_n^m \end{bmatrix},$$

ed il vettore $\mathbf{a} = [a_0, a_1, \dots, a_m]^T$ si vede che risulta

$$\begin{bmatrix} p_m(x_0) \\ p_m(x_1) \\ \vdots \\ p_m(x_n) \end{bmatrix} = B\mathbf{a}$$

e quindi, definito il vettore $\mathbf{y} = [y_0, y_1, \dots, y_n]^T$ risulta

$$\begin{bmatrix} p_m(x_0) - y_0 \\ p_m(x_1) - y_1 \\ \vdots \\ p_m(x_n) - y_n \end{bmatrix} = B\mathbf{a} - \mathbf{y}.$$

Osserviamo che la quantità da minimizzare

$$\sum_{i=0}^{n} (p_m(x_i) - y_i)^2$$

corrisponde alla norma 2 al quadrato del vettore che ha componenti pari a $p_m(x_i) - y_i$ quindi il problema di minimo può essere riscritto in forma algebrica:

$$\min_{\mathbf{a}\in\mathbb{R}^{m+1}} \|B\mathbf{a} - \mathbf{y}\|_2^2. \tag{2.8}$$

Consideriamo quindi la fattorizzazione $QR$ della matrice $B$:

$$B = QR, \qquad Q \in \mathbb{R}^{(n+1)\times(n+1)}, R \in \mathbb{R}^{(n+1)\times(m+1)}$$

essendo $Q$ una matrice ortogonale ed $R$ con la seguente struttura:

$$R = \begin{bmatrix} R_1 \\ \mathrm{O} \end{bmatrix}$$

dove $R_1$ è una matrice triangolare superiore di ordine $m+1$. Sostituendo a $B$ la sua fattorizzazione $QR$ abbiamo:

$$\begin{aligned} \|B\mathbf{a} - \mathbf{y}\|_2^2 &= \|QR\mathbf{a} - \mathbf{y}\|_2^2 = \|Q(R\mathbf{a} - Q^T\mathbf{y})\|_2^2 = \\ &= \|R\mathbf{a} - \mathbf{c}\|_2^2 \end{aligned}$$

avendo posto $\mathbf{c} = Q^T\mathbf{y}$.

$$\begin{aligned} \|B\mathbf{a} - \mathbf{y}\|_2^2 &= \left\| \begin{bmatrix} R_1 \\ \mathrm{O} \end{bmatrix} \mathbf{a} - \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \end{bmatrix} \right\|_2^2 = \\ &= \left\| \begin{bmatrix} R_1\mathbf{a} - \mathbf{c}_1 \\ -\mathbf{c}_2 \end{bmatrix} \right\|_2^2 = \|R_1\mathbf{a} - \mathbf{c}_1\|_2^2 + \|\mathbf{c}_2\|_2^2. \end{aligned}$$

Poichà $\mathbf{c}_2$ è un vettore che non dipende da $\mathbf{a}$ il problema di minimo si riduce a minimizzare $\|R_1\mathbf{a} - \mathbf{c}_1\|_2$ e poichè $R_1$ è una matrice non singolare se $\mathbf{a}$ è il vettore che risolve il sistema

$$R_1\mathbf{a} = \mathbf{c}_1$$

allora tale minimo viene raggiunto. Si noti che questo algoritmo fornisce anche la misura del minimo che è pari a $\|\mathbf{c}_2\|_2$.
Per risolvere il problema dell'approssimazione polinomiale ai minimi quadrati si devono effettuare i seguenti passi:
1. Calcolare la fattorizzazione $B = QR$;
2. Calcolare il vettore $\mathbf{c} = Q^T\mathbf{y}$;

3. Risolvere il sistema triangolare superiore $R_1\mathbf{a} = \mathbf{c}_1$.

In questo paragrafo riportiamo la codifica MatLab dell'algoritmo che serve per risolvere il problema dell'approssimazione polinomiale ai minimi quadrati. La seguente routine riceve in input i due vettori contenenti rispettivamente le ascisse e le ordinate dei dati del problema ed il grado $m$ del polinomio approssimante che si vuole ottenere. In output vengono calcolati il vettore, di dimensione $m + 1$, dei coefficienti di tale polinomio in ordine descrescente (cioè `a(1)` è il coefficiente della potenza più elevata, cioè $x^m$, ed il valore del cosiddetto residuo, cioè il valore del minimo calcolato (si veda (2.8)).

```
function [a,r] = LeastSquares(x,y,m)
%
% Funzione per il calcolo dell'approssimazione
% polinomiale ai minimi quadrati
% [a,r] = LeastSquares(x,y,m)
%
% x = vettore colonna delle n+1 ascisse
% y = vettore colonna delle n+1 misure
% m = grado dell'approssimazione
%
% a = vettore dei coefficienti del polinomio soluzione
% r = residuo calcolato
%
n = length(x);
a = zeros(m+1,1);
B = ones(n,m+1);
%
% Assegnazione della matrice B
%
for j=2:m+1
     B(:,j) = B(:,j-1).*x;
end
%
% Calcolo della fattorizzazione QR di B
%
[Q,R] = myqr(B);
%
% Calcolo del vettore c
```

```
%
c = Q'*y;
%
% Calcolo del vettore c2
%
c2 = c(m+2:n);
%
% Calcolo del vettore a
%
a(m+1) = c(m+1)/R(m+1,m+1);
for i=m:-1:1
     a(i) = (c(i)-R(i,i+1:m+1)*a(i+1:m+1))/R(i,i);
end
a = flipud(a);
%
% Calcolo del residuo
%
r=norm(c2,2);
return
```

**Esempio 2.5.1** *Supponiamo di voler calcolare i polinomi che approssima ai minimi quadrati i punti* $(-1,-1)$, $(0,1)$, $(1,-1)$, $(3,2)$ *e* $(5,6)$. *Nella Figure 2.1, 2.2 e 2.3 sono tracciati rispettivamente i polinomi di approssimazione ai minimi quadrati di grado rispettivamente 1, 2 e 3 mentre i piccoli cerchi* `'o'` *indicano i punti che intendiamo approssimare. È facile osservare come tali polinomi non passano, di solito, per i punti del problema.*

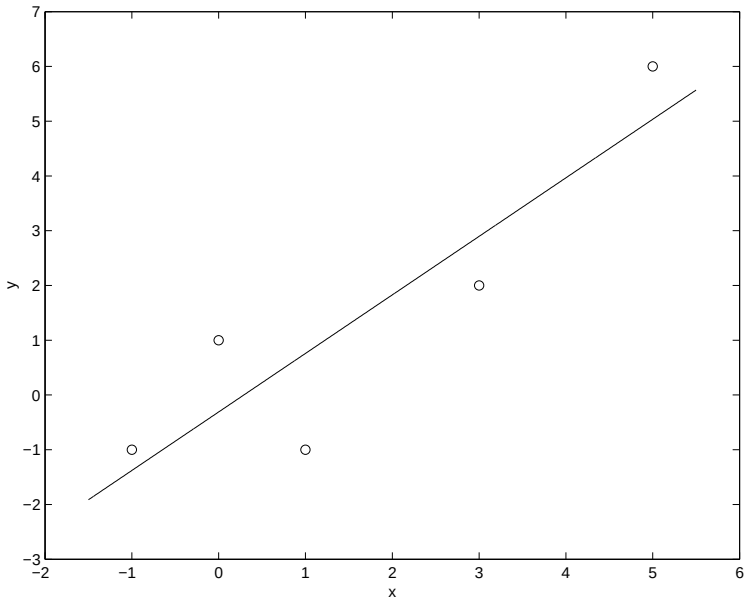



Figura 2.1: Approssimazione lineare ai minimi quadrati

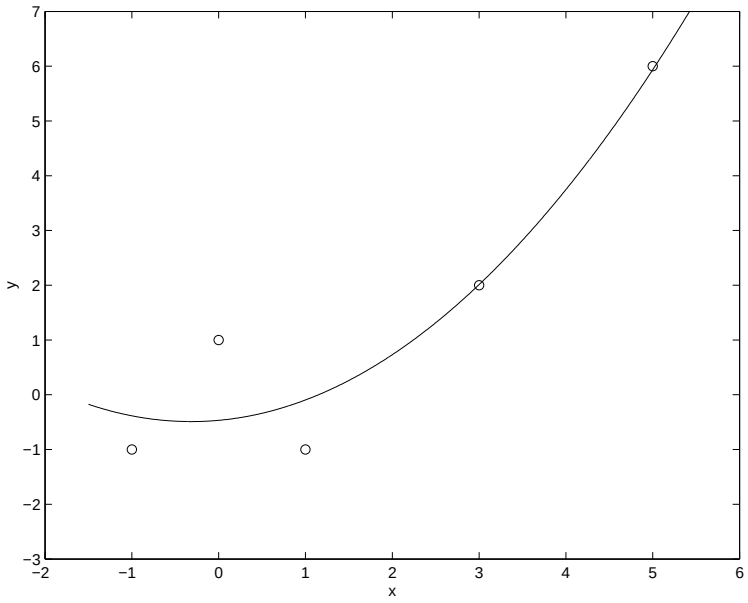



Figura 2.2: Approssimazione polinomiale di secondo grado ai minimi quadrati

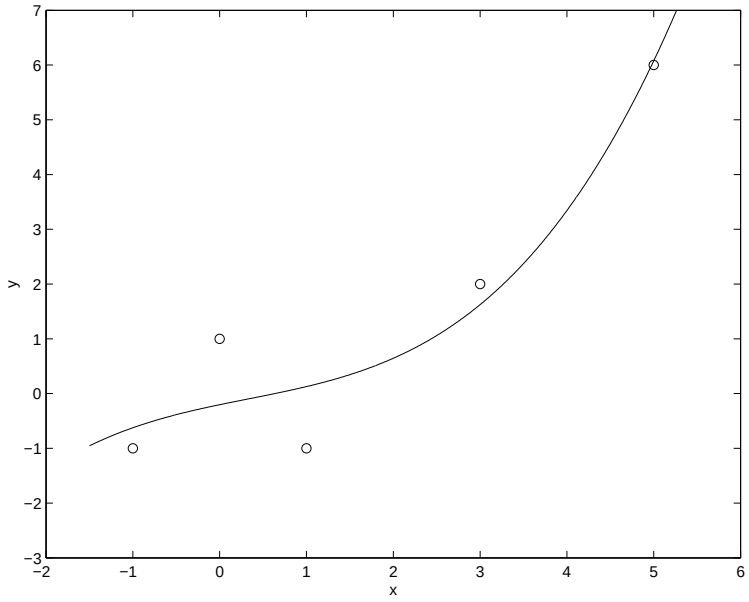



Figura 2.3: Approssimazione polinomiale di terzo grado ai minimi quadrati

# Capitolo 3

# La Decomposizione ai Valori Singolari

## 3.1 Valori singolari

**Definizione 3.1.1** *Se $A \in \mathbb{R}^{m\times n}$ allora esistono due matrici ortogonali $U, V$, con $U \in \mathbb{R}^{m\times m}$ e $V \in \mathbb{R}^{n\times n}$, ed una matrice diagonale $\Sigma \in \mathbb{R}^{m\times n}$, con elementi diagonali non negativi, tali che*

$$A = U\Sigma V^T. \tag{3.1}$$

Tale decomposizione viene appunto detta Decomposizione ai valori singolari. Gli elementi diagonali della matrice $\Sigma$ sono detti valori singolari. Se $m = n$ allora le tre matrici sono quadrate. Invece se $m > n$ allora

$$\Sigma = \begin{bmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_{n-1} \\ & & & & \sigma_n \\ \hline & & \mathrm{O} & \end{bmatrix},$$

mentre se $m < n$ allora risulta

$$\Sigma = \left[\begin{array}{ccccc|c} \sigma_1 & & & & & \\ & \sigma_2 & & & & \\ & & \ddots & & & \mathrm{O} \\ & & & \sigma_{m-1} & & \\ & & & & \sigma_m & \end{array}\right].$$

Si può osservare che il numero di valori singolari di una matrice $A \in \mathbb{R}^{m\times n}$ è pari a $p = \min\{m, n\}$. In ogni caso, per convenzione i valori singolari sono ordinati in modo tale che

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0.$$

Dalla relazione (3.1), moltiplicando i due membri per $V$, segue che

$$AV = U\Sigma, \tag{3.2}$$

mentre moltiplicandola per $U^T$ risulta

$$U^T A = \Sigma V^T \quad \Rightarrow \quad A^T U = V\Sigma^T. \tag{3.3}$$

Indicando con $\mathbf{u}_i$ e $\mathbf{v}_i$ le colonne di $U$ e $V$:

$$U = [\mathbf{u}_1\ \mathbf{u}_2\ \mathbf{u}_3\ \ldots\ \mathbf{u}_{m-1}\ \mathbf{u}_m]$$

$$V = [\mathbf{v}_1\ \mathbf{v}_2\ \mathbf{v}_3\ \ldots\ \mathbf{v}_{n-1}\ \mathbf{v}_n]$$

la relazione (3.2) risulta

$$\begin{array}{l} AV = [A\mathbf{v}_1\ A\mathbf{v}_2\ \ \ldots\ A\mathbf{v}_p\ \mathbf{o}\ \ldots\ \mathbf{o}] \\ U\Sigma = [\sigma_1\mathbf{u}_1\ \sigma_2\mathbf{u}_2\ \ \ldots\ \sigma_p\mathbf{u}_p\ \mathbf{o}\ \ldots\ \mathbf{o}] \end{array}$$

da cui, uguagliando le colonne tra le matrici a primo e secondo membro

$$A\mathbf{v}_i = \sigma_i\mathbf{u}_i, \qquad i = 1, \ldots, p = \min\{m, n\}.$$

Ovviamente se $p = n$ nelle matrici $AV$ e $U\Sigma$ non ci saranno le ultime colonne uguali a zero. Dalla relazione (3.3) risulta invece

$$\begin{array}{l} A^TU = \left[A^T\mathbf{u}_1\ A^T\mathbf{u}_2\ \ \ldots\ A^T\mathbf{u}_p\ \mathbf{o}\ \ldots\ \mathbf{o}\right] \\ V\Sigma^T = [\sigma_1\mathbf{v}_1\ \sigma_2\mathbf{v}_2\ \ \ldots\ \sigma_p\mathbf{v}_p\ \mathbf{o}\ \ldots\ \mathbf{o}] \end{array}$$

da cui, uguagliando le colonne tra le matrici a primo e secondo membro

$$A^T\mathbf{u}_i = \sigma_i\mathbf{v}_i, \qquad i = 1, \ldots, p = \min\{m, n\}.$$

I vettori $\mathbf{u}_i$ sono detti vettori singolari sinistri, mentre i vettori $\mathbf{v}_i$ sono detti vettori singolari destri.
I valori singolari al quadrato sono gli autovalori non nulli della matrice $A^TA$ (e anche di $AA^T$). Infatti supponendo $m > n$

$$A^TA = (U\Sigma V^T)^T(U\Sigma V^T) = V\Sigma^TU^TU\Sigma V^T = V\Sigma^T\Sigma V^T = V\Sigma_1V^T$$

dove

$$\Sigma_1 = \Sigma^T\Sigma = \begin{bmatrix} \sigma_1^2 & & & & \\ & \sigma_2^2 & & & \\ & & \ddots & & \\ & & & \sigma_{n-1}^2 & \\ & & & & \sigma_n^2 \end{bmatrix}.$$

Riscrivendo la relazione

$$A^TA = V\Sigma_1V^T \qquad \Rightarrow \qquad (A^TA)V = V\Sigma_1$$

risulta

$$(A^TA)\mathbf{v}_i = \sigma_i^2\mathbf{v}_i, \qquad i = 1, \ldots, n,$$

quindi $\sigma_i^2$ è l'$i-$esimo autovalore di $A^TA$ e $\mathbf{v}_i$ è il relativo autovettore.
Ricordiamo che si definisce Rango di una matrice (rank) il numero massimo di righe (o colonne) di una matrice linearmente indipendenti. Se $r = \text{rank}(A)$ allora

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_p = 0, \qquad p = \min\{m, n\}.$$

La rappresentazione della SVD può avvenire in forma più compatta eliminando dalle matrici $U, \Sigma$ e $V$ le righe (o le colonne) che sono moltiplicate per elementi uguali a zero. In forma compatta una matrice $A \in \mathbb{R}^{m\times n}$ di rango $r < n < m$ viene rappresentata dalla decomposizione

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

in cui

$$U \in \mathbb{R}^{m\times r}, \qquad \Sigma \in \mathbb{R}^{r\times r}, \qquad V \in \mathbb{R}^{n\times r}.$$

Osserviamo che la matrice $\Sigma$ può essere scritta in modo diverso. Sia $E_i$ la matrice di dimensione $r \times r$ i cui elementi sono tutti uguali a zero tranne quello di posto $(i, i)$ che e uguale a 1. Quindi

$$\Sigma = \sum_{i=1}^{r} \sigma_i E_i.$$

Per esempio se $r = 3$ allora

$$\Sigma = \sigma_1 \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \sigma_2 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \sigma_3 \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Indicato con $\mathbf{e}_i$ l'$i-$esimo vettore della base canonica di $\mathbb{R}^r$ allora la matrice $E_i$ può essere scritta come

$$E_i = \mathbf{e}_i \mathbf{e}_i^T.$$

Per esempio

$$E_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} = \mathbf{e}_2 \mathbf{e}_2^T.$$

Scrivendo $\Sigma$ come

$$\Sigma = \sum_{i=1}^{r} \sigma_i \mathbf{e}_i \mathbf{e}_i^T$$

e sostituendo tale rappresentazione all'interno dell'espressione della SVD risulta:

$$A = U\Sigma V^T = U \sum_{i=1}^{r} \sigma_i \mathbf{e}_i \mathbf{e}_i^T V^T = \sum_{i=1}^{r} \sigma_i U \mathbf{e}_i \mathbf{e}_i^T V^T = \sum_{i=1}^{r} \sigma_i (U\mathbf{e}_i)(V\mathbf{e}_i)^T.$$

Il vettore $U\mathbf{e}_i$ rappresenta la $i-$esima colonna della matrice $U$, cioè l'$i-$esimo vettore singolare sinistro $\mathbf{u}_i$, mentre $V\mathbf{e}_i$ rappresenta la $i-$esima colonna della matrice $V$, cioè l'$i-$esimo vettore singolare destro $\mathbf{v}_i$, quindi

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

in cui ogni matrice $\mathbf{u}_i\mathbf{v}_i^T$ ha rango 1.
Osserviamo che per memorizzare tutti gli elementi della matrice $A$ (indipendentemente dal rango) sono necessarie $m \times n$ locazioni di memoria, mentre per memorizzare la decomposizione ai valori singolari della matrice $A$ di rango $r$ ne servono $m \times r$ locazioni per memorizzare la matrice $U$, $n \times r$ locazioni per memorizzare la matrice $V$ ed $r$ locazioni per memorizzare i valori singolari. In tutto le locazioni necessarie sono:

$$mr + nr + r = (m + n + 1)r,$$

quindi memorizzare la matrice usando la sua SVD conviene se:

$$r(m + n + 1) < mn \qquad \Rightarrow \qquad r < \frac{mn}{m + n + 1}.$$

Per esempio se $m = 640$ e $n = 480$ allora

$$r \leq 274.$$

Sfruttando la positività dei valori singolari si può porre

$$\mathbf{w}_i = \sqrt{\sigma_i}\mathbf{u}_i, \qquad \mathbf{z}_i = \sqrt{\sigma_i}\mathbf{v}_i$$

cosicchè la matrice $A$ viene scritta nel seguente modo

$$A = \sum_{i=1}^{r} \mathbf{w}_i\mathbf{z}_i^T.$$

### 3.1.1 Risoluzione del problema dei minimi quadrati con i valori singolari

Sia $A \in \mathbb{R}^{m\times n}$, di rango $r$, con $m > n \geq r$, allora si vuole risolvere il problema di minimo

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|_2^2$$

essendo nota la SVD della matrice $A$:

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

Poichè la norma 2 è invariante sotto trasformazioni ortogonali, allora

$$\begin{aligned} \|A\mathbf{x} - \mathbf{b}\|_2^2 &= \|U^T(A\mathbf{x} - \mathbf{b})\|_2^2 = \|U^TA\mathbf{x} - U^T\mathbf{b}\|_2^2 \\ &= \|U^TAVV^T\mathbf{x} - U^T\mathbf{b}\|_2^2. \end{aligned}$$

Poniamo $\mathbf{y} = V^T\mathbf{x}$, e inoltre, poichè,

$$\Sigma = U^T AV$$

risulta

$$\|A\mathbf{x} - \mathbf{b}\|_2^2 = \|\Sigma\mathbf{y} - U^T\mathbf{b}\|_2^2$$

e, per trovare un'espressione per il vettore $U^T\mathbf{b}$ possiamo sfruttare la decomposizione di $U$ in colonne:

$$U^T\mathbf{b} = \begin{bmatrix} \mathbf{u}_1^T\mathbf{b} \\ \mathbf{u}_2^T\mathbf{b} \\ \vdots \\ \mathbf{u}_n^T\mathbf{b} \end{bmatrix}$$

ottenendo

$$\|A\mathbf{x} - \mathbf{b}\|_2^2 = \sum_{i=1}^{n}(\sigma_i y_i - \mathbf{u}_i^T\mathbf{b})^2 = \sum_{i=1}^{r}(\sigma_i y_i - \mathbf{u}_i^T\mathbf{b})^2 + \sum_{i=r+1}^{n}(\mathbf{u}_i^T\mathbf{b})^2.$$

Il minimo di tale funzione viene raggiunto quando

$$\sigma_i y_i - \mathbf{u}_i^T\mathbf{b} = 0$$

da cui segue che

$$y_i = \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}, \qquad i = 1, \ldots, r.$$

Il vettore $\mathbf{y}^* \in \mathbb{R}^n$ che risolve il problema di minimo ha come componenti

$$y_i^* = \begin{cases} \dfrac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}, & i = 1, \ldots, r \\ \\ 0 & i = r+1, \ldots, n. \end{cases}$$

Poichè $\mathbf{x} = V\mathbf{y}$ allora il vettore $\mathbf{x}^* \in \mathbb{R}^n$ che risolve il problema di minimo definito inizialmente è

$$\mathbf{x}^* = V\mathbf{y}^* = \sum_{i=1}^{r} y_i^*\mathbf{v}_i = \sum_{i=1}^{r} \frac{\mathbf{u}_i^T\mathbf{b}}{\sigma_i}\mathbf{v}_i$$

mentre il valore minimo cercato è

$$\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|_2^2 = \sum_{i=r+1}^{n} |\mathbf{u}_i^T\mathbf{b}|^2.$$

### 3.1.2 La Pseudoinversa di Moore-Penrose

Se la matrice $A$ fosse quadrata e non singolare allora risolvere il sistema $A\mathbf{x} = \mathbf{b}$ trovando il vettore che minimizza la norma 2 $\|A\mathbf{x} - \mathbf{b}\|_2^2$ significa trovare, il vettore uguale al prodotto tra l'inversa di $A$ (che esiste) ed il vettore dei termini noti $\mathbf{b}$:

$$\mathbf{x} = A^{-1}\mathbf{b}.$$

La matrice inversa non esiste per matrici rettangolari, tuttavia il concetto può essere esteso introducendo la nozione di matrice pseudoinversa, indicata con $A^+$, tale che il vettore $\mathbf{x}^*$ trovato al termine del paragrafo precedente possa essere scritto come

$$\mathbf{x}^* = A^+\mathbf{b}.$$

**Definizione 3.1.2** *Sia $A \in \mathbb{R}^{m\times n}$ una matrice di rango $r$. La matrice $A^+ \in \mathbb{R}^{n\times m}$ tale che*

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

*dove $\Sigma^+ \in \mathbb{R}^{n\times m}$ è la matrice che ha elementi $\sigma_{ij}^+$ nulli per $i \neq j$, mentre se $i = j$:*

$$\sigma_{ii}^+ = \begin{cases} \dfrac{1}{\sigma_i}, & i = 1, \ldots, r \\ \\ 0 & i = r+1, \ldots, p = \min\{m, n\}. \end{cases}$$

*è detta pseudoinversa di Moore-Penrose.*

Valgono la seguenti proprietà:

1. La matrice $X = A^+$ è l'unica matrice che soddisfa le seguenti equazioni, dette di Moore-Penrose:

$$\begin{array}{ll} a) & AXA = A \\ b) & XAX = X \\ c) & (AX)^T = AX \\ d) & (XA)^T = XA; \end{array}$$

2. Se il rango di $A$ è massimo allora:
se $m \geq n$, $A^+ = (A^TA)^{-1}A^T$,
se $m \leq n$, $A^+ = A^T(AA^T)^{-1}$,
se $m = n$, $A^+ = A^{-1}$;

3.
$$A^+ = \sum_{i=1}^{r} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T;$$

4. Considerato il sistema rettangolare
$$A\mathbf{x} = \mathbf{b}$$
allora il vettore soluzione nel senso dei minimi quadrati è
$$\mathbf{x}^* = A^+\mathbf{b}.$$
Infatti
$$A^+\mathbf{b} = \sum_{i=1}^{r} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T \mathbf{b} = \sum_{i=1}^{r} \frac{\mathbf{u}_i^T \mathbf{b}}{\sigma_i} \mathbf{v}_i = \mathbf{x}^*.$$

Ricordiamo che se $A \in \mathbb{R}^{m\times n}$ allora si definisce Range di $A$, e si indica con $R(A)$, l'insieme:
$$R(A) = \{\mathbf{y} \in \mathbb{R}^m \mid \mathbf{y} = A\mathbf{x},\ \mathbf{x} \in \mathbb{R}^n\}.$$
Quindi se $\mathbf{y} \in R(A)$ allora esiste un vettore $\mathbf{x} \in \mathbb{R}^n$ tale che:
$$\mathbf{y} = A\mathbf{x} = \sum_{i=1}^{p} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \mathbf{x} = \sum_{i=1}^{p} (\sigma_i \mathbf{v}_i^T \mathbf{x}) \mathbf{u}_i,$$
quindi le colonne della matrice $U$ rappresentano una base (ovviamente ortonormale) dell'insieme $R(A)$. Procedendo in modo analogo si può definire il Range della matrice $A^T$, $R(A^T)$ e verificare che le colonne della matrice $V$ rappresentano una base per tale spazio.

## 3.2 Un'applicazione della SVD: La Compressione di Immagini Digitali

Il processo di digitalizzazione è un processo di approssimazione (campionamento): fisicamente all'immagine reale viene associata una griglia composta da un elevato numero di piccoli quadrati, detti pixel. A ciascuno di questi pixel viene attribuito un colore (di solito uguale alla media dei colori contenuti nella cella oppure uguale al colore dominante). Nelle foto in bianco

e nero il valore di un pixel è un valore che corrisponde ad una tonalità di grigio. Teoricamente il valore può andare da 0 a $\infty$ perchè infinite sono le tonalità di grigio possibili. Non essendo possibile tuttavia rappresentare infiniti valori allora si pone un limite massimo e si considera un sottoinsieme finito di tali tonalità. Una rappresentazione conveniente è quella di associare ad ognitonalità di grigio un valore reale appartenente all'intervallo $[0, 1]$, considerando che il valore 0 viene associato al bianco e 1 al nero. A questo punto di sceglie un sottoinsieme discreto di tali valori: una quantificazione consueta è quella che prevede 256 livelli di grigio, così ogni livello di grigio può essere rappresentato utilizzando giusto 8 bit ($256 = 2^8$). Le immagini a colori sono rappresentate in modo più complesso perchè l'informazione è multidimensionale. Per rappresentare un colore si usa un determinato spazio definito scegliendo un insieme di colori base. Lo spazio di rappresentazione usato più comunemente è quello RGB (Red-Green-Blue cioè Rosso Verde Blu). In tale spazio ogni pixel è rappresentato da un vettore tridimensionale in cui ogni componente rappresenta l'intensità di uno dei colori di base:

$$Colore = x_R R + x_G G + x_B B.$$

I valori $x_R, x_G, x_B$ variano nell'intervallo di numeri interi $[0, 255]$ oppure assumono valori normalizzati appartenenti all'intervallo reale $[0, 1]$. Simile allo spazio RGB è lo spazio CMYK che si basa sulla scomposizione dei colori nei 4 colori primari Ciano, Magenta, Giallo e Nero. In questo caso il colore viene scomposto in una quadrupla di valori:

$$Colore = x_C C + x_M M + x_Y Y + x_N N.$$

Ci sono altre rappresentazioni molto più complesse come quella $YIQ$, utilizzata dallo standard televisivo stantunitense NTSC (National Television Standard Committee). In questo caso il valore $Y$ rappresenta una tonalità del grigio mentre $I$ e $Q$ sono i cosiddetti segnali di crominanza e rappresentano le coordinate del colore in una particolare rappresentazione. La conversione da RGB a YIQ avviene attraverso la relazione matriciale:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

Una volta stabilito lo spazio dei colori ogni pixel è memorizzato attraverso un vettore definito in tale spazio.
Se l'immagine ha $m \times n$ pixel e lo spazio scelto è quello RGB allora per rappresentarla è necessario memorizzare $3mn$ elementi. Nella seguente tabella sono state riportate le quantità di memoria necessarie per memorizzare immagini digitali di dimensioni variabili nei due spazi usati più comunemente:

| Dimensione | Spazio RGB | Spazio CMYK | Memoria (RGB) in Mb |
|---|---|---|---|
| $640 \times 480$ | 921600 byte | 1228800 byte | 0.9 Mb |
| $800 \times 600$ | 1440000 byte | 1920000 byte | 1.1 Mb |
| $1024 \times 768$ | 2359296 byte | 3145728 byte | 2.25 Mb |

La dimensioni dei file sono piuttosto elevate quindi se un'immagine digitale deve essere trasmessa usando un opportuno canale il costo di trasmissione sia in termini di tempo che di prestazioni richieste è piuttosto elevato. Questa è la motivazione che ha spinto a ricercare meccanismi possibimente affidabili e poco costosi per comprimere l'informazione contenuta in un'immagine digitale. La tecnica di compressione delle immagini si basa sul fatto che queste non sono costituite da un insieme casuale di punti colorati ma esiste una determinata struttura. Questa struttura può essere sfruttata per rappresentare l'immagine utilizzando un numero minore di elementi. Esistono due tipi ci compressione:
1. Compressione lossless (Senza perdita di informazione);
2. Compressione lossy (Con perdita di informazione).
Per misurare la qualità di un algoritmo di compressione si utilizza il cosiddetto indice di compressione

$$C = \frac{n_I}{n_{I_c}}$$

dove $n_I$ è il numero di unità di memoria necessarie a memorizzare l'immagine originale, mentre $n_{I_c}$ è il numero di unità di memoria necessarie a memorizzare l'immagine compressa.
Un valore $C = 10$ significa che per memorizzare l'immagine compressa serve il 10% di quella necessaria a memorizzare l'immagine originale. Nella compressione lossless l'immagine ricostruita è identica, pixel per pixel, all'immagine originale. Le routine di compressione vengono applicate all'insieme dei dati cercando di memorizzare le stesse informazioni ma con un numero inferiore di informazioni (esempio: zone di colore uniforme). Le due tecniche più utilizzate per la compressione di immagini digitali sono:
1. GIF (Graphic Interchange Format): L'immagine viene convertita in una

lunga stringa di dati (concatendo delle righe) a cui è applicato l'algoritmo di compressione LZW;
2. JPEG (Joint Experts Photographic Group): Alla stringa di dati viene applicata prima la Trasformata Discreta Coseno e poi la cosiddetta codifica di Huffman.
Il vantaggio principale di tale tecnica è la ricostruzione fedele dell'immagine di partenza mentre lo svantaggio sta nel fatto che l'indice di compressione potrebbe essere un numero non molto elevato (prossimo a 1) e quindi il processo di compressione risulta poco efficace. Nella compressione lossy l'immagine ricostruita non è perfettamente uguale a quella di partenza. La motivazione di questa scelta dipende dal fatto che il processo attraverso il quale l'occhio umano percepisce un'immagine reale è esattamente lo stesso su cui si basa la digitalizzazione di un'immagine e pertanto l'occhio umano non è in grado di percepire perfettamente la variazione del singolo pixel. Questo tipo di compressione funziona bene solitamente con immagini prese dalla vita reale (foto non artistiche). L'immagine ricostruita è simile a quella originale nel senso che l'occhio umano percepisce l'immagine nel suo insieme e non nei dettagli.

### 3.2.1 La Decomposizione ai Valori Singolari Troncata

Ricordiamo che se $A \in \mathbb{R}^{m\times n}$, $m \geq n$, $r = \text{rank}(A)$, allora

$$A = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T.$$

La matrice $A$ (di rango $r$) viene scritta come una somma di $r$ matrici di rango 1. Ricordiamo che se $A \in \mathbb{R}^{m\times n}$ si definisce la norma 2 della matrice come il numero

$$\|A\|_2 = \sqrt{\rho(A^T A)}$$

dove $\rho(\cdot)$ indica il cosiddetto raggio spettrale della matrice, cioè il massimo modulo di un autovalore. Poichè gli autovalori della matrice $A^T A$ coincidono con il quadrato dei valori singolari di $A$ segue che

$$\|A\|_2 = \sqrt{\rho(A^T A)} = \sqrt{\sigma_1^2} = \sigma_1.$$

Fissato un valore $k$, intero compreso tra 1 ed $r$, $1 \leq k < r$, vogliamo trovare la matrice $A_k$, di rango $k$, più vicina ad $A$. Si vuole risolvere cioè il problema

di minimo:

$$\|A - A_k\|_2 = \min_{\text{rank}(X)=k} \|A - X\|_2.$$

La risoluzione a questo problema è legata ancora una volta alla SVD della matrice $A$ ed è la cosiddetta SVD Troncata:

$$A_k = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_k \mathbf{u}_k \mathbf{v}_k^T.$$

Infatti poichè la matrice $A_k$ ha $k$ valori singolari diversi da zero, ha ovviamente rango $k$. Inoltre, calcolando la differenza tra le matrici $A$ e $A_k$, risulta

$$A - A_k = \sum_{i=k+1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

e quindi i valori singolari della matrice $A - A_k$ sono

$$\sigma_{k+1} \geq \sigma_{k+2} \geq \cdots \geq \sigma_r.$$

La norma 2 di tale matrice è

$$\|A - A_k\|_2 = \sigma_{k+1}.$$

Se il più grande dei valori singolari esclusi dalla sommatoria ha un valore molto piccolo allora l'approssimazione di rango $k$ della matrice $A$ è una buona approssimazione di $A$. L'uso della SVD nella compressione di immagini digitali avviene nel seguente modo: se $A$ è la matrice $m \times n$ dei pixel dell'immagine digitale si può ragionevolmente pensare che ci siano diverse righe (o colonne) che variano poco (cioè tali da poter essere considerate come linearmente dipendenti), quindi il rango di $A$ può essere un valore molto più piccolo del $\min\{m, n\}$. All'immagine definita dalla $A$ viene sostituita l'immagine definita da $A_k$.
Se indichiamo con $\widehat{A}$ la matrice dell'immagine approssimata allora la qualità dell'immagine percepita dall'occhio umano è ovviamente una misura molto soggettiva. Definiamo la seguente quantità:

$$RSME = \sqrt{\frac{\sum_{i=1}^{m}\sum_{j=1}^{n}(a_{ij} - \widehat{a}_{ij})^2}{mn}}$$

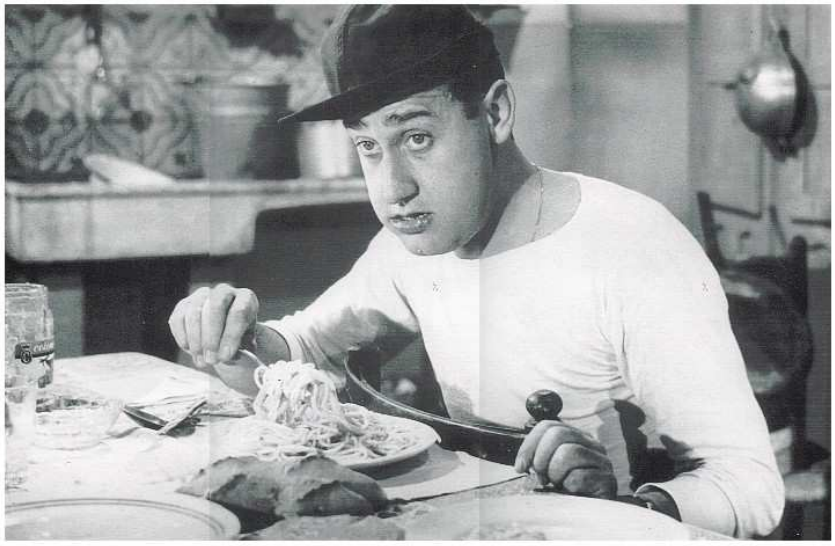


Figura 3.1: Immagine originale

Il valore RSME (Root Square Mean Error-ma spesso riportato anche come RMSE) è una misura (media) oggettiva di quanto l'immagine approssimata differisce da quella originale. Il seguente valore

$$PSNR = 10 \log_{10} \left( \frac{1}{RSME} \right)$$

misura invece l'errore percepito dall'occhio. Poichè il calcolo della SVD di una matrice di grandi dimensioni ha un alto costo computazionale (pari a circa $m^2 n$) allora può essere utile suddividere l'immagine in un certo numero di parti (anche di dimensioni diseguali) e manipolarle in modo indipendente. Questa tecnica consente di utilizzare un valore del rango non costante per le parti dell'immagine. Per esempio una parte di immagine che rappresenta il cielo (o il mare) può essere rappresentata con una matrice di rango 1 mentre per la parte in cui sono presenti elementi dai colori e forme diversi si può utilizzare un rango superiore.
Nelle pagine seguenti sono presentati alcuni esempi di applicazione della SVD alla compressione di un'immagine in bianco e nero $774 \times 523$ di rango 475, utilizzando un'approssimazione troncata avente rango crescente.

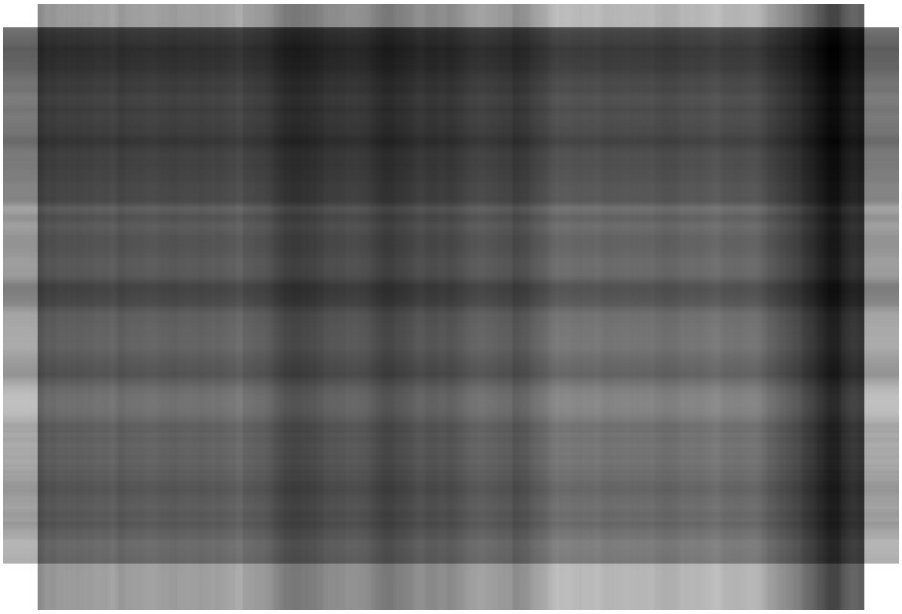


Figura 3.2: Approssimazione di rango 1

Figura 3.3: Approssimazione di rango 10

Figura 3.4: Approssimazione di rango 50

Figura 3.5: Approssimazione di rango 100

# Capitolo 4

# Interpolazione con funzioni spline

## 4.1 Funzioni Spline

L'interpolazione polinomiale con un numero di nodi piuttosto alto può dar luogo a polinomi interpolanti che mostrano un comportamento fortemente oscillatorio che può essere inaccettabile. In questo caso si preferisce usare una diversa strategia consistente nell'approssimare la funzione con polinomi di basso grado su sottointervalli dell'intervallo di definizione. Per esempio si potrebbe utilizzare una serie di polinomi di basso grado che interpolano solo alcuni punti consecutivi. Supponendo che $n$ sia un multiplo di 3 e denotando con $P_{3,j}(x)$ il polinomio di interpolazione di terzo grado associato ai nodi $x_{3j-3}$, $x_{3j-2}$, $x_{3j-1}$, $x_{3j}$, $j = 1, 2, \ldots, n/3$, si potrebbe costruire la funzione interpolante

$$I_n(x) = P_{3,j}(x) \qquad \text{in } [x_{3j-3}, x_{3j}]$$

che prende il nome di Funzione di tipo polinomiale a tratti. La tecnica esposta non è l'unica, anzi la più popolare è forse quella basata sull'uso delle cosiddette Funzioni Spline.
Con il termine *spline* si indica in lingua inglese un sottile righello usato nella progettazione degli scafi dagli ingegneri navali, per raccordare su un piano un insieme di punti $(x_i, y_i)$, $i = 0, \ldots, n + 1$.
Imponendo mediante opportune guide che il righello passi per i punti assegnati, si ottiene una curva che li interpola. Detta $y = f(x)$ l'equazione della curva definita dalla spline, sotto opportune condizioni $f(x)$ può essere

approssimativamente descritta da pezzi di polinomi di terzo grado in modo che $f(x)$ e le sue prime due derivate risultino ovunque continue. La derivata terza può presentare discontinuità nei punti $x_i$. La spline può essere concettualmente rappresentata e generalizzata nel seguente modo.
Sia

$$\Delta =: a \equiv x_0 < x_1 < x_2 < \cdots < x_n < x_{n+1} \equiv b$$

una decomposizione dell'intervallo $[a, b]$.

**Definizione 4.1.1** *Si dice funzione Spline di grado $m \geq 1$ relativa alla decomposizione $\Delta$ una funzione $s(x)$ soddisfacente le seguenti proprietà:*

1. *la funzione $s(x)$ in ogni intervallo $[x_i, x_{i+1}]$, $i = 0, \ldots, n$, è un polinomio di grado al più $m$;*

2. *le derivate di $s(x)$ di ordine $1, 2, \ldots, m-1$ sono funzioni continue nell'intervallo $[a, b]$.*

In generale le spline vengono utilizzate in tutte quelle situazioni dove l'approssimazione polinomiale sull'intero intervallo non è soddisfacente. Per $m = 1$ si hanno le cosiddette spline lineari, mentre per $m = 3$ si hanno le spline cubiche.

### 4.1.1 Costruzione della Spline Cubica Interpolante

Assegnata la decomposizione:

$$\Delta =: a \equiv x_0 < x_1 < x_2 < \cdots < x_n < x_{n+1} \equiv b$$

si vuole determinare una spline cubica $s(x)$ tale che

$$s(x_i) = y_i, \qquad i = 0, 1, \ldots, n+1. \tag{4.1}$$

dove $y_i$, $i = 0, \ldots, n+1$, sono $n+2$ assegnati valori.
Indichiamo con $s_i(x)$ la restrizione della spline nell'intervallo $[x_i, x_{i+1}]$, in cui $s_i''(x)$ è una funzione lineare mentre $s_i^{(3)}(x)$ è una costante, quindi

$$s_i''(x) = s_i''(x_i) + s_i^{(3)}(x_i)(x - x_i) \tag{4.2}$$

ovvero, posto

$$M_i = s''_{[x_i, x_{i+1}]}(x_i) \qquad\qquad c_i = s^{(3)}_{[x_i, x_{i+1}]}(x_i)$$

abbiamo

$$s_i''(x) = M_i + c_i(x - x_i). \tag{4.3}$$

Valutando (4.3) in $x_{i+1}$ si ottiene

$$c_i = \frac{M_{i+1} - M_i}{h_i}, \qquad h_i = x_{i+1} - x_i. \tag{4.4}$$

Scriviamo lo sviluppo in serie di Taylor di $s_i(x)$ prendendo come punto iniziale $x_i$ :

$$s_i(x) = s_i(x_i) + s_i'(x_i)(x - x_i) + M_i\frac{(x - x_i)^2}{2} + c_i\frac{(x - x_i)^3}{6} \tag{4.5}$$

e calcoliamola in $x_{i+1}$ sostituendo le condizioni di interpolazione

$$y_{i+1} = y_i + s_i'(x_i)h_i + M_i\frac{h_i^2}{2} + c_i\frac{h_i^3}{6}$$

$$\frac{y_{i+1} - y_i}{h_i} = s_i'(x_i) + M_i\frac{h_i}{2} + c_i\frac{h_i^2}{6}. \tag{4.6}$$

Scriviamo ora lo sviluppo in serie di Taylor di $s_{i-1}(x)$ prendendo come punto iniziale $x_i$ :

$$s_{i-1}(x) = s_{i-1}(x_i) + s_{i-1}'(x_i)(x - x_i) + M_i\frac{(x - x_i)^2}{2} + c_{i-1}\frac{(x - x_i)^3}{6}. \tag{4.7}$$

e calcoliamola in $x_{i-1}$, sostituendo le condizioni di interpolazione sul nodo anche sul nodo $x_i$ in modo tale da assicurare la continuità della spline ed imponendo anche che la derivata seconda, in $x_i$ assuma, nei due intervalli adiacenti lo stesso valore $M_i$, garantendo la continuità anche della derivata seconda

$$y_{i-1} = y_i - s_{i-1}'(x_i)h_{i-1} + M_i\frac{h_{i-1}^2}{2} - c_{i-1}\frac{h_{i-1}^3}{6}$$

$$\frac{y_{i-1} - y_i}{h_{i-1}} = -s_{i-1}'(x_i) + M_i\frac{h_{i-1}}{2} - c_{i-1}\frac{h_{i-1}^2}{6}. \tag{4.8}$$

Osserviamo dalle relazioni (4.7) e (4.6) che $s_i'(x_i)$ può essere calcolata se sono noti i valori $M_i$. Di conseguenza la spline è completamente determinata se si conoscono i valori $M_0$, $M_1, \ldots, M_{n+1}$ (che sono detti momenti). A questo punto imponendo le condizioni di continuità della derivata prima, ovvero

$$s_{i-1}'(x_i) = s_i'(x_i)$$

sommando le equazioni (4.7) e (4.8) le derivate prime si semplificano ricavando l'equazione

$$M_i\frac{h_i+h_{i-1}}{2}-c_{i-1}\frac{h_{i-1}^2}{6}+c_i\frac{h_i^2}{6}=\frac{y_{i+1}-y_i}{h_i}+\frac{y_{i-1}-y_i}{h_{i-1}},$$

sostituendo l'espressione delle derivate terze nei due intervalli

$$c_{i-1}=\frac{M_i-M_{i-1}}{h_{i-1}},\qquad c_i=\frac{M_{i+1}-M_i}{h_i}$$

$$3M_i(h_i+h_{i-1})+(M_{i+1}-M_i)h_i-(M_i-M_{i-1})h_{i-1}=6\left(\frac{y_{i+1}-y_i}{h_i}+\frac{y_{i-1}-y_i}{h_{i-1}}\right)$$

ottenendo, per $i=1,\dots,n$, le equazioni:

$$h_{i-1}M_{i-1}+2(h_i+h_{i-1})M_i+h_iM_{i+1}=6\left(\frac{y_{i+1}-y_i}{h_i}+\frac{y_{i-1}-y_i}{h_{i-1}}\right). \quad (4.9)$$

Le (4.9) rappresentano un sistema di $n$ equazioni nelle $n+2$ incognite $M_0$, $M_1$, $\dots$, $M_{n+1}$. Se per esempio fissiamo i valori $M_0=M_{n+1}=0$ otteniamo la cosiddetta spline cubica naturale e il sistema (4.9) si scrive:

$$A\mathbf{m}=\mathbf{b}$$

dove

$$A=\begin{bmatrix} 2(h_0+h_1) & h_1 & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} & \\ & & & h_{n-1} & 2(h_{n-1}+h_n) \end{bmatrix}$$

e

$$\mathbf{m}=\begin{bmatrix} M_1\\ M_2\\ M_3\\ \vdots\\ M_{n-1}\\ M_n \end{bmatrix}\qquad \mathbf{b}=\begin{bmatrix} b_1\\ b_2\\ b_3\\ \vdots\\ b_{n-1}\\ b_n \end{bmatrix}$$

con

$$b_i = 6\left(\frac{y_{i+1}-y_i}{h_i} + \frac{y_{i-1}-y_i}{h_{i-1}}\right) \qquad i = 1, 2, \ldots, n.$$

Se anzichè fissare i valori delle derivate seconde nulli negli estremi, si fissano i valori della derivata prima negli estremi, si impone cioè che sia

$$\begin{aligned} s'(x_0) &= k_0 \\ s'(x_{n+1}) &= k_{n+1} \end{aligned}$$

con $k_0$ e $k_{n+1}$ valori assegnati, si ottiene un sistema analogo al precedente, di dimensione $n+2$, e la spline interpolante prende il nome di spline cubica completa.

*Osservazione.* La matrice dei coefficienti del sistema ottenuto è simmetrica e definita positiva.

### 4.1.2 Proprietà di Regolarità delle Spline Cubiche

Per ogni $f \in \mathcal{C}^2([a,b])$ definiamo

$$\sigma(f) = \int_{x_0}^{x_{n+1}} [f''(x)]^2 dx$$

che è, in prima approssimazione, una misura del grado di oscillazione di $f$. Infatti ricordando che:

$$f''(x)(1 + (f'(x))^2)^{-3/2}$$

definisce la curvatura della funzione $f$ nel punto $x$, se $|f'(x)|$ è una quantità sufficientemente piccola rispetto a 1 allora la curvatura è definita approssimativamente da $f''(x)$. Conseguentemente

$$\int_a^b [f''(x)]^2 dx.$$

è una misura approssimata della curvatura totale di $f$ su $[a,b]$.

Sia ora $s(x)$ la spline cubica naturale soddisfacente il problema di interpolazione (4.1) ed $f(x)$ una qualunque funzione con derivata seconda continua su $[a,b]$ soddisfacente anch'essa lo stesso problema di interpolazione. Assegnata cioè la decomposizione:

$$\Delta =: a \equiv x_0 < x_1 < x_2 < \cdots < x_n < x_{n+1} \equiv b$$

ed i valori $y_0, y_1, y_2, \ldots, y_n, y_{n+1}$ abbiamo

$$\begin{array}{ll} s(x_i) & = y_i \qquad i = 0, 1, \ldots, n+1 \\ f(x_i) & = y_i \qquad i = 0, 1, \ldots, n+1. \end{array}$$

Sia inoltre $e(x) = f(x) - s(x)$. Vale il seguente risultato.

**Lemma 4.1.1**

$$\int_{x_0}^{x_{n+1}} e''(x)\Phi(x)dx = e'(x_{n+1})\Phi(x_{n+1}) - e'(x_0)\Phi(x_0),$$

*per ogni* $\Phi \in M_0^1([a,b], \Delta)$.

*Dimostrazione.* Osservato che:

$$e''(x)\Phi(x) = \frac{d}{dx}(e'(x)\Phi(x)) - e'(x)\Phi'(x),$$

si ha

$$\begin{aligned} \int_{x_0}^{x_{n+1}} e''(x)\Phi(x)dx &= \int_{x_0}^{x_{n+1}} \left[\frac{d}{dx}(e'(x)\Phi(x)) - e'(x)\Phi'(x)\right] = \\ &= e'(x_{n+1})\Phi(x_{n+1}) - e'(x_0)\Phi(x_0) + \\ &\quad -\sum_{i=0}^{n} \int_{x_i}^{x_{i+1}} e'(x)\Phi'(x)dx. \end{aligned}$$

Poichè $\Phi'$ è costante su ogni intervallino, detta $c_i$ tale costante, possiamo anche scrivere:

$$\int_{x_0}^{x_{n+1}} e''(x)\Phi(x)dx = e'(x_{n+1})\Phi(x_{n+1}) - e'(x_0)\Phi(x_0) - \sum_{i=0}^{n} c_i \int_{x_i}^{x_{i+1}} e'(x)dx.$$

La tesi segue poichè $e(x_i) = f(x_i) - s(x_i) = 0$. □

**Teorema 4.1.1** *Se $s(x)$ è la spline naturale soddisfacente il problema di interpolazione (4.1) allora:*

$$\sigma(s) \leq \sigma(f)$$

*qualunque sia $f$ di classe $\mathcal{C}^2([a,b])$ soddisfacente il problema di interpolazione e tale che $f''(a) = f''(b) = 0$.*

*Dimostrazione.*

$$\begin{aligned}\sigma(f) &= \int_{x_0}^{x_{n+1}} [f''(x)]^2 dx = \int_{x_0}^{x_{n+1}} [f''(x) - s''(x) + s''(x)]^2 dx = \\ &= \int_{x_0}^{x_{n+1}} [e''(x) + s''(x)]^2 dx \\ &= \int_{x_0}^{x_{n+1}} [e''(x)]^2 dx + \int_{x_0}^{x_{n+1}} [s''(x)]^2 dx + 2\int_{x_0}^{x_{n+1}} e''(x)s''(x)dx.\end{aligned}$$

Poichè $s''(x)$ è una spline lineare possiamo applicare il lemma (4.1.1) al terzo integrale a secondo membro, ottenendo

$$\begin{aligned}\sigma(f) &= \int_{x_0}^{x_{n+1}} [e''(x)]^2 dx + \int_{x_0}^{x_{n+1}} [s''(x)]^2 dx + \\ &\quad +2[e'(x_{n+1})s''(x_{n+1}) - e'(x_0)s''(x_0)].\end{aligned}$$

Poichè la spline in oggetto è naturale, segue:

$$\sigma(f) = \int_{x_0}^{x_{n+1}} [e''(x)]^2 dx + \sigma(s)$$

e dunque la tesi. □

*Osservazione 1.* Se $\sigma(s) = \sigma(f)$ allora $e''(x)$ è identicamente nulla. Pertanto $e(x)$ è un polinomio di primo grado e poichè esso si annulla in almeno due nodi è identicamente nullo. Di conseguenza $s \equiv f$.

*Osservazione 2.* La tesi del teorema (4.1.1) è verificata anche dalla spline cubica completa. Infatti in questo caso il termine

$$e'(x_{n+1})s''(x_{n+1}) - e'(x_0)s''(x_0) = 0$$

in virtù del fatto che $e'(x_{n+1}) = e'(x_0) = 0$. In definitiva abbiamo provato che la spline cubica naturale è l'unica funzione che risolve il problema di minimo:

$$\min_{f \in \mathcal{C}^2([a,b])} \int_a^b [f''(x)]^2 dx$$

$$f(x_i) = y_i \qquad i = 0, 1, \ldots, n+1$$

$$f''(a) = f''(b) = 0.$$

Analogamente la spline cubica completa risolve il problema di minimo:

$$\min_{f\in\mathcal{C}^2([a,b])} \int_a^b [f''(x)]^2 dx$$

$$f(x_i) = y_i \qquad i = 0, 1, \ldots, n+1$$

$$f'(a) = k_0, \qquad f'(b) = k_{n+1}.$$

### 4.1.3 Errore nell'Interpolazione con Spline Cubiche

Sia $f \in \mathcal{C}^2([a,b])$ e

$$\Delta =: a \equiv x_0 < x_1 < x_2 < \cdots < x_n < x_{n+1} \equiv b$$

ed $s(x)$ la spline cubica naturale interpolante i nodi $x_i$, $i = 0, \ldots, n+1$, cioè tale che

$$s(x_i) = f(x_i) \qquad i = 0, 1, \ldots, n+1.$$

Ci poniamo il problema di valutare

$$e(x) = f(x) - s(x) \qquad \forall x \in [a,b]$$

e

$$e'(x) = f'(x) - s'(x) \qquad \forall x \in [a,b].$$

A tal fine ricordiamo che la funzione $(\cdot,\cdot)$ che ad ogni coppia di funzioni $g$ ed $h$ continue sull'intervallo $[a,b]$ associa

$$\int_a^b g(x)h(x)dx$$

definisce un prodotto scalare e

$$\|g\| = \left(\int_a^b g^2(x)dx\right)^{1/2}$$

è la corrispondente norma indotta. Tale prodotto scalare verifica la disuguaglianza di Schwartz

$$|(g,h)| \leq \|g\|\|h\|.$$

Osserviamo che

$$e(x_i) = f(x_i) - s(x_i) = 0 \qquad i = 1, 2, \ldots, n+1$$

possiamo perciò applicare il teorema di Rolle in ogni intervallo di tipo $[x_i, x_{i+1}]$, $i = 0, \ldots, n$, e concludere che

$$\exists \xi_i \in ]x_i, x_{i+1}[ \quad \ni' \quad e'(\xi_i) = 0 \qquad i = 0, \ldots, n.$$

Sia ora $x \in ]x_i, x_{i+1}[$ e per fissare le idee supponiamo $x > \xi_i$ (quanto segue può essere ricavato anche nel caso opposto). Allora

$$e'(x) = \int_{\xi_i}^{x} e''(t)dt$$

perciò

$$\begin{aligned} |e'(x)|^2 &= \left| \int_{\xi_i}^{x} e''(t)dt \right|^2 \leq \left( \textstyle\int_{\xi_i}^{x} |e''(t)| \cdot 1 dt \right)^2 \leq \\ &\leq \int_{\xi_i}^{x} |e''(t)|^2 dt \int_{\xi_i}^{x} dt = (x - \xi_i) \int_{\xi_i}^{x} |e''(t)|^2 dt. \end{aligned}$$

e di conseguenza

$$|e'(x)|^2 \leq h \int_a^b |e''(t)|^2 dt \tag{4.10}$$

dove $h = \max_{i=0,\ldots,n} (x_{i+1} - x_i)$.
Abbiamo già stabilito in precedenza (teorema 4.1.1) che

$$\int_a^b |e''(t)|^2 dt = \sigma(f) - \sigma(s) \leq \sigma(f)$$

e pertanto la (4.10) può ulteriormente essere maggiorata da

$$|e'(x)| \leq h^{\frac{1}{2}} \sigma^{\frac{1}{2}}(f). \tag{4.11}$$

Analogamente possiamo scrivere

$$e(x) = \int_{x_i}^{x} e'(t)dt$$

e conseguentemente

$$\begin{aligned} |e(x)|^2 &= \left| \int_{x_i}^{x} e'(t)dt \right|^2 \leq \left( \int_{x_i}^{x} |e'(t)| \cdot 1 dt \right)^2 \\ &= (|e'|, 1)^2 \leq \| \ |e'| \ \|^2 \|1\|^2 = \int_{x_i}^{x} |e'(t)|^2 dt \int_{x_i}^{x} dt \\ &\leq \int_{x_i}^{x} h\sigma(f)dt \ h \leq h^3 \sigma(f). \end{aligned}$$

In definitiva abbiamo

$$|e(x)| \leq h^{\frac{3}{2}} \sigma^{\frac{1}{2}}(f)$$

e quindi in generale vale il seguente

**Teorema 4.1.2** *Sia $f \in \mathcal{C}^2([a,b])$ ed $s(x)$ la spline cubica interpolante i nodi*

$$a \equiv x_0 < x_1 < \cdots < x_n < x_{n+1} \equiv b.$$

*Allora*

$$\forall x \in [a,b] : |f^{(j)}(x) - s^{(j)}(x)| \leq h^{\frac{3}{2}-j} \sigma(f) \qquad j = 0, 1. \ \square$$

# Capitolo 5

# Metodi numerici per equazioni differenziali ordinarie

## 5.1 Introduzione

Supponiamo che sia assegnato il seguente problema differenziale:

$$\begin{cases} y'(t) = f(t, y(t)) \\ y(t_0) = y_0 \end{cases} \tag{5.1}$$

dove $f : [t_0, T] \times \mathbb{R} \to \mathbb{R}$ è una funzione continua rispetto a $t$ e Lipschitziana rispetto a $y$, cioè esiste una costante positiva $L$ tale che, per ogni $x, y \in \mathbb{R}$, risulta

$$|f(t, x) - f(t, y)| \leq L|x - y|, \qquad \forall t \in [t_0, T].$$

Il problema (5.1) prende il nome di problema di Cauchy del primo ordine ai valori iniziali. Risolvere (5.1) significa determinare una funzione $y(t)$ di classe $\mathcal{C}^1([t_0, T])$ la cui derivata prima soddisfi l'equazione assegnata e che passi per il punto $(t_0, y_0)$. In base alle ipotesi fatte sulla funzione $f(t, y(t))$ il teorema di Cauchy assicura l'esistenza e l'unicità di tale funzione.

**Teorema 5.1.1 (di Cauchy)** *Sia $f(t, y) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, una funzione definita e continua per ogni $(t, y)$ appartenente alla regione $[t_0, T] \times \mathbb{R}$, e sia inoltre Lipschitziana rispetto a $y$ allora per ogni condizione iniziale esiste un'unica soluzione continua e differenziabile $y(t)$ del problema (5.1).*

Descriveremo nel seguito alcune classi di metodi per equazioni differenziali del primo ordine, considerando sempre che tali metodi possono essere applicati

anche a sistemi. Tali metodi ovviamente non forniscono in forma chiusa l'espressione della soluzione $y(t)$ ma solo una sua approssimazione in un insieme discreto di punti. Se siamo interessati alla funzione $y(t)$ nell'intervallo $[t_0, T]$ lo suddividiamo in $N$ sottointervalli ciascuno di ampiezza $h = (T - t_0)/N$ e definiamo i punti

$$t_n = t_{n-1} + h = t_0 + nh, \qquad n = 0, \ldots, N$$

dove la soluzione verrà approssimata.
Prima di studiare le principali classi di metodi numerici per equazioni differenziali vediamo prima di descriverne alcuni piuttosto semplici. Partendo dal problema di Cauchy

$$y'(t) = f(t, y(t)) \tag{5.2}$$

e supponendo di voler calcolare la funzione in $t_{n+1}$ noto il suo valore in $t_n$, andiamo ad integrare membro a membro (5.2):

$$\int_{t_n}^{t_{n+1}} y'(t)dt = \int_{t_n}^{t_{n+1}} f(t, y(t))dt \tag{5.3}$$

cioè

$$y(t_{n+1}) - y(t_n) = \int_{t_n}^{t_{n+1}} f(t, y(t))dt \tag{5.4}$$

quindi il problema equivale ad approssimare l'integrale a secondo membro. Un

## 5.2 I metodi multistep lineari

L'espressione generale di un metodo multistep lineare è la seguente

$$\sum_{j=0}^{k} \alpha_j y_{n+j} - h \sum_{j=0}^{k} \beta_j f_{n+j} = 0, \tag{5.5}$$

dove $f_{n+j} = f(t_{n+j}, y_{n+j})$. Il valore intero $k$ indica il numero di passi del metodo ed indica il numero di approssimazioni necessarie per poter calcolare un valore $y_n$. Infatti un metodo multistep lineare funziona in questo modo. Inizialmente si suppone di conoscere le quantità $y_0, y_1, \ldots, y_{k-1}$ e, scrivendo l'espressione (5.5), per $n = 0$ si ricava:

$$\sum_{j=0}^{k} \alpha_j y_j - h \sum_{j=0}^{k} \beta_j f_j = 0,$$

da cui è possibile ricavare (o esplicitamente o attraverso un metodo numerico, come vedremo in seguito) l'unica incognita $y_k$. Una volta calcolata tale approssimazione si scrive l'equazione (5.5), per $n = 1$:

$$\sum_{j=0}^{k} \alpha_j y_{j+1} - h \sum_{j=0}^{k} \beta_j f_{j+1} = 0,$$

e si ricava l'unica incognita $y_{k+1}$ e, in questo modo, applicando ripetutamente il metodo si calcolano tutte le approssimazioni fino all'ultima $y_N$. Da quanto detto tuttavia emergono due questioni aperte:

1. Inizialmente si conosce solo il valore $y_0$ mentre sono necessari anche i valori $y_1, \ldots, y_{k-1}$;

2. Se $\beta_k \neq 0$ e la funzione $f(t, y)$ è non lineare rispetto a $y$ allora ad ogni passo non è possibile calcolare esplicitamente il valore $y_{n+k}$.

Vedremo in seguito come affrontare queste due problematiche. Adesso facciamo alcune considerazioni preliminari sulle condizioni da imporre ai coefficienti del metodo $\alpha_j$ e $\beta_j$.
Osservando che l'espressione di un metodo multistep lineari rimane inalterata se viene moltiplicata per una costante non nulla allora per normalizzare i coefficienti si impone che sia $\alpha_k = 1$. Un altro vincolo sui coefficienti è che sia

$$|\alpha_0| + |\beta_0| \neq 0$$

in modo tale che siano certamente coinvolti nella formula $y_n$ e $y_{n+k}$. Il motivo di tale scelta è quello di non considerare metodi del tipo

$$y_{n+2} - y_{n+1} - h f_{n+1} = 0$$

che è essenzialmente un metodo ad un passo e non a 2 passi ed è sostanzialmente indistingubile dal metodo ad un passo:

$$y_{n+1} - y_n - h f_n = 0.$$

Prima di vedere quali sono le altre condizioni che i coefficienti del metodo devono soddisfare definiamo i due seguenti polinomi:

$$\rho(z) = \sum_{j=0}^{k} \alpha_j z^j$$

detto primo polinomio caratteristico del metodo, e

$$\sigma(z) = \sum_{j=0}^{k} \beta_j z^j$$

detto secondo polinomio caratteristico del metodo.
Una prima classificazione tra i metodi multistep lineari è quella in base al valore assunto dal coefficiente $\beta_k$. Infatti se in (5.5) $\beta_k = 0$ il metodo si dice esplicito, altrimenti si dice implicito. Nel primo caso il metodo si può riscrivere come:

$$y_{n+k} = h \sum_{j=0}^{k-1} \beta_j f_{n+j} - \sum_{j=0}^{k-1} \alpha_j y_{n+j},$$

mettendo in evidenza che il valore $y_{n+k}$ può essere calcolato esplicitamente utilizzando solo valori già noti. Invece se $\beta_k \neq 0$ e la funzione $f(t, y)$ è non lineare rispetto alla variabile $y$ allora ad ogni passo il metodo può essere scritto come:

$$y_{n+k} = h\beta_k f_{n+k} + h \sum_{j=0}^{k-1} \beta_j f_{n+j} - \sum_{j=0}^{k-1} \alpha_j y_{n+j},$$

cosicchè per poter calcolare $y_{n+k}$ è necessario risolvere l'equazione non lineare

$$y_{n+k} = h\beta_k f(t_{n+k}, y_{n+k}) + G_{n,k} \tag{5.6}$$

avendo posto

$$G_{n,k} = h \sum_{j=0}^{k-1} \beta_j f_{n+j} - \sum_{j=0}^{k-1} \alpha_j y_{n+j},$$

che è una quantità nota. L'equazione (5.6) è anche un'equazione di punto fisso

$$x = \Phi(x)$$

dove

$$\Phi(x) = h\beta_k f(t_{n+k}, x) + G_{n,k}.$$

In questo caso è naturale definire il metodo di iterazione funzionale:

$$y_{n+k}^{[m+1]} = \Phi\left(y_{n+k}^{[m]}\right)$$

prendendo come approssimazione iniziale il valore dell'approssimazione più vicina a quella incognita, cioè:

$$y_{n+k}^{[0]} = y_{n+k-1}.$$

Il metodo iterativo è quindi

$$\begin{aligned} y_{n+k}^{[0]} &= y_{n+k-1} \\ y_{n+k}^{[m+1]} &= h\beta_k f(t_{n+k}, y_{n+k}^{[m]}) + G_{n,k}. \end{aligned} \tag{5.7}$$

Affinchè tale metodo converga deve essere

$$|\Phi'(x)| < 1$$

in un opportuno intorno della soluzione. Calcolando la derivata prima della funzione $\Phi(x)$ si ha:

$$\Phi'(x) = h\beta_k \frac{\partial f}{\partial y}(t_{n+k}, x)$$

quindi la condizione per la convergenza diviene:

$$h|\beta_k| \left|\frac{\partial f}{\partial y}\right| < 1 \quad \Rightarrow \quad \left|\frac{\partial f}{\partial y}\right| < \frac{1}{h|\beta_k|}.$$

Tale condizione è sicuramente verificata se risulta $h|\beta_k|L < 1$, con $L$ uguale alla costante di Lipschitz della funzione $f(t, y)$, infatti per definizione

$$\left|\frac{\partial f}{\partial y}\right| \leq L$$

quindi

$$h|\beta_k| \left|\frac{\partial f}{\partial y}\right| \leq h|\beta_k|L$$

ed il metodo è convergente se

$$h|\beta_k|L < 1. \tag{5.8}$$

Tale condizione non presenta eccessivi problemi per la gran parte delle equazioni differenziali, se la (5.8) non è rispettata è sufficiente scegliere un valore $h$ più piccolo, in modo tale che essa sia soddisfatta, e procedere. Un caso

particolare sono i cosiddetti problemi stiff in cui la costante $L \gg 1$ ed in questo caso la (5.8) rappresenta una restrizione molto forte sul passo $h$.
Un ultimo aspetto da analizzare è il fatto che, per applicare il metodo al primo passo (cioè quando $n = 0$) si devono conoscere le approssimazioni $y_1, y_2, \ldots, y_{k-1}$ per poter calcolare $y_k$. Un modo per ottenerle è quello di applicare un metodo che richieda un numero di passi inferiore (per esempio metodi ad un passo come Eulero esplicito o implicito) e di applicare il metodo multistep solo quando si abbiano tutte le approssimazioni necessarie.
Le proprietà che devono soddisfare i coefficienti del metodo sono principalmente tre:
1. Consistenza
2. Zero Stabilità
3. Convergenza.

Vediamo ora di descrivere in maggior dettaglio tali proprietà.

### Convergenza

Quando il passo di integrazione $h$ tende a zero l'insieme di punti discreti $\{t_n\}$ diventa l'intero intervallo $[t_0, T]$. Una proprietà ovvia da richiedere ad un qualsiasi metodo numerico è che, quando $h \to 0$ la soluzione numerica $y_n$ diventa la soluzione teorica $y(t)$, $t \in [t_0, T]$. Questa proprietà è detta Convergenza.

**Definizione 5.2.1** *Un metodo numerico si dice convergente se, per ogni problemi ai valori iniziali soddisfacente le ipotesi si ha:*

$$\lim_{\substack{h \to 0 \\ t = t_0 + nh}} y_n = y(t)$$

*per ogni $t \in [t_0, T]$. Un metodo che non è convergente si dice divergente.*

Tale definizione necessita di alcuni chiarimenti. Consideriamo infatti un punto $t$ della discretizzazione (cioè tale che $t = t_n = t_0 + nh$), un metodo convergente deve essere tale che la soluzione numerica $y_n$ nel punto della discretizzazione $t = t_n$ tende a quella teorica $y(t)$ quando $h \to 0$. La definizione puntualizza l'esigenza che, anche se $h$ tende a zero (e quindi $n \to \infty$), la quantità $nh$ si mantiene costante all'ampiezza dell'intervallo $[t_0, t]$. Una definizione alternativa di convergenza richiede che

$$\lim_{h \to 0} \max_{0 \le n \le N} |y(t_n) - y_n| = 0$$

quando il metodo numerico viene applicato ad un qualsiasi probema ai valori iniziali che soddisfa le ipotesi del Teorema 5.1.1. Adesso l'obiettivo diventa quello di stabilire quali sono le condizioni che consentano di stabilire la convergenza del metodo. Un primo concetto è quello di consistenza.

### Consistenza

Sia $z(t) \in \mathcal{C}^1([t_0, T])$ e definiamo il seguente operatore differenza lineare:

$$\mathcal{L}[z(t); h] = \sum_{j=0}^{k} \left[\alpha_j z(t + jh) - h\beta_j z'(t + jh)\right].$$

Se calcoliamo tale operatore nella soluzione del problema di Cauchy assegnato in un punto della discretizzazione $t_n$ otteniamo un'espressione del residuo, detto anche errore locale di troncamento:

$$\begin{aligned} \tau_{n+k}(h) &= \sum_{j=0}^{k} \left[\alpha_j y(t_{n+j}) - h\beta_j y'(t_{n+j})\right] = \\ &= \sum_{j=0}^{k} \left[\alpha_j y(t_{n+j}) - h\beta_j f(t_{n+j}, y(t_{n+j}))\right]. \end{aligned}$$

Se le approssimazioni $y_n$ coincidessero con la soluzione teorica allora il residuo sarebbe nullo pertanto possiamo prendere tale quantità come possibile misura della qualità dell'approssimazione fornita dal metodo. Una condizione ovvia che il residuo deve soddisfare è che

$$\lim_{h\to 0} \tau_{n+k}(h) = 0$$

ma questo implicherebbe una condizione solo sui coefficienti $\alpha_j$, pertanto si preferisce la seguente definizione.

**Definizione 5.2.2** *Un metodo numerico si dice consistente (con il problema differenziale) se*

$$\lim_{h\to 0} \frac{\tau_{n+k}(h)}{h} = 0 \tag{5.9}$$

Per esplicitare le condizioni per la consistenza di un metodo cerchiamo un'espressione alternativa del residuo. Sostituiamo infatti al posto di $y(t_{n+j})$ e di $y'(t_{n+j})$ gli sviluppi in serie di Taylor (ipotizzando ovviamente che $y(t)$ sia una funzione di classe $\mathcal{C}^\infty([t_0, T])$):

$$\begin{aligned} y(t_{n+j}) &= y(t_n + jh) = \sum_{i=0}^{\infty} \frac{(jh)^i}{i!} y^{(i)}(t_n), \\ y'(t_{n+j}) &= y'(t_n + jh) = \sum_{i=0}^{\infty} \frac{(jh)^i}{i!} y^{(i+1)}(t_n). \end{aligned}$$

Ponendo

$$C_0 = \sum_{j=0}^{k} \alpha_j, \qquad C_i = \sum_{j=0}^{k} \left( \alpha_j \frac{j^i}{i!} - \beta_j \frac{j^{i-1}}{(i-1)!} \right)$$

si ha che il residuo $\tau_{n+k}(h)$ ammette uno sviluppo asintotico del tipo:

$$\tau_{n+k}(h) = C_0 y(t_n) + C_1 h y'(t_n) + C_2 h^2 y''(t_n) + C_3 h^3 y'''(t_n) + \dots.$$

Da tale proprietà segue che il metodo multistep può essere consistente solo se i primi due coefficienti dello sviluppo, $C_0$ e $C_1$, sono nulli, cioè:

$$\begin{aligned} C_0 &= \sum_{j=0}^{k} \alpha_j = 0 \\ C_1 &= \sum_{j=0}^{k} (\alpha_j j - \beta_j) = 0. \end{aligned}$$

In termini di coefficienti dei polinomi caratteristici

$$\begin{aligned} C_0 &= \sum_{j=0}^{k} \alpha_j = \rho(1) = 0 \\ C_1 &= \sum_{j=0}^{k} (\alpha_j j - \beta_j) = \rho'(1) - \sigma(1) = 0 \end{aligned}$$

Riassumendo abbiamo il seguente risultato:

**Teorema 5.2.1** *Un metodo multistep lineare avente $\rho(z)$ e $\sigma(z)$ come primo e secondo polinomi caratteristici, rispettivemente, è consistente se e solo se:*

$$\rho(1) = 0, \qquad \rho'(1) - \sigma(1) = 0.$$

Lo sviluppo in serie del residuo (o dell'errore locale di troncamente) induce un'altra considerazione. Poichè esso può essere considerato come una misura dell'errore commesso e di solito il passo di integrazione $h$ è una quantità relativamente piccola, possiamo ipotizzare che i metodi che forniscono risultati più accurati (a parità di passo) sono quelli il cui residuo dipende da potenze di $h$ di ordine più grande. Tale considerazione può essere formalizzata nella seguente definizione.

**Definizione 5.2.3** *Un metodo multistep lineare si dice di ordine $p$ se risulta $C_0 = C_1 = \cdots = C_p = 0$ e $C_{p+1} \neq 0$. La costante $C_{p+1}$ viene detta costante dell'errore.*

Se un metodo multistep ha ordine $p$ allora

$$\tau_{n+k}(h) = C_{p+1}h^{p+1}y^{(p+1)}(t_n) + C_{p+2}h^{p+2}y^{(p+2)}(t_n) + \dots$$

ovvero si scrive

$$\tau_{n+k}(h) = O(h^{p+1})$$

quindi dal punto di vista teorico i metodi che dovrebbero fornire un errore più piccolo sono quelli che hanno con ordine più elevato.
Osserviamo che le condizioni per la consistenza di un metodo multistep lineare sono equivalenti alla richiesta che il metodo numerico abbia almeno ordine $p = 1$. Abbiamo visto che la convergenza è un processo che avviene al limite, quando $h \simeq 0$, quindi può essere interessante studiare la soluzione numerica $y_n$ quando $h = 0$. Così facendo il metodo diventa

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = 0. \tag{5.10}$$

Tale equazione prende il nome di equazione alle differenze lineare a coefficienti costanti e può essere considerata come l'analogo discreto delle equazioni differenziali a coefficienti costanti. Per risolvere (5.10) si procede esattamente come nel caso delle analoghe equazioni differenziali a coefficienti costanti,

cioè si cerca una soluzione di tipo esponenziale. In questo caso specifico si cerca una soluzione del tipo

$$y_n = \xi^n$$

con $\xi \in \mathbb{C}$. Imponendo che tale $y_n$ sia soluzione di (5.10) segue

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = \sum_{j=0}^{k} \alpha_j \xi^{n+j} = \xi^n \sum_{j=0}^{k} \alpha_j \xi^j = \xi^n \rho(\xi).$$

Quindi $y_n = \xi^n$ è soluzione di (5.10) solo se $\xi$ è uno zero del polinomio $\rho(z)$. Se questo ammette $k$ radici distinte ognuna dà luogo ad una soluzione dell'equazione (5.10) pertanto, proprio in analogia con quanto accade nel caso delle equazioni differenziali, dette $\xi_1, \dots \xi_k$ tali radici, la soluzione generale della (5.10) è:

$$y_n = \sum_{j=1}^{k} c_j \xi_j^n$$

dove le $c_j$ sono delle costanti che dipendono dalle condizioni iniziali imposte su $y_n$. Nel caso in cui una delle radici $\xi$ abbia molteplicità doppia, cioè

$$\rho(\xi) = \rho'(\xi) = 0$$

allora si può provare che $y_n = n\xi^n$ è soluzione di (5.10). Infatti:

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = \sum_{j=0}^{k} \alpha_j (n+j)\xi^{n+j} = \xi^n \left[ n \sum_{j=0}^{k} \alpha_j \xi^j + \sum_{j=0}^{k} \alpha_j j \xi^j \right] =$$

$$= \xi^n \left[ n\rho(\xi) + \sum_{j=1}^{k} \alpha_j j \xi^j \right] = \xi^n \left[ n\rho(\xi) + \xi \sum_{j=1}^{k} \alpha_j j \xi^{j-1} \right] =$$

$$= \xi^n \left[ n\rho(\xi) + \xi\rho'(\xi) \right] = 0.$$

Consideriamo ora che, posto $h = 0$, è come se $y_n$ fosse la soluzione numerica del problema di Cauchy

$$y'(t) = 0, \qquad y(t_0) = y_0$$

che ammette come soluzione $y(t) = y_0$ per $t \geq t_0$. Poichè la soluzione teorica è limitata è naturale richiedere che anche la soluzione numerica lo sia. In base al discorso fatto sulle radici del polinomio $\rho(z)$ valgono le seguenti considerazioni:

1. Il valore $z = 1$ è radice del polinomio $\rho(z)$;

2. Se tutte le radici $\xi_j$ di $\rho(z)$ sono distinte e tali che

$$|\xi_j| \leq 1, \qquad j = 1, \ldots, k$$

allora la soluzione numerica $y_n$ si mantiene limitata (come quella teorica). Tale situazione resta invariata se $\rho(z)$ ha una radice multipla $\xi_j$ di modulo minore di 1, infatti in questo caso $n\xi_j$ è soluzione dell'equazione alle differenza 5.10 ma

$$\lim_{n\to\infty} n\xi_j^n = 0.$$

3. Se invece $\rho(z)$ ha una radice $\xi_j$ di modulo maggiore di 1 oppure di modulo 1 ma di molteplicità maggiore di 1 allora la soluzione $y_n$ non può mantenersi limitata.

Riassumiamo quanto appena detto nella seguente definizione.

**Definizione 5.2.4** *Un polinomio $\rho(z)$, di grado $k$, soddisfa il Criterio delle radici se le sue radici $\xi_1, \ldots, \xi_k$ sono tali che*

$$|\xi_i| \leq 1, \qquad i = 1, \ldots, k$$

*e ogni radice di modulo 1 è semplice.*

Se il primo polinomio caratteristico $\rho(z)$ associato ad un metodo multistep lineare soddisfa il criterio delle radici allora il metodo numerico si dice Zero-stabile, intendendo in questo modo che la soluzione numerica di una particolare equazione differenziale che ammette soluzione teorica limitata è a sua volta limitata. Le condizioni di Consistenza e Zero-stabilità di un metodo multistep lineare possono essere utilizzate per dimostrare il seguente risultato che enunciamo senza dimostrare.

**Teorema 5.2.2** *Un metodo multistep lineare è convergente se e solo se è consistente e zero-stabile.*

Avendo già fatto alcune considerazioni su consistenza e zero-stabilità possiamo enunciare il precedente risultato in modo equivalente:

**Teorema 5.2.3** *Un metodo multistep lineare (5.5) è convergente se e solo se:*

1. $\rho(1) = 0$;
2. $\rho'(1) - \sigma(1) = 0$;
3. *il polinomio* $\rho(z)$ *soddisfa la condizione delle radici.*

Concludiamo questo paragrafo sui metodi multistep lineari descrivendo in breve le classi di metodi più utilizzate.

### 5.2.1 Alcune classi di metodi multistep lineari

La classe più famosa di metodi multistep sono i Metodi di Adams, che hanno come primo polinomio caratteristico:

$$\rho(z) = z^k - z^{k-1}.$$

Tale polinomio soddisfa ovviamente il criterio delle radici perchè i suoi zeri sono $z = 1$ (radice semplice) e $z = 0$ (radice di molteplicità $k - 1$). Un metodo di Adams ha la forma:

$$y_{n+k} - y_{n+k-1} - h \sum_{j=0}^{k} \beta_j f_{n+j} = 0. \tag{5.11}$$

I metodi di Adams espliciti ($\beta_k = 0$) sono noti come Metodi di Adams-Bashforth, mentre quelli impliciti sono detti Metodi di Adams-Moulton. Il metodo di Adams-Bashforth ad un passo coincide con il metodo di Eulero Esplicito, mentre il metodo di Adams-Moulton ad un passo è il metodo dei Trapezi. I metodi di Adams sono tra i più antichi (erano noti già nel diciannovesimo secolo) eppure continuano ad essere utilizzati anche negli algoritmi più moderni.
Una seconda classe di metodi è definita dal primo polinomio caratteristico:

$$\rho(z) = z^k - z^{k-2}.$$

Anche questi metodi sono zero-stabili perchè il polinomio $\rho(z)$ ha come radici $z = \pm 1$ (radici semplici) e $z = 0$ (radice di molteplicità $k - 2$). I metodi espliciti di questa classe sono i Metodi di Nyström, quelli impliciti sono detti Metodi di Milne-Simpson Generalizzati. Un esempio di metodo di Nyström

è il Metodo del Midpoint Esplicito, mentre tra i metodi di Milne-Simpson Generalizzati, possiamo citare il Metodo di Simpson:

$$y_{n+2} - y_n - \frac{h}{3}\left(f_{n+2} + 4f_{n+1} + f_n\right) = 0.$$

Un'ultima classe di metodi multistep che vale la pena di citare sono le cosiddette Backward Differentiation Formulae (BDF in breve), che sono metodi impliciti definiti dal secondo polinomio caratteristico:

$$\sigma(z) = \beta_k z^k.$$

Per $k = 1$ il metodo appartenente a questa classe è il metodo di Eulero Implicito, per $k = 2$ abbiamo il seguente metodo che non ha un nome particolare:

$$y_{n+2} - \frac{4}{3}y_{n+1} + \frac{1}{3}y_n - \frac{2}{3}hf_{n+2} = 0,$$

mentre per $k = 3$ abbiamo:

$$y_{n+3} - \frac{18}{11}y_{n+2} + \frac{9}{11}y_{n+1} - \frac{2}{11}y_n - \frac{6}{11}hf_{n+3} = 0.$$

### 5.2.2 Il Metodo di Newton per Sistemi non Lineari

Supponiamo che sia $\Omega$ un sottoinsieme di $\mathbb{R}^n$ e che siano assegnate le $n$ funzioni

$$f_i : \Omega \to \mathbb{R}, \qquad i = 1, \ldots, n.$$

Ogni vettore $\mathbf{x} \in \mathbb{R}^n$, soluzione del sistema non lineare di $n$ equazioni in $n$ incognite

$$\begin{aligned} &f_1(x_1, x_2, \ldots, x_n) = 0 \\ &f_2(x_1, x_2, \ldots, x_n) = 0 \\ &\vdots \\ &f_n(x_1, x_2, \ldots, x_n) = 0 \end{aligned} \tag{5.12}$$

prende il nome di radice dell'equazione

$$F(\mathbf{x}) = 0 \tag{5.13}$$

oppure di zero della funzione vettoriale

$$F : \Omega \to \mathbb{R}^n$$

dove il vettore $F(\mathbf{x})$ è definito da:

$$F(\mathbf{x}) = \begin{pmatrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) \end{pmatrix}.$$

Come nel caso scalare l'equazione $F(\mathbf{x}) = 0$ viene trasformata in un problema del tipo

$$\mathbf{x} = \Phi(\mathbf{x}) \tag{5.14}$$

ovvero

$$x_i = \Phi_i(x_1, x_2, \ldots, x_n), \qquad i = 1, 2, \ldots, n$$

con $\Phi(\mathbf{x})$ funzione definita in $\Omega$ e scelta in modo tale che le proprietà richiesta ad $F(\mathbf{x})$ si trasferiscano su $\Phi$, cioè anch'essa deve essere continua in un opportuno intorno della radice e calcolabile nell'insieme di definizione. Il motivo di tali richieste è che la funzione $\Phi(\mathbf{x})$ viene utilizzata per definire una successione di vettori nel seguente modo. Sia $\mathbf{x}^{(0)}$ un vettore iniziale appartenente a $\Omega$ e definiamo la seguente successione

$$\mathbf{x}^{(k+1)} = \Phi(\mathbf{x}^{(k)}), \qquad k = 0, 1, 2, 3, \ldots \tag{5.15}$$

ovvero

$$x_i^{(k+1)} = \Phi_i(x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)}), \qquad i = 1, 2, \ldots, n.$$

La funzione $\Phi(\mathbf{x})$ prende il nome di funzione iteratrice dell'equazione non lineare $F(\mathbf{x}) = 0$. Ricordiamo che un vettore $\mathbf{x}^*$ che soddisfa la (5.14) viene detto punto fisso di $\Phi(\mathbf{x})$ (oppure punto unito). La successione dei vettori $\mathbf{x}^{(k)}$ definisce il metodo delle approssimazioni successive per il calcolo appunto di tale punto fisso. Quello che si richiede a tale successione è che essa converga al vettore $\mathbf{x}^*$, soluzione del sistema non lineare. In questo caso per convergenza si intende che

$$\lim_{k\to\infty} \mathbf{x}^{(k)} = \mathbf{x}^*$$

cioè, in termini di componenti,

$$\lim_{k\to\infty} \mathbf{x}_i^{(k)} = \mathbf{x}_i^*.$$

Per la convergenza del metodo delle approssimazioni successive vale quindi il seguente teorema.

**Teorema 5.2.4** *Se la funzione* $\Phi(\mathbf{x})$ *è differenziabile con continuità in un intorno del punto fisso* $\mathbf{x}^*$*, e risulta*

$$\rho(J_\Phi(\mathbf{x}^*)) < 1$$

*allora, scelto* $\mathbf{x}^{(0)}$ *appartenente a tale intorno, la successione costruita con il metodo delle approssimazioni successive è convergente a* $\mathbf{x}^*$*.*

Se si conosce abbastanza bene l'approssimazione iniziale della soluzione del sistema di equazioni

$$F(\mathbf{x}) = 0$$

il metodo di Newton risulta molto efficace. Il Metodo di Newton per risolvere il sistema (5.13) può essere derivato in modo semplice come segue. Sia $\mathbf{x}^{(k)}$ una buona approssimazione a $\mathbf{x}^*$, soluzione di $F(\mathbf{x}) = 0$, possiamo allora scrivere lo sviluppo in serie della funzione $F$ valutata nella soluzione del sistema non lineare prendendo come punto iniziale proprio il vettore $\mathbf{x}^{(k)}$ :

$$0 = F(\mathbf{x}^*) = F(\mathbf{x}^{(k)}) + J_F(\xi_k)(\mathbf{x}^* - \mathbf{x}^{(k)})$$

dove

$$\xi_k = \theta\mathbf{x}^* + (1-\theta)\mathbf{x}^{(k)} \qquad 0 \leq \theta \leq 1$$

ovvero, suppondendo che lo Jacobiano sia invertibile,

$$\mathbf{x}^* - \mathbf{x}^{(k)} = -J_F^{-1}(\xi_k)F(\mathbf{x}^{(k)}) \Rightarrow \mathbf{x}^* = \mathbf{x}^{(k)} - J_F^{-1}(\xi_k)F(\mathbf{x}^{(k)}). \tag{5.16}$$

Se $\mathbf{x}^{(k)}$ è abbastanza vicino a $\mathbf{x}^*$ allora possiamo confondere $\mathbf{x}^{(k)}$ con $\xi_k$; in tal modo però (5.16) non fornirà esattamente $\mathbf{x}^*$ ma un valore ad esso vicino, diciamo $\mathbf{x}^{(k+1)}$. Abbiamo quindi lo schema iterativo:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - J_F^{-1}(\mathbf{x}^{(k)})F(\mathbf{x}^{(k)}). \tag{5.17}$$

Il discorso può essere ripetuto ora partendo da $\mathbf{x}^{(k+1)}$; dunque $k$ può essere visto come un indice di iterazione. La (5.17) è appunto la formula che definisce il metodo di Newton.
Può essere interessante soffermarsi su alcuni dettagli di implementazione del metodo (5.17). Poniamo infatti

$$\mathbf{z}^{(k)} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$$

e osserviamo che, moltiplicando per la matrice $J_F(\mathbf{x}^{(k)})$ l'espressione del metodo di Newton diventa

$$J_F(\mathbf{x}^{(k)})\mathbf{z}^{(k)} = -F(\mathbf{x}^{(k)})$$

da cui, risolvendo il sistema lineare che ha $J_F(\mathbf{x}^{(k)})$ come matrice dei coefficienti e $-F(\mathbf{x}^{(k)})$ come vettore dei termini noti si può ricavare il vettore $z^{(k)}$ e ottenere il vettore al passo $k+1$:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{z}^{(k)}.$$

L'algoritmo, ad un generico passo $k$, può essere così riassunto:

1. Calcolare la matrice $J_F(\mathbf{x}^{(k)})$ e il vettore $-F(\mathbf{x}^{(k)})$;
2. Risolvere il sistema lineare $J_F(\mathbf{x}^{(k)})\mathbf{z}^{(k)} = -F(\mathbf{x}^{(k)})$;
3. Calcolare il vettore $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{z}^{(k)}$;
4. Valutare la convergenza: fissata una tolleranza $\varepsilon$, se risulta

   $$\|\mathbf{z}^{(k)}\| \leq \varepsilon$$

   allora $\mathbf{x}^{(k+1)}$ è una buona approssimazione della soluzione, altrimenti si ritorna al passo 1.

Occorre evidenziare subito come il punto 2. richieda un notevole costo computazionale. Infatti se il sistema in questione viene risolto utilizzando la fattorizzazione $LU$ ad ogni passo si deve calcolare tale fattorizzazione e poi procedere alla risoluzione di due sistemi triangolari. Per diminuire tale costo si potrebbe procedere nel seguente modo. Anzichè calcolare la matrice al punto 1. ad ogni $k$ si potrebbe calcolarla ogni $m$ iterate, cosicchè si calcolerebbe la fattorizzazione $LU$ alla $m-$esima iterazione e ad ogni passo per risolvere il sistema al punto 2. si risolverebbero solo i due sistemi triangolari. L'algoritmo potrebbe essere riassunto nel seguente modo:

1. Se $k = qm$ allora calcolare la matrice $A = J_F(\mathbf{x}^{(k)})$ e la sua fattorizzazione $A = LU$;
2. Calcolare il vettore $-F(\mathbf{x}^{(k)})$;
3. Risolvere il sistema triagolare inferiore $L\mathbf{w}^{(k)} = -F(\mathbf{x}^{(k)})$;

4. Risolvere il sistema triagolare superiore $U\mathbf{z}^{(k)} = \mathbf{w}^{(k)}$;
5. Calcolare il vettore $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \mathbf{z}^{(k)}$;
6. Valutare la convergenza, come al punto 4. del metodo di Newton.

Questo metodo viene detto Metodo di Newton Modificato (o Congelato) e fa parte della classe dei metodi cosiddetti quasi-Newton in quanto derivati da tale metodo.

## 5.3 Esercitazione MatLab

In questo paragrafo riportiamo invece i codici MatLab per la risoluzione approssimata di sistemi di equazioni differenziali utilizzando alcuni metodi, sia espliciti che impliciti, descritti nel capitolo precedente. Il primo è quello che utilizza il metodo di Eulero esplicito. I parametri di input sono:

- `f`: Funzione che definisce il problema di Cauchy;
- `t0`: Estremo sinistro dell'intervallo di integrazione;
- `T`: Estremo destro dell'intervallo di integrazione;
- `y0`: Condizione iniziale del problema di Cauchy;
- `N`: Numero di sottointervalli in cui è stato suddiviso l'intervallo di integrazione.

In uscita la routine produce il vettore `tn` degli istanti in cui è stata approssimata la soluzione e la variabile `yn`, che contiene le approssimazioni della soluzione $y(t)$ nei punti $t_n$. Poichè il codice qui riportato va bene anche per sistemi differenziali, la variabile `yn` è una matrice avente come numero di righe la dimensione del sistema differenziale e come numero di colonne il numero di istanti di tempo in cui è stata approssimata la soluzione. La prima colonna di `yn` coincide con il vettore colonna `y0`.

```
function [tn,yn]=eulero_esp(f,t0,T,y0,N)
%
% [tn,yn]=eulero_esp(f,t0,T,y0,N)
%
```

```
% Calcola l'approssimazione della soluzione del
% problema di Cauchy con il metodo di Eulero Esplicito
%
% f = funzione che definisce il problema di Cauchy
% t0 = istante iniziale di tempo
% T = istante finale di tempo
% y0 = condizione iniziale
% N = numero di sottointervalli
%
tn = linspace(t0,T,N+1);
h = (T-t0)/N;
yn(:,1) = y0;
for n = 2:N+1
   yn(:,n) = yn(:,n-1)+h*feval(f,tn(n-1),yn(:,n-1));
end
return
```

Come abbiamo detto in precedenza i codici che stiamo descrivendo vanno bene anche per sistemi differenziali. Nel caso di metodi impliciti abbiamo visto che è necessario utilizzare un metodo di iterazione tipo approssimazioni successive per calcolare la soluzione $y_{n+1}$. Il seguente codice applica tale metodo esattamente come descritto in (5.7) in cui:

- `f`: Funzione che definisce il problema di Cauchy;
- `t`: Istante di tempo in cui si approssima la soluzione, cioè $t_{n+k}$;
- `y0`: Approssimazione iniziale, cioè $y_{n+k-1}$;
- `h`: Passo di integrazione $h$;
- `betak`: Coefficiente $\beta_k$ del metodo utilizzato;
- `g`: Termine noto del metodo, cioè $G_{n,k}$.

Il metodo fissa un numero massimo di iterazioni pari a 100 per raggiungere la condizione di convergenza che è:

$$\|y_{n+k}^{[m]} - y_{n+k}^{[m-1]}\|_\infty \leq 10^{-10}.$$

Se il numero di iterazioni supera il valore 100 viene segnalata la mancata convergenza attraverso un messaggio di errore.

```
function y1=fixedpoint(f,t,y0,h,betak,g)
%
% y1=fixedpoint(f,t,y0,h,betak,g)
%
% Calcola l'approssimazione al passo n+1 con
% il metodo delle approssimazioni successive
%
% f = funzione che definisce il problema di Cauchy
% t = istante di tempo
% h = passo di integrazione
% betak = coefficiente del metodo multistep
% y0 = approssimazione al passo precedente
% g = termine noto del sistema non lineare
%
epsilon = 1e-10;
y1 = zeros(size(y0));
for k = 1:100
    y1 = h*betak*feval(f,t,y0)+g;
    err = norm(y1-y0,'inf');
    if err <= epsilon
       return
    else
       y0 = y1;
    end
end
error('Mancata convergenza')
```

La routine `fixedpoint` viene utilizzata nei due seguenti codici che implementano i metodi impliciti di Eulero Implicito e dei Trapezi. Ciò che cambia sono solo i parametri del metodo di iterazione funzionale. Infatti nel caso di Eulero Implicito il metodo diviene:

$$\begin{aligned} y_{n+1}^{[0]} &= y_n \\ y_{n+1}^{[m+1]} &= hf(t_{n+1}, y_{n+1}^{[m]}) + y_n. \end{aligned}$$

e risulta

$$\beta_k = 1, \qquad G_{n,k} = y_n.$$

I parametri di ingresso (e uscita) della routine sono gli stessi visti nel metodo di Eulero Esplicito. È bene osservare che nel codice l'indice `n` viene shiftato di uno in avanti poichè, come noto, la numerazione dei vettori, in MatLab, inizia dall'indice 1.

```
function [tn,yn]=eulero_imp(f,t0,T,y0,N)
%
% [tn,yn]=eulero_imp(f,t0,T,y0,N)
%
% Calcola l'approssimazione della soluzione del
% problema di Cauchy con il metodo di Eulero Implicito
%
% f = funzione che definisce il problema di Cauchy
% t0 = istante iniziale di tempo
% T = istante finale di tempo
% y0 = condizione iniziale
% N = numero di sottointervalli
%
tn = linspace(t0,T,N+1);
h = (T-t0)/N;
yn(:,1) = y0;
for n = 2:N+1
   yn(:,n) = fixedpoint(f,tn(n),yn(:,n-1),h,1,yn(:,n-1));
end
return
```

Il metodo dei Trapezi è descritto dalla formula

$$y_{n+1} = y_n + \frac{h}{2}\left[f(t_{n+1}, y_{n+1}) + f(t_n, y_n)\right].$$

In questo caso il metodo di iterazione funzionale diviene:

$$\begin{aligned} y_{n+1}^{[0]} &= y_n \\ y_{n+1}^{[m+1]} &= \frac{h}{2} f(t_{n+1}, y_{n+1}^{[m]}) + y_n + \frac{h}{2} f(t_n, y_n). \end{aligned}$$

e risulta

$$\beta_k = \frac{1}{2}, \qquad G_{n,k} = y_n + \frac{h}{2} f(t_n, y_n).$$

```
function [tn,yn]=trapezi(f,t0,T,y0,N)
%
% [tn,yn]=trapezi(f,t0,T,y0,N)
%
% Calcola l'approssimazione della soluzione del
% problema di Cauchy con il metodo dei Trapezi
%
% f = funzione che definisce il problema di Cauchy
% t0 = istante iniziale di tempo
% T = istante finale di tempo
% y0 = condizione iniziale
% N = numero di sottointervalli
%
tn = linspace(t0,T,N+1);
h = (T-t0)/N;
yn(:,1) = y0;
for n = 2:N+1
    g = yn(:,n-1) + h/2 * feval(f,tn(n-1),yn(:,n-1));
    yn(:,n) = fixedpoint(f,tn(n),yn(:,n-1),h,0.5,g);
end
return
```

Come esempio di applicazione di tali routine abbiamo considerato il seguente sistema differenziale del secondo ordine:

$$\begin{aligned} x'(t) &= \alpha x(t) + \beta x(t)y(t) \\ \\ y'(t) &= \gamma y(t) + \delta x(t)y(t) \end{aligned}$$

dove $\alpha = 2$, $\gamma = -1$, $\beta = -0.001$ e $\delta = 0.001$, e condizioni iniziali $x(0) = 15$ e $y(0) = 22$. Il seguente è il codice MatLab che definisce tale funzione. Osserviamo che sia il vettore `y` in ingresso che quello di uscita `y1` hanno le medesime dimensioni.

```
function y1=f(t,y)
y1 = zeros(size(y));
y1(1) = 2*y(1)-0.001*y(1)*y(2);
y1(2) = -y(2)+0.001*y(1)*y(2);
return
```

Per integrare il sistema nell'intervallo $[0, 20]$, prendendo $N = 1000$, e applicando tutti i metodi implementati, si devono scrivere le seguenti righe di codice MatLab:

```
>> y0 = [15 ; 22];
>> [tn,yn_EE] = eulero_esp(@f,0,20,y0,1000);
>> semilogy(tn,yn_EE(1,:),'b',tn,yn_EE(2,:),'r');
>> [tn,yn_EI] = eulero_imp(@f,0,20,y0,1000);
>> semilogy(tn,yn_EI(1,:),'b',tn,yn_EI(2,:),'r');
>> [tn,yn_TR] = trapezi(@f,0,20,y0,1000);
>> semilogy(tn,yn_TR(1,:),'b',tn,yn_TR(2,:),'r');
```

Nelle Figure 5.1-5.3 sono tracciati i grafici ottenuti con i 3 metodi descritti. Nella Figura 5.4 sono tracciate le orbite, nel piano $(x, y)$ delle soluzioni numeriche (verde=Eulero Esplicito, rosso=Eulero Implicito, verde=Metodo dei Trapezi) a partire dalla condizione iniziale evidenziata con `'o'`. Osserviamo che la soluzione numerica del metodo di Eulero esplicito tende a divergere, quella di Eulero implicito converge verso un punto interno, mentre quella descritta dal metodo dei Trapezi torna su se stessa, cioè è di tipo periodico. Uno studio teorico della soluzione metterebbe in evidenza che proprio quest'ultima è una caratteristica delle funzioni $(x(t), y(t))$. Questo perchè il metodo dei Trapezi ha alcune proprietà di convergenza migliori rispetto agli altri metodi (oltre ad un ordine superiore) legate al concetto di stabilità di un metodo che non è possibile approfondire oltre.

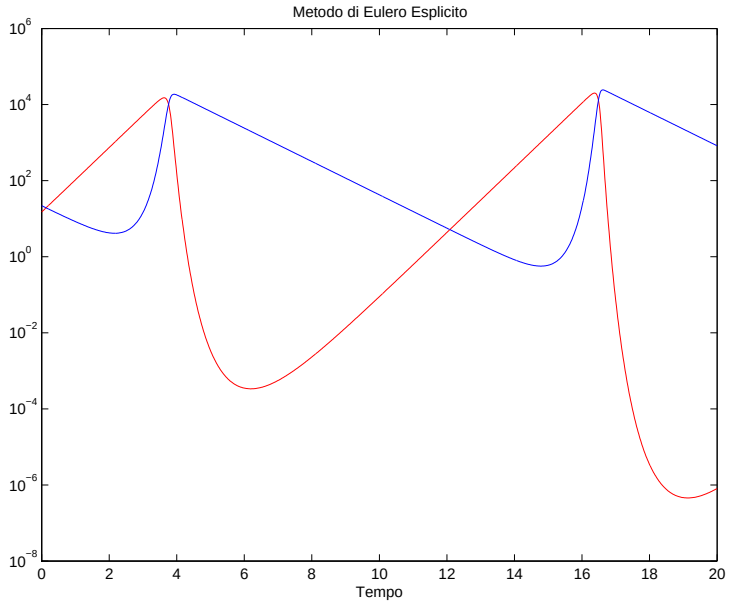



Figura 5.1: Approssimazione numerica con il metodo di Eulero Esplicito

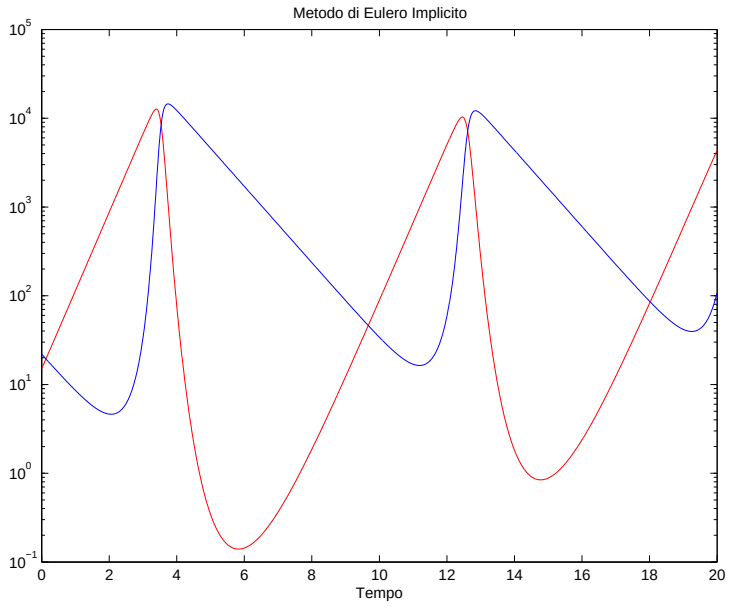



Figura 5.2: Approssimazione numerica con il metodo di Eulero Implicito

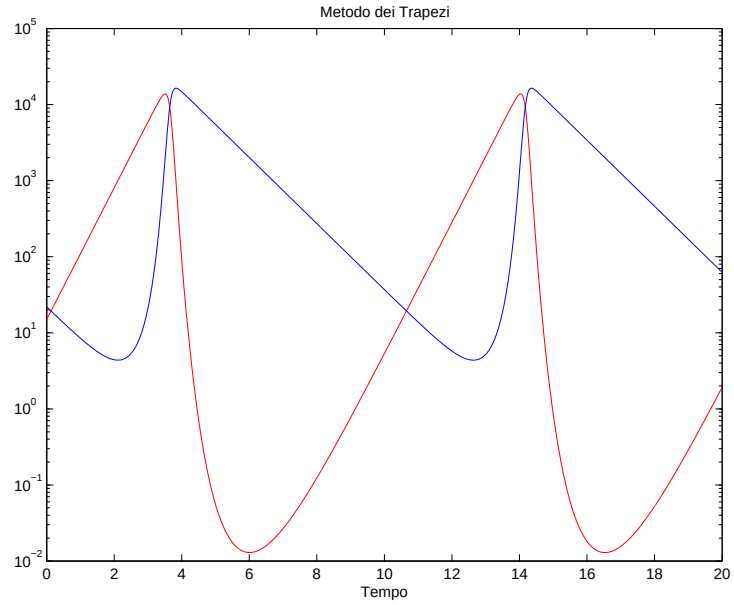



Figura 5.3: Approssimazione numerica con il metodo dei Trapezi

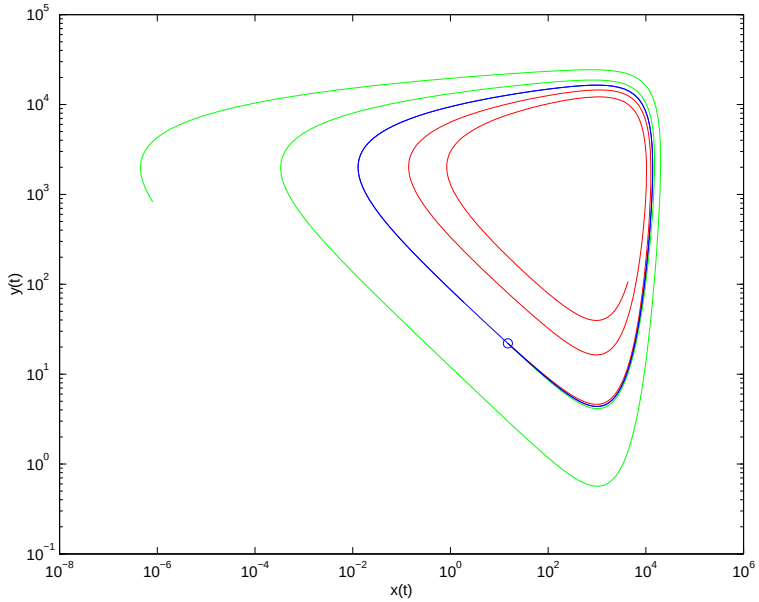



Figura 5.4: Sovrapposizione delle orbite ottenute con i 3 diversi metodi