# Università degli Studi di Padova

Dipartimento di ingegneria civile, edile e ambientale

*Department of civil, enviromental and architectural engineering*

Corso di laurea in ingegneria civile



Tesi di laurea

# Calcolo degli autovalori di matrici simmetriche definite positive con applicazioni


Relatore: Chiar.mo Prof. Massimiliano Ferronato

Laureando: Marco Pilotto


**Anno accademico 2021–2022**

A chi crede in me

# Indice

# Introduzione

Il problema della ricerca degli autovalori di una matrice quadrata $A$ è di fondamentale importanza in diversi ambiti dell'ingegneria.

Autovalori ed autovettori sono fondamentali nell'analisi della stabilità delle strutture, nella determinazione delle frequenze proprie di oscillazione del sistema. Altri esempi sono nel campo della statistica, dell'economia o negli algoritmi di page ranking utilizzati dai browser web.

La dimensione e la complessità di questi problemi rendono impraticabile una soluzione analitica, l'unica possibilità è risolvere numericamente il problema utilizzando opportuni algoritmi.

Nel seguente elaborato si affronta la determinazione dello spettro di matrici simmetriche definite positive. In particolare, si considereranno matrici che scaturiscono dalla discretizzazione di equazioni differenziali alla derivate parziali di interesse specifico per applicazioni nell'ambito dell'ingegneria civile.

Data una matrice $A$ simmetrica, il problema della ricerca degli autovalori consiste nella determinazione delle coppie $(\lambda, \vec{v})$,con $\lambda \in \mathbb{R}$ e $\vec{v} \in \mathbb{R}^n$, tali che:

$$A\vec{v} = \lambda\vec{v} \tag{1}$$

Il polinomio caratteristico associato ad $A$, descritto dalla seguente equazione, ha come radici proprio gli autovalori della matrice.

$$det(A - \lambda I) = 0 \tag{2}$$

Una volta determinati i coefficienti del polinomio caratteristico è possibile calcolarne le radici, come vedremo però seguire questo procedimento comporta alcune difficoltà.

Per cominciare assumiamo che gli $n$ autovettori di $A$ siano linearmente indipendenti e che vi corrispondano $n$ diversi autovalori. Posso sempre esprimere un generico vettore $\vec{z}_0 \in \mathbb{R}^n$ come combinazione lineare degli autovettori della matrice $A$.

$$\vec{z}_0 = c_1\vec{v}_1 + c_2\vec{v}_2 + \cdots + c_n\vec{v}_n \tag{3}$$

Moltiplicando per $A$ più volte il vettore iniziale ottengo:

$$\begin{aligned}
\vec{z}_1 &= A\vec{z}_0 = c_1\lambda_1\vec{v}_1 + c_2\lambda_2\vec{v}_2 + \cdots + c_n\lambda_n\vec{v}_n \\
\vec{z}_2 &= A\vec{z}_1 = A^2\vec{z}_0 = c_1\lambda_1^2\vec{v}_1 + c_2\lambda_2^2\vec{v}_2 + \cdots + c_n\lambda_n^2\vec{v}_n \\
&\vdots \\
\vec{z}_n &= A\vec{z}_{n-1} = A^n\vec{z}_0 = c_1\lambda_1^n\vec{v}_1 + c_2\lambda_2^n\vec{v}_2 + \cdots + c_n\lambda_n^n\vec{v}_n
\end{aligned} \tag{4}$$

Lo spazio vettoriale in cui si trovano i vettori della successione ha dimensione $n$, ciò significa che necessariamente i vettori generati saranno linearmente dipendenti. Una loro combinazione lineare darà come risultato il vettore nullo

$$a_0\vec{z}_0 + a_1\vec{z}_1 + \cdots + a_{n-1}\vec{z}_{n-1} + \vec{z}_n = 0 \tag{5}$$

L'equazione (5) da luogo ad un sistema lineare dove le incognite sono i coefficienti $a_0, a_1, \cdots, a_{n-1}$. Esprimendo i vettori $\vec{z}$ come nella (4), l'equazione (5) diviene:

$$(a_0+a_1\lambda_1+a_2\lambda_1^2+\cdots+\lambda_1^n)c_1\vec{v}_1+\cdots+(a_0+a_1\lambda_n+a_2\lambda_n^2+\cdots+\lambda_n^n)c_n\vec{v}_n = 0 \tag{6}$$

Ponendo $P(\lambda)$ come:

$$P(\lambda) = a_0 + a_1\lambda + a_2\lambda^2 + \cdots + \lambda^n \tag{7}$$

Riscrivo la (6) come:

$$P(\lambda_1)c_1\vec{v}_1 + P(\lambda_2)c_2\vec{v}_2 + \cdots + P(\lambda_n)c_n\vec{v}_n = 0 \tag{8}$$

Dato che i vettori $\vec{v}_1, \vec{v}_2, \cdots, \vec{v}_n$ sono linearmente indipendenti e che i coefficienti $c_1, c_2, \cdots, c_n$ sono le componenti del vettore $\vec{z}_0$ scelto arbitrariamente, la (8) risulta nulla se e solo se:

$$P(\lambda_1) = P(\lambda_2) = \cdots = P(\lambda_n) = 0 \tag{9}$$

$P(\lambda)$ è il polinomio caratteristico di $A$ in quanto esso ha come radici gli autovalori della matrice.

Determinare gli autovalori della matrice $A$ trovando gli zeri del polinomio caratteristico può diventare un'operazione piuttosto complessa per matrici molto grandi. Per prima cosa devo trovare i coefficienti $a_0, a_1, \cdots, a_{n-1}$ risolvendo il sistema (5). Le ultime colonne del sistema tenderanno ad essere parallele, infatti la componente sul primo autovettore prevale sulle altre dopo pochi passi rendendo i vettori della successione $\vec{z}_k$ circa paralleli, il problema quindi è mal-condizionato. Supponendo di riuscire a calcolare con precisione i coefficienti dal sistema lineare rimane da risolvere il polinomio caratteristico. Percorrere questa via risulta computazionalmente sconveniente.

Il problema del calcolo degli autovalori di una matrice simmetrica definita positiva (SDP) di grandi dimensioni va quindi affrontato diversamente, mediante l'utilizzo di opportuni algoritmi numerici che dipendono dal tipo e dal numero di autovalori cercati.

Spesso l'interesse ricade solamente verso la ricerca degli autovalori alle estremità dello spettro. In questo caso conviene utilizzare algoritmi come il metodo delle potenze oppure il gradiente coniugato applicato al quoziente di Rayleigh, essi servono rispettivamente a trovare l'autovalore di modulo massimo $\lambda_1$ e quello di modulo minimo $\lambda_n$. La tecnica della deflazione, invece, fa utilizzo di questi due algoritmi opportunamente modificati per andare a determinare più autovalori appartenenti alle estremità dello spettro. Quest'ultima tecnica risulta sconveniente se l'interesse è quello di determinare l'intero insieme di autovalori di $A$, in questi casi si preferisce usare metodi più sofisticati ed efficienti come l'algoritmo QR che permettono di determinare l'intero spettro di $A$. Il metodo QR applicato a matrici arbitrarie può risultare molto dispendioso, si preferisce operare una similitudine tra matrici in modo tale da

ricavare una matrice $T$ tri-diagonale con gli stessi autovalori di $A$. Il metodo QR applicato ad una matrice tri-diagonale risulta molto più efficiente.

Il presente lavoro di tesi è strutturato come segue. Nel capitolo 1 si parla della ricerca degli autovalori estremi di una matrice SDP, si introducono il metodo delle potenze e il metodo del gradiente coniugato applicato al quoziente di Rayleigh, mentre nel capitolo 2, introducendo il metodo QR, l'interesse si estende all'intero spettro di $A$. Infine nel capitolo 3 si applicano i metodi implementati ad una matrice derivante dalla soluzione di un problema di ingegneria civile.

# Capitolo 1

# Calcolo degli autovalori estremi

## 1.1 Metodo delle potenze

Il metodo delle potenze è un algoritmo iterativo utilizzato per trovare l'autovalore di modulo massimo $\lambda_1$ e corrispondente autovettore.

Supponiamo di avere una matrice A di ordine n con gli autovalori ordinati da $\lambda_1$ a $\lambda_n$, rispettivamente autovalore di modulo massimo e minimo, e con gli autovettori corrispondenti linearmente indipendenti. Creiamo quindi una successione di vettori $\vec{z}_k \in \mathbb{R}^n$ tali che $\vec{z}_{k+1} = A\vec{z}_k$ partendo da un vettore arbitrario $\vec{z}_0$ di modulo unitario. Riscrivendo il vettore $\vec{z}_k$ come combinazione lineare degli autovettori di A ottengo:

$$\vec{z}_k = c_1 A_1^k \vec{v}_1 + c_2 A_2^k \vec{v}_2 + \cdots + c_n A_n^k \vec{v}_n \tag{1.1}$$

Ricordando che $\lambda_i v_i = A v_i$

$$\vec{z}_k = c_1 \lambda_1^k \vec{v}_1 + c_2 \lambda_2^k \vec{v}_2 + \cdots + c_n \lambda_n^k \vec{v}_n \tag{1.2}$$

dove $c_1, \cdots, c_n$ sono i coefficienti della combinazione lineare.
Ora raccogliendo $\lambda_1^k$ ottengo:

$$\vec{z}_k = \lambda_1^k \left[ c_1 \vec{v}_1 + c_2 \left( \frac{\lambda_2}{\lambda_1} \right)^k \vec{v}_2 + \cdots + c_n \left( \frac{\lambda_n}{\lambda_1} \right)^k \vec{v}_n \right] \tag{1.3}$$

Dato che $\lambda_1 > \lambda_i$ per ogni $i = 2, \cdots, n$ quando $k \to \infty$ i termini che moltiplicano gli autovettori diversi da $\vec{v}_1$ tenderanno a 0 e quindi $\vec{z}_k \simeq c_1 \lambda_1^k \vec{v}_1$. Dunque, con $k$ sufficientemente grande, $\vec{z}_k \to \vec{v}_1$, ovvero tende all'autovettore associato a $\lambda_1$. Il modulo di $\lambda_1$ si trova come limite del rapporto tra le norme di due vettori consecutivi della successione $\vec{z}_k$

$$|\lambda_1| = \lim_{k \to \infty} \frac{\| \vec{z}_{k+1} \|}{\| \vec{z}_k \|} \tag{1.4}$$

A seguire viene riportata l'implementazione dell'algoritmo del metodo delle potenze (algoritmo 1).

**Algorithm 1** Metodo delle potenze

**procedure** POWERMETHOD(A,n,toll)
▷ Preparo valori di partenza
$\vec{z} = zeros(n, 1);\ \vec{z}(1) = 1;$
$iter = 0;\quad itmax = 300;\quad \lambda = 0;$

**while** $err \geq toll$, $iter \leq itmax$ **do**
$iter = iter + 1;$
$\vec{z}_{k+1} = A\vec{z}_k;$ ▷ Calcolo il nuovo vettore della sequenza
$\alpha = norm(\vec{z}_{k+1});$
$\vec{z}_k = \vec{z}_{k+1}/\alpha;$
$\lambda_1 = \alpha;$
$err = abs(\lambda_1 - \lambda)/\lambda;$ ▷ Controllo sulla convergenza
$\lambda = \lambda_1;$
**end while**
**end procedure**

Se la matrice A è simmetrica e definita positiva è possibile fare un'ulteriore osservazione.

Tenendo presente che gli autovettori di una matrice simmetrica sono ortogonali, possiamo scalarli in modo che essi abbiamo norma-2 unitaria ed eseguiamo la seguente operazione:

$$\vec{z}_k^T \vec{z}_k = \lambda_1^{2k}\left[c_1^2 + c_2^2\left(\frac{\lambda_2}{\lambda_1}\right)^{2k} + \cdots + c_n^2\left(\frac{\lambda_n}{\lambda_1}\right)^{2k}\right] \tag{1.5}$$

L'approssimazione di $\lambda_1$ si ricava dal seguente rapporto:

$$|\lambda_1| = \lim_{k\to\infty} \frac{\sqrt{\vec{z}_{k+1}^T \vec{z}_{k+1}}}{\sqrt{\vec{z}_k^T \vec{z}_k}} = \lim_{k\to\infty} \frac{\| \vec{z}_{k+1} \|_2}{\| \vec{z}_k \|_2} \tag{1.6}$$

Come è visibile dal confronto tra la (1.3) e la (1.5), nel secondo caso la convergenza è più rapida in quanto i termini sono elevati a $2k$ e tenderanno a zero più velocemente. Dato che il rapporto definito dalla (1.6) non è altro che un rapporto tra le norme-2 dei vettori $\vec{z}_{k+1}$ e $\vec{z}_k$. Per matrici simmetriche, conviene utilizzare quest'ultima norma nell'equazione (1.4) per velocizzare la convergenza.

Al fine di capire quanto più rapida sarà la convergenza, basta osservare dalla (1.3) che $\vec{z}_k \to \vec{v}_1$ con un errore il cui termine dominante è $(\lambda_2/\lambda_1)^k$. Esprimiamo quindi l'errore come:

$$\epsilon_k = C\left(\frac{\lambda_2}{\lambda_1}\right)^k \tag{1.7}$$

Dove C è una costante. Applicando il logaritmo la (1.7) diventa:

$$\log \epsilon_k = \log C + k \log\left(\frac{\lambda_2}{\lambda_1}\right) \tag{1.8}$$

Se $A$ è simmetrica definita positiva e nel metodo delle potenze si utilizza la norma-2 nella (1.8) il secondo addendo diventerà $2k \log(\lambda_2/\lambda_1)$ e la velocità di convergenza sarà doppia.

Al fine di verificare il funzionamento del metodo proposto si costruisce una matrice test $C$ $(50 \times 50)$ simmetrica e definita positiva. È possibile conoscere gli autovalori esatti di $C$ utilizzando la funzione $eig(C)$ su Matlab. La stessa matrice test verrà utilizzata anche nei paragrafi successivi.

$$C = \begin{bmatrix} 5 & 5 & 1 & & & & & & \\ 5 & 9 & 5 & 1 & & & & & \\ 1 & 5 & 9 & 5 & 1 & & & & \\ & 1 & 5 & 9 & 5 & 1 & & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & 1 & 5 & 9 & 5 & 1 & \\ & & & & 1 & 5 & 9 & 5 & 1 \\ & & & & & 1 & 5 & 9 & 5 \\ & & & & & & 1 & 5 & 5 \end{bmatrix} \tag{1.9}$$

Applicando l'algoritmo 1 alla matrice $C$ si vuole ricavare l'autovalore di modulo massimo $\lambda_1$. In figura 1.1 visualizziamo graficamente l'andamento ad ogni iterazione del valore approssimato per $\lambda_1$

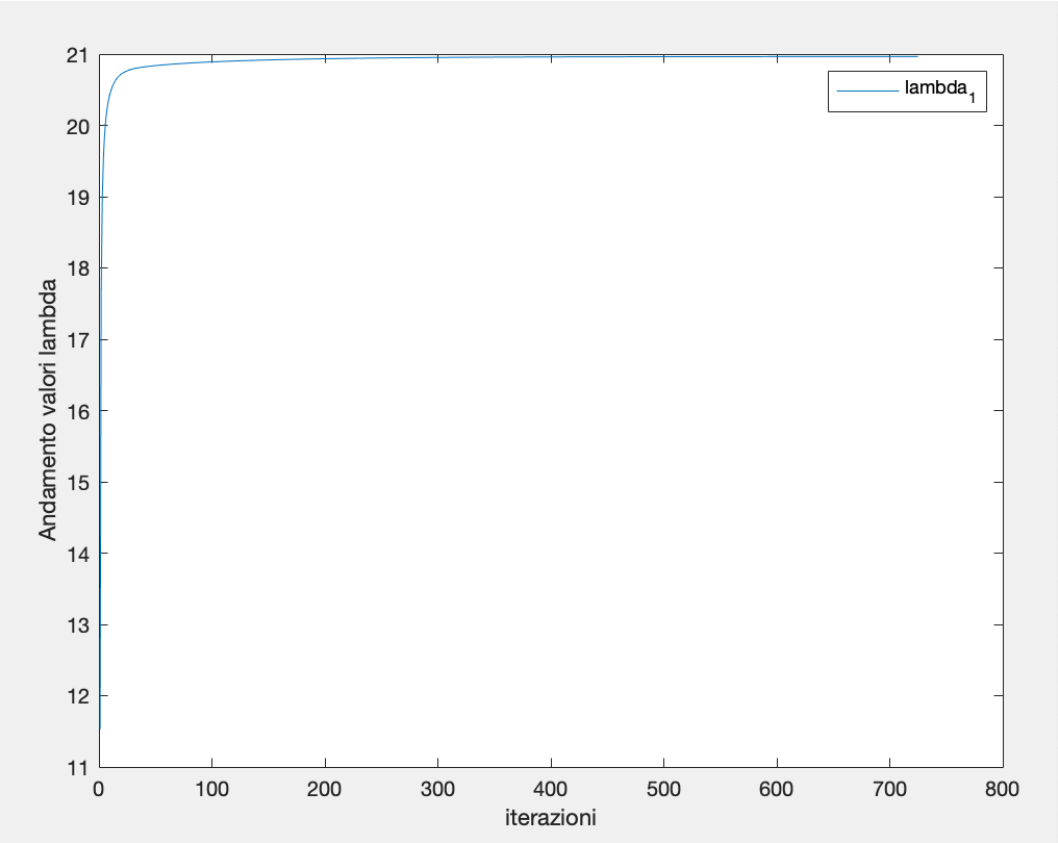



Figura 1.1: Andamento del valore approssimato di $\lambda_1$

| | valore approssimato | valore reale | iterazioni |
|---|---|---|---|
| $\lambda_1$ | 20.965011621 | 20.965219707 | 725 |

Tabella 1.1: Risultati numerici dell'applicazione del metodo delle potenze

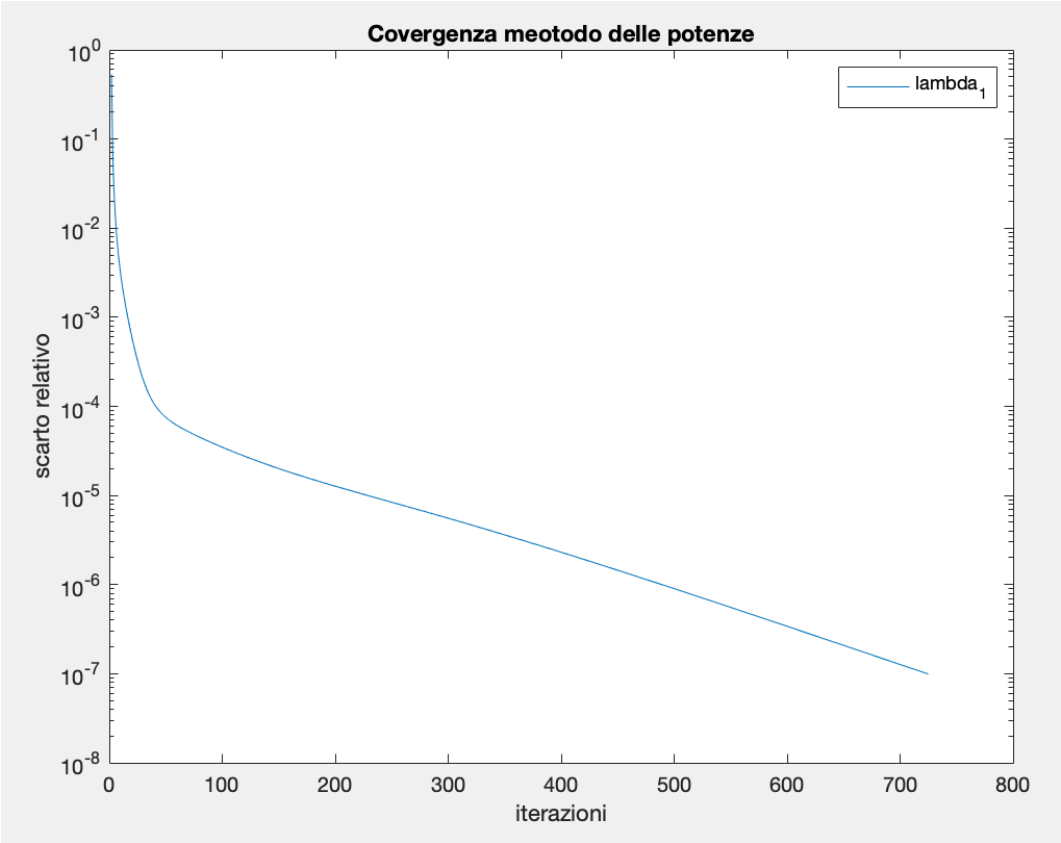



Figura 1.2: Profilo di convergenza del metodo delle potenze

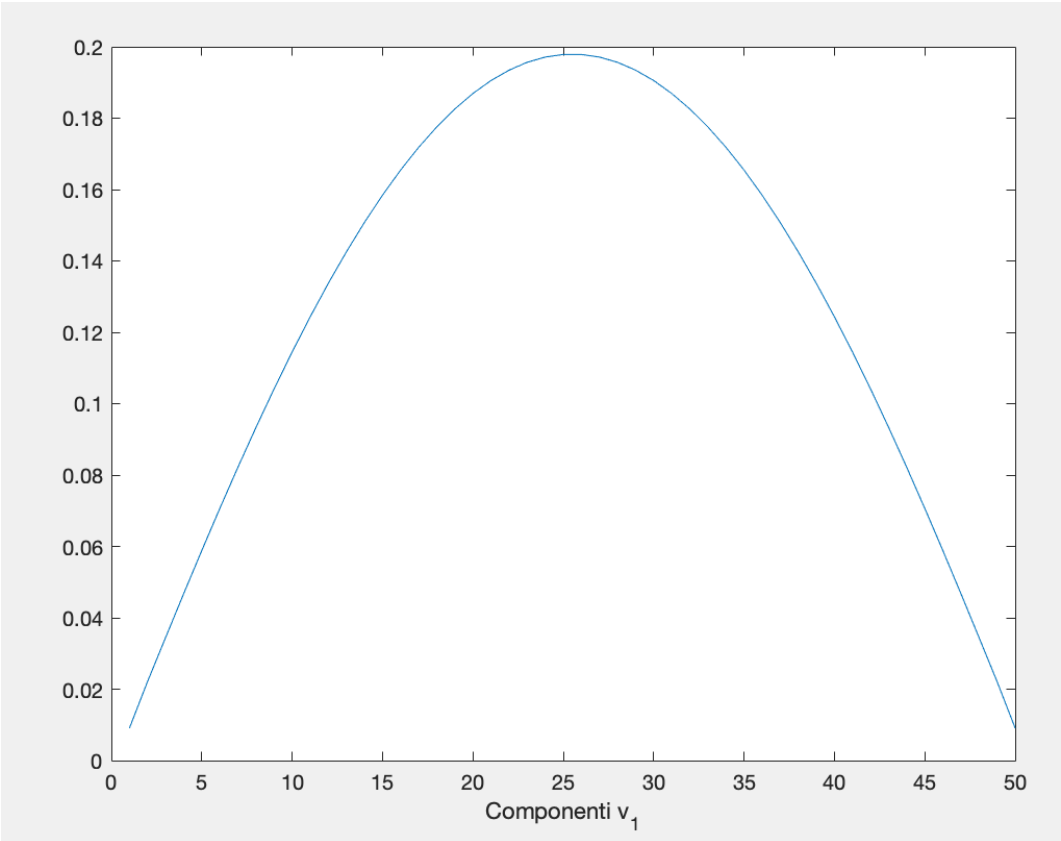



Figura 1.3: Rappresentazione delle componenti di $\vec{v}_1$

In questo caso la convergenza del metodo, visibile in figura 1.2, non è molto veloce, questo accade perché il valore di $\lambda_2$ è 20.861189485 quindi molto vicino in modulo a $\lambda_1$.

Troviamo riscontro di ciò anche nell'equazione (1.8) che rappresenta la velocità di convergenza del metodo. Più il rapporto $(\lambda_2/\lambda_1)$ si avvicina all'unità più il metodo sarà lento a convergere.

A seguire, introducendo la deflazione, vedremo come cercando autovalori diversi da $\lambda_1$, con la proprietà di essere più "isolati" dal successivo, il profilo di convergenza del metodo delle potenze diventi più ripido.

## 1.2 Gradiente coniugato applicato al quoziente di Rayleigh

Questo metodo è applicabile solo nel caso la matrice A sia simmetrica e definita positiva, e permette di calcolare l'autovalore minimo $\lambda_n$. Si definisce il quoziente di Rayleigh come:

$$q(\vec{z}) = \frac{\vec{z}^T A \vec{z}}{\vec{z}^T \vec{z}} \tag{1.10}$$

È verificabile che questo rapporto soddisfa le seguenti proprietà:

1. se $\vec{v}_i$ è un autovettore di A, allora $q(\vec{v}_i)$ è uguale a $\lambda_i$.
2. $\nabla q(\vec{z})$ è pari a zero $\forall$ autovettore di A.
3. $q(\vec{z})$ è limitato e assume valori minimo e massimo, per $\vec{z} = \vec{v}_n$ e $\vec{z} = \vec{v}_1$, rispettivamente.

Riscrivo $q(\vec{z})$ nel sistema di riferimento degli autovettori e ottengo la seguente espressione:

$$q(\vec{z}) = \frac{(c_1\vec{v}_1 + \cdots + c_n\vec{v}_n)^T A (c_1\vec{v}_1 + \cdots + c_n\vec{v}_n)}{(c_1\vec{v}_1 + \cdots + c_n\vec{v}_n)^T (c_1\vec{v}_1 + \cdots + c_n\vec{v}_n)} \tag{1.11}$$

Ricordo che gli autovettori di una matrice simmetrica sono ortogonali e riscrivo la (1.11) come:

$$q(\vec{z}) = \frac{\lambda_1 c_1^2 + \lambda_2 c_2^2 + \cdots + \lambda_n c_n^2}{c_1^2 + c_2^2 \cdots + c_n^2} \tag{1.12}$$

Noto che la (1.12) è sempre positiva ed il valore di $q(\vec{z})$ dovrà essere compreso tra $\lambda_1$ e $\lambda_n$.

Essendo quindi il quoziente di Rayleigh una funzione sempre non negativa posso immaginare di applicare un metodo iterativo come quello del gradiente coniugato per trovarne il minimo che sappiamo corrispondere a $\lambda_n$.

Poniamo $\vec{z}_k$ come approssimazione dell'autovettore $\vec{v}_n$ e calcoliamo la nuova approssimazione come:

$$\vec{z}_{k+1} = \vec{z}_k + \alpha_k \vec{p}_k \tag{1.13}$$

Il fattore $\alpha_k$ presente nella (1.13) deve rendere minimo il quoziente di Rayleigh lungo la direzione $\vec{p}_k$. Per trovare il corrispondente valore risolvo la seguente equazione:

$$\frac{\partial}{\partial \alpha_k}\left[\frac{\vec{z}_{k+1}^T A \vec{z}_{k+1}}{\vec{z}_{k+1}^T \vec{z}_{k+1}}\right] = 0 \tag{1.14}$$

L'equazione (1.14) ha due radici:

$$\alpha_{k1,2} = \frac{(ed - mb) \pm \sqrt{\Delta}}{2(bc - ad)} \tag{1.15}$$

dove:

$$a = \vec{p}_k^T A\vec{z}_k \quad b = \vec{p}_k^T A\vec{p}_k \quad c = \vec{p}_k^T \vec{z}_k \quad d = \vec{p}_k^T \vec{p}_k \quad e = \vec{z}_k^T A\vec{z}_k \quad m = \vec{z}_k^T \vec{z}_k$$

$$\Delta = (ed - mb)^2 - 4(bc - ad)(ma - ec)$$

Le due radici corrispondono al punto di minimo e di massimo di $q(\vec{z}_{k+1})$ lungo la direzione $\vec{p}_k$. Dal momento che l'interesse è la ricerca del minimo, tra le due soluzioni prendo quella con segno positivo che mi assicura la convergenza verso $\lambda_n$. Al contrario calcolando $\alpha$ dalla (1.15) con segno negativo lo schema convergerebbe all'autovalore massimo.

Come nel gradiente coniugato usato per la risoluzione di sistemi lineari, si utilizza come direzione di ricerca quella del gradiente che viene opportunamente ruotata mediante l'uso di un coefficiente $\beta_k$ che rende le direzioni coniugate tra loro rispetto alla conica generata dalla matrice $A$. La direzione di ricerca è quindi aggiornata mediante la seguente formula ricorrente:

$$\vec{p}_{k+1} = \vec{g}_{k+1} + \beta_k \vec{p}_k \tag{1.16}$$

Il gradiente è definito dalla seguente equazione:

$$\begin{aligned} \nabla q(\vec{z}) = \vec{g}_{k+1} &= \frac{2A\vec{z}_{k+1}(\vec{z}_{k+1}^T \vec{z}_{k+1}) - 2\vec{z}_{k+1}(\vec{z}_{k+1}^T A\vec{z}_{k+1})}{(\vec{z}_{k+1}^T \vec{z}_{k+1})^2} \\ &= \frac{2}{\vec{z}_{k+1}^T \vec{z}_{k+1}}[A\vec{z}_{k+1} - q(\vec{z}_{k+1})\vec{z}_{k+1}] = \frac{2\vec{r}_{k+1}}{\vec{z}_{k+1}^T \vec{z}_{k+1}} \end{aligned} \tag{1.17}$$

dove $\vec{r}_{k+1}$ è il residuo associato all'uso di $\vec{z}_{k+1}$ come approssimazione di $\vec{v}_n$.

Noto che esso è pari a zero ogni volta che $\vec{z}_{k+1}$ è uguale ad un qualsiasi autovalore della matrice (seconda proprietà del quoziente di Rayleigh).

Il controllo di convergenza dello schema viene fatto su una norma del residuo $\vec{r}_k$. Nel caso lo schema non sia giunto a convergenza si procede aggiornando la direzione di ricerca $\vec{p}_k$.

Il coefficiente $\beta_k$ viene calcolato facendo in modo che la nuova direzione di ricerca sia A-ortogonale alla precedente. Si può dimostrare, dalla proprietà di ottimo del gradiente coniugato, che questo implica che tutte le direzioni di ricerca siano A-ortogonali. Questa condizione si impone con la (1.18) da cui posso ricavare direttamente $\beta_k$.

$$\vec{p}_{k+1}^T A\vec{p}_k = (\vec{g}_{k+1} + \beta_k \vec{p}_k)^T A\vec{p}_k = 0 \quad \rightarrow \quad \beta_k = -\frac{\vec{g}_{k+1}^T A\vec{p}_k}{\vec{p}_k^T A\vec{p}_k} \tag{1.18}$$

A questo punto è stato ricavato tutto il necessario per la costruzione dell'algoritmo del gradiente coniugato applicato al quoziente di Rayleigh.

L'algoritmo 2 riportato alla fine del paragrafo però non è quello appena ricavato, in quanto il gradiente coniugato deve essere opportunamente modificato per raggiungere più velocemente la convergenza.

Immaginiamo di introdurre una matrice di precondizionamento $X$ simmetrica e non singolare tale che $\vec{y} = X\vec{z}$

$$q(\vec{y}) = \frac{\vec{z}^T X X^{-1} A X^{-1} X \vec{z}}{\vec{z}^T X X^{-1} X^{-1} X \vec{z}} = \frac{\vec{y}^T G \vec{y}}{\vec{y}^T K^{-1} \vec{y}} \tag{1.19}$$

con $G = X^{-1}AX^{-1}$ e $K^{-1} = X^{-1}X^{-1}$.

La nuova matrice $KG$ è simile alla matrice A quindi gli autovalori non cambiano:

$$KG = XXX^{-1}AX^{-1} = XAX^{-1} \tag{1.20}$$

Ripercorrendo tutte le operazioni fatte in precedenza per l'algoritmo non modificato si può dimostrare che le uniche differenze nelle formule ricorrenti usate per calcolare i termini si hanno nel calcolo di $\vec{p}_k$ e $\beta_k$.

$$\vec{p}'_{k+1} = X^{-1}\vec{p}_{k+1} = X^{-1}\vec{g}_{k+1} + \beta_k X \vec{p}_k \tag{1.21}$$

moltiplicando entrambi i membri per $X^{-1}$ ottengo:

$$K^{-1}\vec{p}_{k+1} = K^{-1}\vec{g}_{k+1} + \beta_k \vec{p}_k \tag{1.22}$$

$$\beta_k{}' = \frac{\vec{g}^T_{k+1} X^{-1} X^{-1} A X^{-1} X \vec{p}_k}{\vec{p}^T_k X X^{-1} A X^{-1} X \vec{p}_k} = \frac{(K^{-1}\vec{g}_{k+1})^T A \vec{p}_k}{\vec{p}^T_k A \vec{p}_k} \tag{1.23}$$

L'algoritmo modificato con il precondizionatore differisce soltanto nel calcolo del vettore $\vec{v} = K^{-1}\vec{g}$ ad ogni iterazione. L'applicazione di $K^{-1}$ può essere fatta a parte come mostrato nell'algoritmo 2.

L'algoritmo così modificato ha una convergenza molto più rapida a patto che la matrice $X$, e quindi $K^{-1}$, sia scelta opportunamente.

Si può dimostrare che la convergenza del metodo dipende dal numero di condizionamento spettrale dell'Hessiano di $q(\vec{z})$ calcolato in $\vec{v}_n$, che risulta pari alla seguente espressione:

$$\kappa[H(\vec{v}_n)] = \frac{\lambda_1 - \lambda_n}{\lambda_{n-1} - \lambda_n} \tag{1.24}$$

Generalmente il valore di $\kappa$ non è piccolo, infatti la differenza tra $\lambda_1$ e $\lambda_n$ è molto maggiore rispetto a quella tra $\lambda_{n-1}$ e $\lambda_n$. Mi aspetto quindi di raggiungere la convergenza lentamente.

Nel caso dell'algoritmo modificato la (1.24), scegliendo $K^{-1} = A^{-1}$, diventa:

$$\kappa[H(\vec{y}_n)] = \frac{\lambda_1 - \lambda_n}{\lambda_{n-1} - \lambda_n}\frac{\lambda_{n-1}}{\lambda_1} = \kappa[H(\vec{v}_n)]\frac{\lambda_{n-1}}{\lambda_1} \tag{1.25}$$

Dal momento che $\lambda_{n-1}/\lambda_1 < 1$ è immediato concludere che il numero di condizionamento spettrale dell'Hessiano è molto più piccolo rispetto al caso non precondizionato e quindi lo schema avrà convergenza molto più rapida.

Come apprendo dalle (1.24), (1.25) utilizzando $A^{-1}$ come matrice di precondizionamento si ottiene una buona accelerazione dello schema, tuttavia l'applicazione dell'inversa di una matrice è molto costosa. Al suo posto, si opta per l'applicazione di un'approssimazione dell'inversa di A, esplicita o implicita.

La scelta del precondizionatore è fondamentale, la sua applicazione tramite il calcolo di $\vec{v}$ deve avere un costo computazionale il più basso possibile e l'accelerazione ricavata deve giustificare la maggior complessità del nuovo algoritmo modificato.

Nelle applicazioni a seguire verranno utilizzati due diversi precondizionatori.

La matrice di precondizionamento di Jacobi deriva dalla seguente scomposizione di $A$:

$$A = F - M - N \tag{1.26}$$

La matrice $F$ è una matrice diagonale che contiene gli elementi sulla diagonale di A, $-M$ è una matrice triangolare bassa ed $-N$ è triangolare alta contenenti i restanti elementi di $A$. Ipotizzando che la matrice $A$ sia diagonalmente dominante gli elementi delle matrici $-M$ e $-N$ risultano trascurabili rispetto ai coefficienti della diagonale contenuti in $F$. L'inversa di $F$ sarà quindi un'approssimazione di $A^{-1}$.

$$K^{-1} = F^{-1} \tag{1.27}$$

Chiaramente questo precondizionatore produce migliori risultati tanto più la matrice $A$ è diagonalmente dominante. L'altro precondizionatore utilizzato è la decomposta incompleta di Cholesky dove si pone:

$$K^{-1} = (\tilde{L}\tilde{L}^T)^{-1} \tag{1.28}$$

$\tilde{L}$ è il fattore triangolare incompleto di Cholesky, esso infatti viene calcolato solamente in alcune posizioni predefinite.

Operativamente non viene calcolata esplicitamente l'inversa di $\tilde{L}\tilde{L}^T$ ma vengono risolti in successione due sistemi triangolari.

Ponendo $\vec{v} = (\tilde{L}\tilde{L}^T)^{-1}\vec{r}_{k+1}$ e $\vec{w} = \tilde{L}^T\vec{v}$

$$\tilde{L}\vec{w} = \vec{r}_{k+1} \tag{1.29}$$

$$\tilde{L}^T\vec{v} = \vec{w} \tag{1.30}$$

Risolvendo il secondo sistema (1.30) ottengo il vettore $\vec{v}$ che posso applicare allo schema.

Il costo computazionale di questo precondizionatore è maggiore e dipende da quanti elementi di $\tilde{L}$ calcolo. Nella versione più semplice di questo precondizionatore si calcolano i termini solamente nelle posizioni dove $A$ contiene elementi non nulli, ma questo non significa che non si possano adottare altri criteri.

**Algorithm 2** GCM applicato al quoziente di Rayleigh

**procedure** GCM RAYLEIGH(A,K,toll,iprec,n)
  ▷ Preparo valori di partenza
  $\vec{z} = zeros(n,1);\ \vec{z}(1) = 1;$
  $\vec{s} = A\vec{z}; \vec{e} = \vec{z}'\vec{s};\ \vec{m} = \vec{z}'\vec{z}; q = \vec{e}/\vec{m};$
  $\vec{r} = \vec{s} - q\vec{z};\ r_0 = norm(\vec{r});$
  $\vec{g} = (2\vec{r})/\vec{m};$
  $\vec{p} = appl\,prc(K, iprec, \vec{g});$ ▷ Applico precondizionatore
  $itmax = 300;\ iter = 0;\ rr = 1;$

  **while** $iter \leq itmax,\ rr \geq toll$ **do**
    $iter = iter + 1;$
    $\vec{t} = A\vec{p};$ ▷ Preparo prodotto matrice-vettore
    $\vec{a} = \vec{p}'\vec{s}; \vec{b} = \vec{p}'\vec{t}; \vec{c} = \vec{p}'\vec{z}; \vec{d} = \vec{p}'\vec{p};$
    $\Delta = (\vec{e}\vec{d} - \vec{m}\vec{b})^2 - 4(\vec{b}\vec{c} - \vec{a}\vec{d})(\vec{m}\vec{a} - \vec{e}\vec{c});$
    $\alpha = 0.5(\vec{e}\vec{d} - \vec{m}\vec{b} + \sqrt{\Delta})/(\vec{b}\vec{c} - \vec{a}\vec{d})$ ▷ Calcolo il coefficiente $\alpha$

    $\vec{z} = \vec{z} + \alpha\vec{p};$ ▷ Trovo la nuova approssimazione dell'autovettore $\vec{v}_n$

    $\vec{s} = s + \alpha\vec{t};\ \vec{e} = \vec{z}'\vec{s}; \vec{m} = \vec{z}'\vec{z};$
    $q = \vec{e}/\vec{m};$ ▷ Noto che $q = \lambda_n$ se $\vec{z} = \vec{v}_n$
    $\vec{r} = \vec{s} - q\vec{z};$ ▷ Calcolo il nuovo residuo

    $\vec{g} = (2\vec{r})/\vec{m}; \vec{v} = appl\,prc(K, iprec, \vec{g});$ ▷ Applico precondizionatore
    $\beta = -(\vec{v}'\vec{t})/\vec{b};$
    $\vec{p} = \vec{v} + \beta\vec{p};$ ▷ Trovo la nuova direzione di ricerca
    $rr = norm(\vec{r})/r_0$ ▷ Controllo sulla convergenza
  **end while**
**end procedure**

Applichiamo l'algoritmo 2 alla matrice $C$ introdotta nell'equazione (1.9).

In figura 1.4 vengono riportati i profili di convergenza ottenuti con il metodo del gradiente coniugato applicato a $q(\vec{z})$ variando il precondizionatore.

Vediamo come l'effetto dei precondizionatori sia evidente anche per matrici relativamente piccole. I risultati migliori in termini di velocità di convergenza si ottengono con la decomposta incompleta di Cholesky. Il vantaggio che deriva dal suo utilizzo giustifica il maggior costo computazionale per ciascuna iterazione.

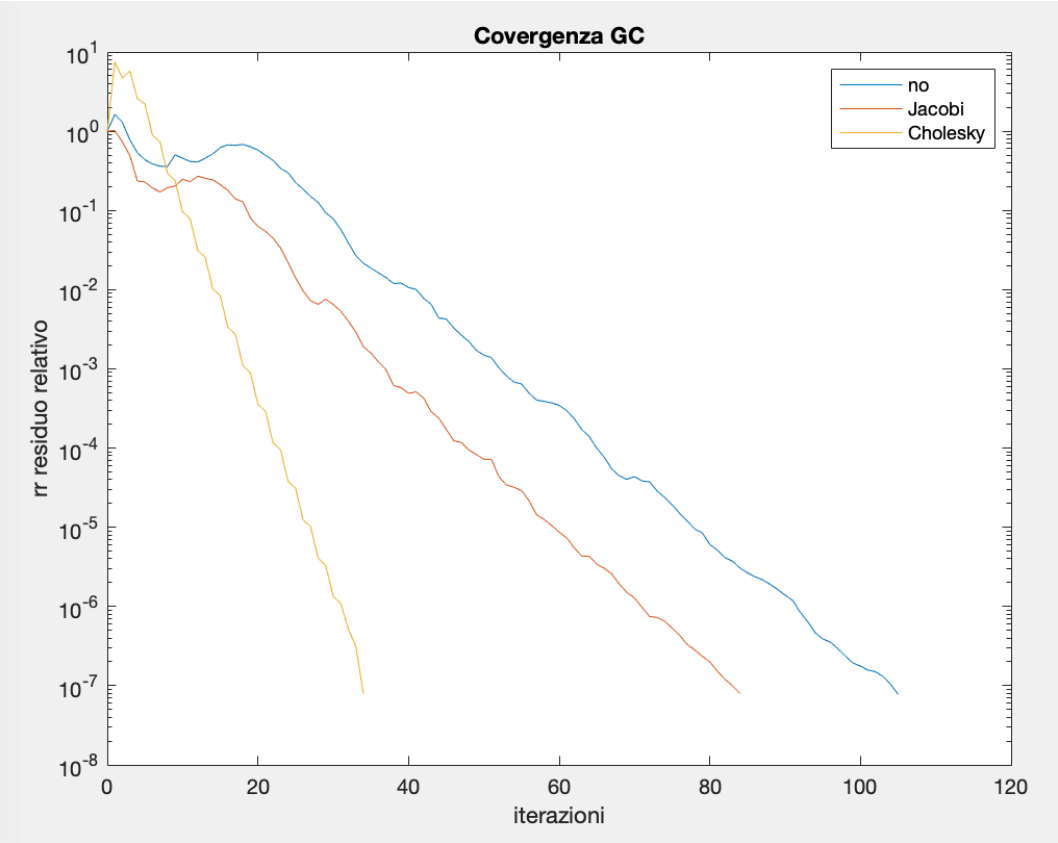



Figura 1.4: Andamento del valore di $\lambda_n$ al variare del precondizionatore. Matrice $C \quad n = 50$

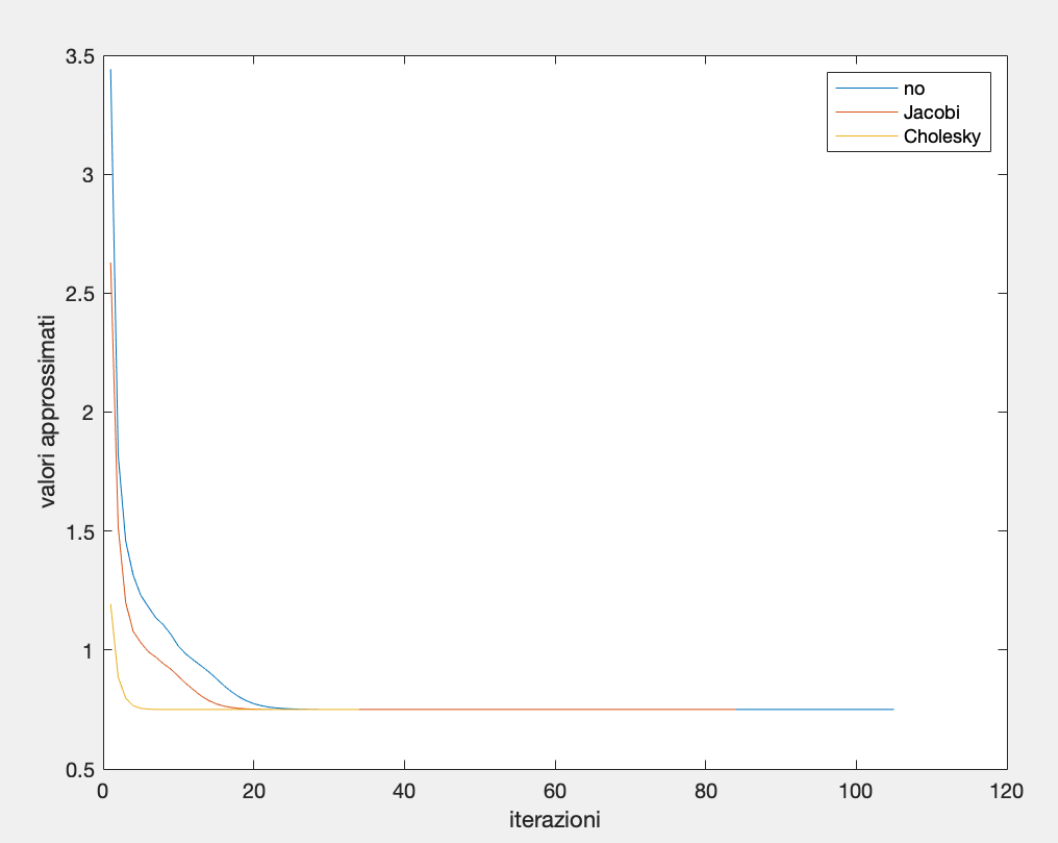



Figura 1.5: Profili di convergenza del calcolo dell'autovalore minimo di $C \quad n = 50$ al variare del precondizionatore

La figura 1.5 mostra come variano i valori approssimati di $\lambda_n$ durante l'applicazione dello schema. Anche qui è possibile vedere come gli schemi modificati raggiungano più velocemente un intorno del valore finale di $\lambda_n$.

In seguito sono riportati in tabella 1.2 i risultati numerici ottenuti:

| Valore approssimato | Iterazioni | Precondizionatore |
|---|---|---|
| 0.750000000 | 105 | $K^{-1} = I$ |
| 0.750000000 | 84 | $K^{-1} = F^{-1}$ |
| 0.750000000 | 34 | $K^{-1} = (\tilde{L}\tilde{L}^T)^{-1}$ |

Tabella 1.2: Risultati del calcolo di $\lambda_n$ al variare del precondizionatore

Il valore reale di $\lambda_n$ è 0.750000000. In tutti e tre in casi il valore approssimato è molto vicino a quello reale. Fissata la tolleranza del residuo relativo a $10^{-8}$, la differenza sta nel numero di iterazioni compiute fino alla convergenza

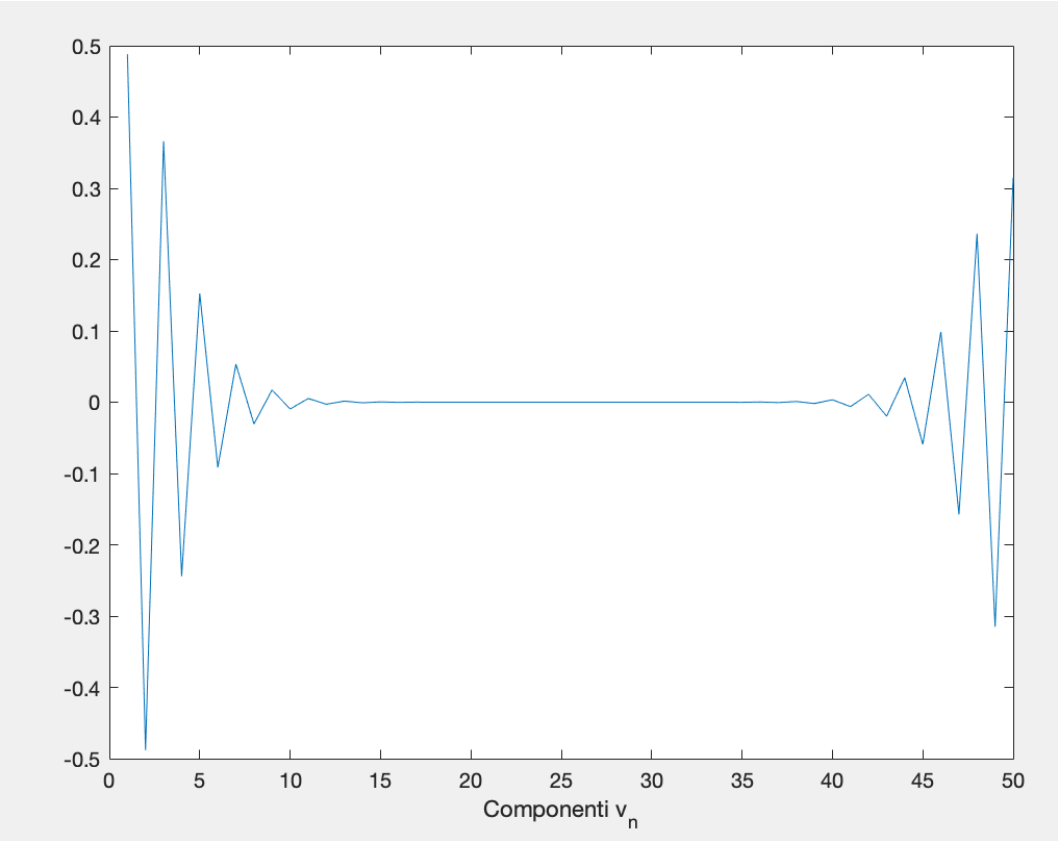



Figura 1.6: Rappresentazione delle componenti dell'autovettore $\vec{v}_n$

## 1.3 Deflazione

La tecnica della deflazione viene utilizzata quando, per specifiche applicazioni, non sono solamente interessato agli autovalori estremi $\lambda_1$ e $\lambda_n$ ma, ad esempio, i $j$ autovalori più grandi o più piccoli.

Come visto in precedenza il metodo delle potenze applicato ad una matrice A permette di trovare l'autovalore $\lambda_1$ e rispettivo autovettore.

Immaginando di applicare una trasformazione alla matrice $A$ che renda "invisibile" al metodo delle potenze $\lambda_1$ e $\vec{v}_1$ si può isolare il secondo autovalore

più grande e rispettivo autovettore. Costruiamo la matrice $A_1$ come:

$$A_1 = A - \lambda_1 \vec{v}_1 \vec{v}_1^T \tag{1.31}$$

$\vec{v}_1$ viene precedentemente scalato in modo da avere norma-2 unitaria.

Vediamo che una matrice $A_1$ così costruita mantiene gli stessi autovalori della matrice $A$ escluso $\lambda_1$ che diventa uguale a zero.

$$A_1 \vec{v}_1 = A\vec{v}_1 - \lambda_1 \vec{v}_1 \vec{v}_1^T \vec{v}_1 = \lambda_1 \vec{v}_1 - \lambda_1 \vec{v}_1 = 0 \tag{1.32}$$

$$A_1 \vec{v}_k = A\vec{v}_k - \lambda_1 \vec{v}_1 \vec{v}_1^T \vec{v}_k = \lambda_k \vec{v}_k \quad k = 2, \cdots, n \tag{1.33}$$

L'applicazione della deflazione al metodo delle potenze consiste nel modificare la matrice di partenza in modo da azzerare man mano gli autovalori più grandi che trovo di volta in volta con il metodo iterativo. Per farlo come vedo dalla (1.31) necessito di $\lambda_i$ e rispettivo autovettore normalizzato in modo da avere norma-2 unitaria.

Per la ricerca del j-esimo autovalore più grande della matrice $A$ la matrice $A_j$ sarà così determinata:

$$A_j = A - \sum_{i=1}^{j} \lambda_i \vec{v}_i \vec{v}_i^T \tag{1.34}$$

Deflazionando il metodo j volte posso trovare i $j$ autovalori più grandi.

Per quanto riguarda l'implementazione su Matlab, l'unica modifica apportata all'algoritmo 1 si trova nella riga 8 dove viene inserita l'equazione (1.34) con l'accortezza di non calcolare la matrice $A_j$, che altrimenti perderebbe la sua sparsità, ma moltiplicando direttamente per $\vec{z}_k$.

$$A_j \vec{z}_k = A\vec{z}_k - \lambda_1 \vec{v}_1^T \vec{z}_k \vec{v}_1 - \cdots - \lambda_1 \vec{v}_j^T \vec{z}_k \vec{v}_j \tag{1.35}$$

Si riportano i risultati ottenuti applicando la tecnica appena descritta alla matrice $C$ (1.9).

La figura 1.7 ci mostra, per ogni autovalore da $\lambda_1$ a $\lambda_8$, l'andamento dei valori approssimati.

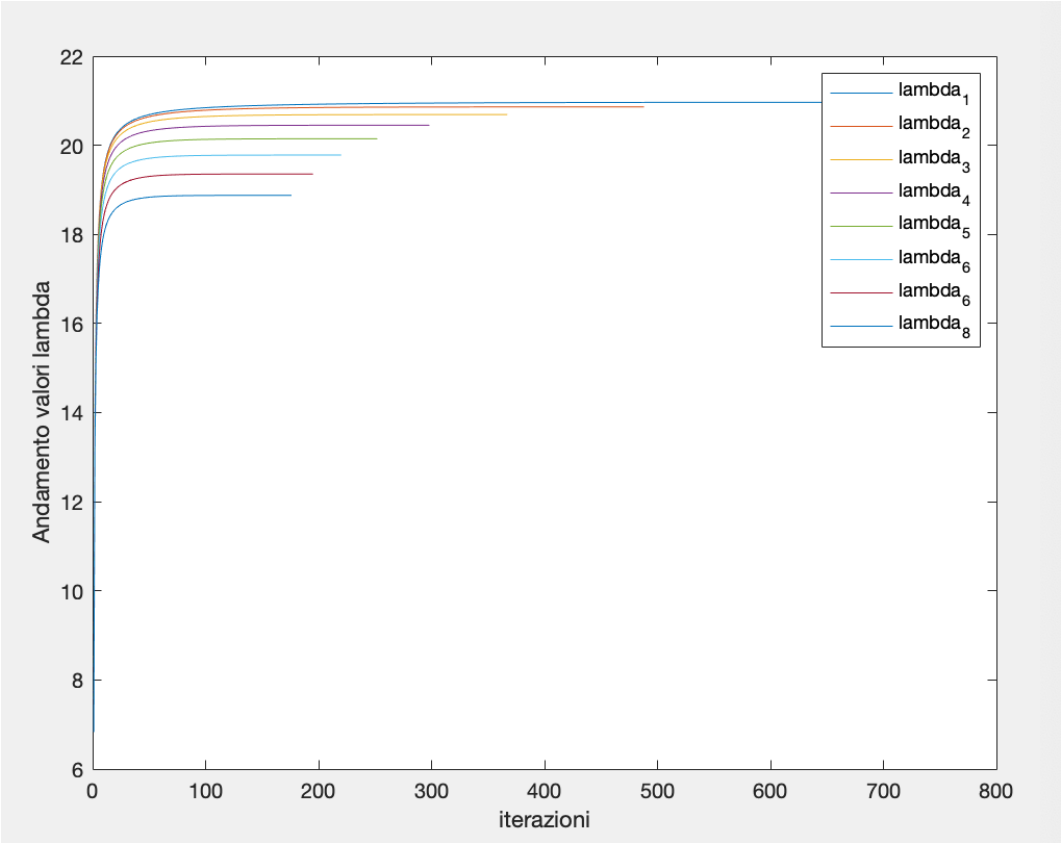



Figura 1.7: Andamento del modulo degli autovalori all'estremità superiore dello spettro di $C$

Il numero di iterazioni necessarie per raggiungere la convergenza diminuisce ad ogni nuovo autovalore trovato (figura 1.8).

La spiegazione di questo fenomeno si ha ricordando che il metodo delle potenze è meno efficace quando sono presenti autovalori consecutivi molto vicini in modulo. Questo comporta che il rapporto con peso maggiore sull'errore $(\lambda_{i+1}/\lambda_i)^k$ dell'equazione (1.3) tenda a zero molto più lentamente. In questi casi il metodo convergerà rapidamente ad un valore prossimo a $\lambda_i$ e $\lambda_{i+1}$, la successiva distinzione dei due autovalori avverrà più lentamente.

Noto, dalla tabella 1.3, come il distanziamento tra $\lambda_i$ e $\lambda_{i+1}$ aumenti progressivamente, di conseguenza aumenta la velocità di convergenza. Questa osservazione è verificata per $\lambda_1, \cdots, \lambda_8$, non è possibile generalizzare.

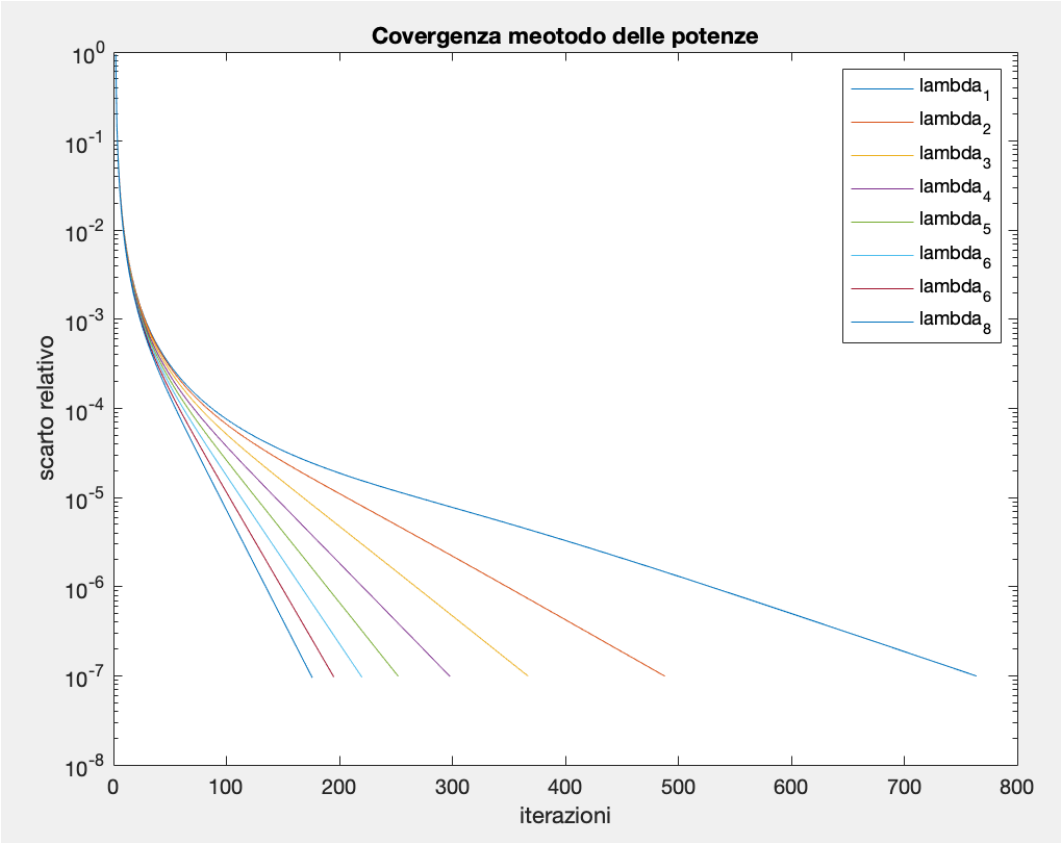



Figura 1.8: Profili di convergenza del metodo delle potenze deflazionato per i primi otto autovalori di $C$

A seguire, in tabella 1.3, si riportano i risultati numerici ottenuti deflazionando il metodo delle potenze.

| | valore approssimato | valore reale | iterazioni |
|---|---|---|---|
| $\lambda_1$ | 20.965011174 | 20.965219707 | 764 |
| $\lambda_2$ | 20.861273977 | 20.861189485 | 483 |
| $\lambda_3$ | 20.688873221 | 20.688837421 | 363 |
| $\lambda_4$ | 20.449718181 | 20.449697430 | 298 |
| $\lambda_5$ | 20.145902209 | 20.145890108 | 251 |
| $\lambda_6$ | 19.780106337 | 19.780096315 | 220 |
| $\lambda_7$ | 19.355529660 | 19.355523856 | 195 |
| $\lambda_8$ | 18.875873296 | 18.875867748 | 176 |

Tabella 1.3: Confronto tra i valori reali ed approssimati per gli autovalori più grandi di $C$

Spostando l'attenzione all'altra estremità dello spettro e quindi alla ricerca degli autovalori più piccoli si può pensare di deflazionare il gradiente coniugato applicato al quoziente di Rayleigh facendo una semplice osservazione.

Nel caso il vettore iniziale $\vec{z}_0$ abbia componente nulla ($\vec{v}_n^T \vec{z}_0 = 0$) lungo l'autovettore $\vec{v}_n$ lo schema non converge a $\lambda_n$ ma a $\lambda_{n-1}$.

Infatti, non essendoci una componente lungo $\vec{v}_n$ sul vettore iniziale neppure le successive direzioni di ricerca $\vec{p}_k$ e vettori $\vec{z}_k$ avranno componente in $\vec{v}_n$. Se riscriviamo $\vec{z}_k$ come combinazione lineare dello spazio degli autovettori di $A$ il coefficiente moltiplicativo di $\vec{v}_n$ sarà sempre zero.

Se l'interesse fosse rivolto a $\lambda_{n-2}$ sarà necessario scegliere un vettore iniziale ortogonale contemporaneamente a $\vec{v}_n$ e $\vec{v}_{n-1}$ e così via.

Sulla base di questa osservazione posso applicare la deflazione per ricavare i $j$ autovalori più piccoli di una matrice simmetrica definita positiva. Non mi basta far altro che trovare gli autovettori $\vec{v}_n, \vec{v}_{n-1}, \cdots, \vec{v}_{j+1}$ e relativi autovalori applicando l'algoritmo 2 più volte. Successivamente posso riapplicare lo schema scegliendo come vettore iniziale $\vec{v}_0$, avente componente nulla lungo tutte le direzioni dei precedenti autovettori, in questo modo lo schema convergerà a $\lambda_j$ e $\vec{v}_j$.

**Algorithm 3** Ortogonalizzazione di Gram-schmidt

1: **procedure** GRAM-SCHMIDT($\vec{z}_k$,V,i)
2: ▷ Questa funzione ortogonalizza $\vec{z}$ in entrata agli altri vettori contenuti in $V$
3: ▷ Le colonne della matrice $V$ contengono gli autovettori, $i$ è la dimensione
4: **for** $j = 1 \rightarrow i - 1$ **do**
5: $w = V_{:,j}$; $\vec{x} = \vec{z}$;
6: $\vec{z} = \vec{z} - (\vec{x}^T\vec{w})/(\vec{w}^T\vec{w})\vec{w}$
7: **end for**
8: $\vec{z} = \vec{z}/norm(\vec{z})$
9: **end procedure**

L'algoritmo 3 appena riportato ortonormalizza ad un sottospazio di autovettori di $A$ il vettore iniziale $\vec{z}_0$. In linea teorica basterebbe richiamare questa funzione una sola volta in modo da trovare il vettore da cui partire ed applicare la deflazione.

Per via dell'accumularsi degli errori di arrotondamento dopo alcuni passi si svilupperanno delle componenti non nulle associate a $\vec{v}_n$. Ciò significa che pur scegliendo un vettore iniziale "appropriato" lo schema seppur più lentamente convergerà comunque a $\lambda_n$.

Dalle figure 1.9 e 1.10 possiamo vedere come la scelta del vettore iniziale possa influire sulle proprietà di convergenza dello schema, tuttavia ad un certo punto si innesca la convergenza asintotica all'autovalore minimo della matrice.

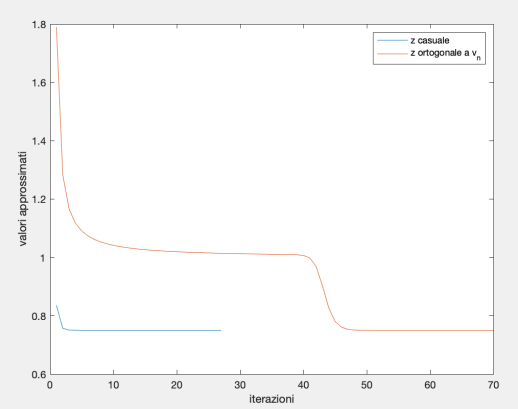



Figura 1.9: Andamento dei valori di $\lambda$ calcolati durante la convergenza con $\vec{z}$ casuale e $\vec{z}$ ortogonale a $\vec{v}_n$

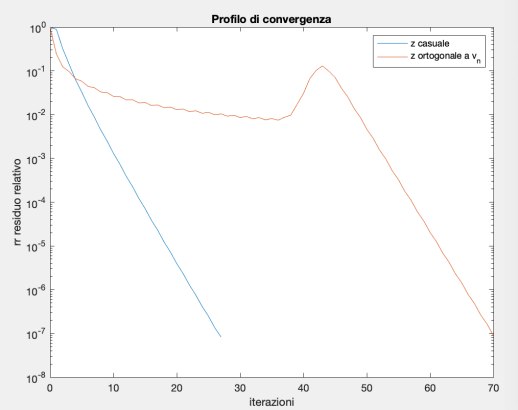



Figura 1.10: Profili di convergenza del GCM applicato a $q(\vec{z})$ ottenuti con due diversi vettori iniziali

Scegliendo un vettore iniziale $\vec{z}_0$ ortogonale a $\vec{v}_n$ il valore trovato di $\lambda$ inizialmente tende tende a $\lambda_{n-2}$, ma dopo poche iterazioni si stabilizza su $\lambda_n$. A

fronte dello stesso risultato abbiamo però una convergenza più lenta.

In questo caso, $\vec{z}_0$ avrà componente lungo $\vec{v}_{n-2}$ maggiore rispetto a quella su $\vec{v}_{n-1}$.

Risulta necessario, al fine di mantenere l'ortogonalità tra i vettori, richiamare l'algoritmo 3 all'interno dell'algoritmo 2 dopo aver trovato la nuova approssimazione di $\vec{z}_k$ alla riga 17.

Non è necessario richiamare Gram-Schmidt ad ogni iterazione. Dopo qualche prova si può concludere che la frequenza di richiamo dipende molto dalle caratteristiche della matrice e dalla sua dimensione. Nel caso di $C$ è necessario ri-ortogonalizzare ogni 2/3 iterazioni mentre per matrici più grandi è sufficiente farlo ogni 5/10.

In seguito si riportano i risultati ottenuti deflazionando il gradiente coniugato applicato a $q(\vec{z})$.

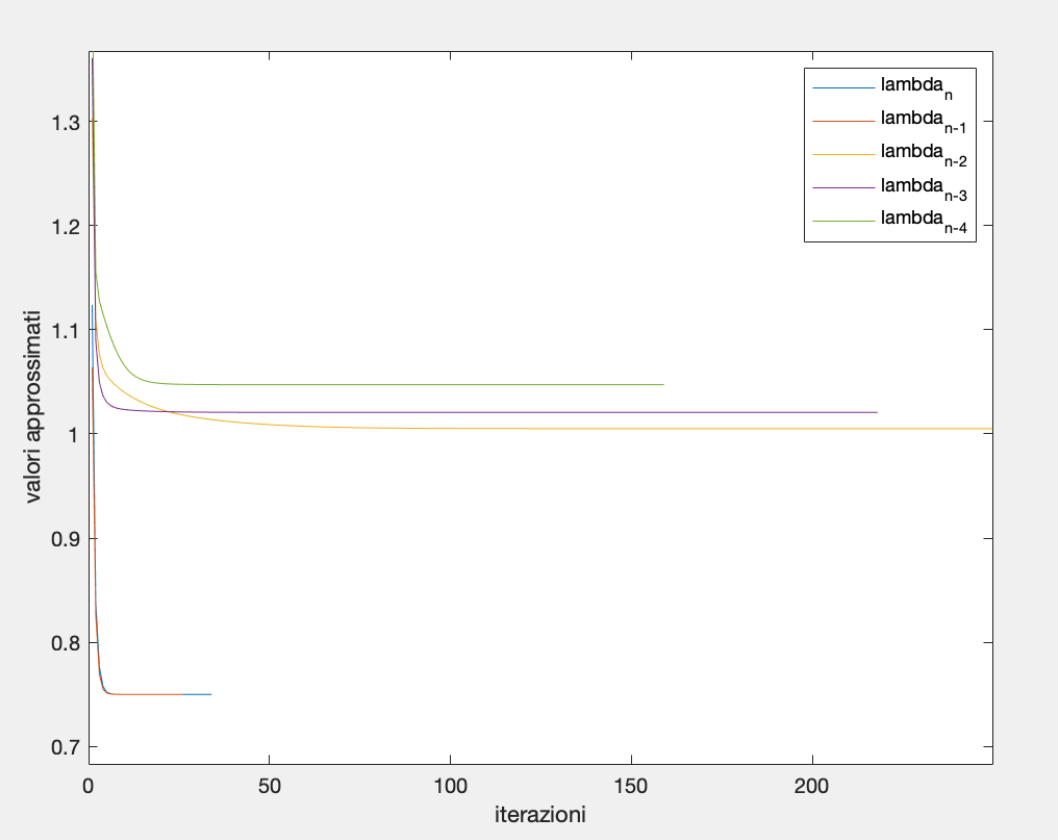



Figura 1.11: Andamento del modulo degli autovalori all'estremità inferiore dello spettro di $C$

Da notare come richiamando Gram-Schmidt (algoritmo 3) più volte lo schema riesce a distinguere $\lambda_n$ da $\lambda_{n-1}$. In tabella 1.4 si riportano i risultati numerici ottenuti.

| | valore approssimato | valore reale | iterazioni |
|---|---|---|---|
| $\lambda_n$ | 0.749999999 | 0.749999999 | 34 |
| $\lambda_{n-1}$ | 0.750000000 | 0.750000000 | 26 |
| $\lambda_{n-2}$ | 1.005121067 | 1.005121067 | 500 |
| $\lambda_{n-3}$ | 1.020691316 | 1.020691316 | 218 |
| $\lambda_{n-4}$ | 1.047323300 | 1.047323300 | 159 |

Tabella 1.4: Confronto tra i valori reali ed approssimati per gli autovalori più piccoli di $C$

# Capitolo 2

# Calcolo dell'intero spettro della matrice

## 2.1 Metodo QR

L'algoritmo $QR$ è un metodo iterativo che permette di determinare l'intero spettro di una matrice.

Il metodo consiste in una successione di operazioni di similitudine applicate alla matrice $A$ tali da creare una matrice triangolare superiore avente sulla diagonale gli autovalori di $A$.

**Teorema 1** *Ogni matrice quadrata A di ordine n può essere decomposta nei seguenti fattori:*

$$A = \begin{bmatrix} \vec{q}_1 & \vec{q}_2 & \dots & \vec{q}_n \end{bmatrix} \begin{bmatrix} r_{1,1} & r_{1,2} & \dots & r_{1,n} \\ & r_{2,2} & \dots & r_{2,n} \\ & & \ddots & \vdots \\ & & & r_{n,n} \end{bmatrix} = QR \tag{2.1}$$

- *Una matrice ortogonale Q ($Q^TQ = I$)*
- *Una matrice triangolare superiore R*

La fattorizzazione QR è alla base del metodo e verrà ora descritta.

L'idea si basa sostanzialmente sull'ortogonalizzazione di Gram-Schmidt già introdotta nell'algoritmo 3.

Moltiplicando Q per la prima colonna di R si dovrà ottenere necessariamente la prima colonna di A, ovvero $\vec{a}_1$.

$$r_{1,1}\vec{q}_1 = \vec{a}_1 \tag{2.2}$$

Sapendo però che $\| \vec{q}_1 \|_2 = 1$, $r_{1,1} = \| \vec{a}_1 \|_2$ e quindi $\vec{q}_1 = \vec{a}_1 / \| \vec{a}_1 \|_2$. Per la seconda colonna, invece risulta:

$$r_{1,2}\vec{q}_1 + r_{2,2}\vec{q}_2 = \vec{a}_2 \tag{2.3}$$

Premoltiplicando per $\vec{q}_1^T$ e ricordando che $\vec{q}_i^T\vec{q}_i = 1$ e che $\vec{q}_i^T\vec{q}_j = 0$ ottengo:

$$r_{1,2} = \vec{q}_1^T\vec{a}_2 \tag{2.4}$$

Dalla (2.3), ricordando che $\| \vec{q_2} \|_2$ deve essere pari a 1, ricavo:

$$r_{2,2} = \| \vec{a}_2 - r_{1,2}\vec{q_1} \|_2 \tag{2.5}$$

E sempre dalla (2.3), conoscendo ora $r_{1,2}$ e $r_{2,2}$, posso calcolare $\vec{q_2}$

$$\vec{q_2} = \frac{1}{r_{2,2}}(\vec{a}_2 - r_{1,2}\vec{q_1}) \tag{2.6}$$

A questo punto risultano definite le prime 2 colonne delle matrici Q ed R, non mi resta che generalizzare la procedura per il calcolo di tutte le altre colonne. Moltiplicando una generica colonna $k$ di R per Q ottengo $\vec{a}_k$ come:

$$\sum_{i=1}^{k} r_{i,k}\vec{q_i} = \vec{a}_k \tag{2.7}$$

Come fatto in precedenza se premoltiplico la precedente espressione per $\vec{q_j}^T$ con $j = 1, \ldots, k-1$, sfruttando l'ortogonalità della matrice Q, posso ricavare uno ad uno i coefficienti della colonna $k$ della matrice R a parte $r_{k,k}$.

$$r_{j,k} = \vec{q_j}^T \vec{a}_k \quad j = 1, \ldots, k-1 \tag{2.8}$$

A questo punto non resta che riscrivere la (2.7) isolando il termine relativo a $\vec{q_k}$.

$$r_{k,k}\vec{q_k} = \vec{a}_k - \sum_{i=1}^{k-1} r_{i,k}\vec{q_i} \tag{2.9}$$

Ricordando che $\| \vec{q_k} \|_2 = 1$:

$$r_{k,k} = \| \vec{a}_k - \sum_{i=1}^{k-1} r_{i,k}\vec{q_i} \|_2 \qquad \vec{q_k} = \frac{1}{r_{k,k}}\left(\vec{a}_k - \sum_{i=1}^{k-1} r_{i,k}\vec{q_i}\right) \tag{2.10}$$

Una volta in possesso delle (2.8), (2.10) sono in grado di calcolare tutti i termini di cui necessito per la costruzione delle matrici Q ed R, e quindi sono in grado di operare la fattorizzazione.

L'algoritmo 4 riportato in sequenza rappresenta l'implementazione del processo appena descritto al calcolatore.

**Algorithm 4** Fattorizzazione QR

```
1:  procedure QRM FACT(A)
2:      [n, m] = size(A); Q = zeros(n, m); R = zeros(n, m);   ▷ Creo matrici Q e R
3:      R_{1,1} = norm(A_{:,1}); Q_{:,1} = A_{:,1}/R_{1,1};   ▷ Assegno valori di partenza
4:
5:      for k = 2 → m do
6:          Q_{:,k} = A_{:,k};
7:          for j = 1 → k − 1 do
8:              R_{j,k} = Q'_{:,j} Q_{:,k};
9:              Q_{:,k} = Q_{:,k} − R_{j,k} Q_{:,j};
10:         end for
11:         R_{k,k} = norm(Q_{:,k});
12:         Q_{:,k} = Q_{:,k}/R_{k,k};
13:     end for
14:
15: end procedure
```

Come detto in precedenza il metodo QR consiste in una serie di operazioni di similitudine. Prendiamo la matrice $A$ e costruiamo una successione di matrici con la seguente formula:

$$A_{k+1} = Q_k^T A_k Q_k = Q_k^T Q_k R_k Q_k = R_k Q_k \tag{2.11}$$

Noto che moltiplicando sia a destra che a sinistra $A_k$ per la matrice $Q_k$, ottenuta dalla fattorizzazione QR, non si va a modificare lo spettro di $A_{k+1}$ in quanto l'operazione risulta essere una similitudine tra matrici.

Al limite la successione (2.11) diventa:

$$\lim_{k\to\infty} A_k = U \quad \lim_{k\to\infty} Q_k = I \quad \lim_{k\to\infty} R_k = U \tag{2.12}$$

Avrò quindi ottenuto una matrice $R$ triangolare superiore che avrà sulla propria diagonale gli autovalori della matrice $A$ di partenza.

**Algorithm 5** Metodo QR

```
1:  procedure QR METHOD(A)
2:      err = 1; toll = 10^-8;                    ▷ Assegno valori di partenza
3:      itmax = 100; iter = 0;
4:      [Q, R] = QRfact(A);
5:
6:      while iter ≤ itmax, err ≥ toll do
7:          iter = iter + 1;
8:          A = RQ;                    ▷ Calcolo nuova matrice A e la fattorizzo
9:          [Q, R1] = QRfact(A);
10:         err = norm(diag(R1) − diag(R));          ▷ Controllo sulla convergenza
11:         R = R1;
12:     end while
13:
14: end procedure
```

Riguardo l'algoritmo 5 appena riportato si possono fare alcune osservazioni. Innanzitutto, come si può vedere alla riga 8, lo schema contiene un prodotto matrice-matrice. Questa operazione ha un costo computazionale molto elevato, nell'ordine delle $n^2$ operazioni, per matrici dense. La soluzione è modificare la matrice A prima di applicare il metodo QR come descritto nel paragrafo 2.2.

Per quanto riguarda il controllo di convergenza sullo schema è possibile operare calcolando e poi confrontando con una tolleranza fissata la norma dello scarto tra la diagonale di $R_{k+1}$ ed $R_k$.

Un'altra possibilità è quella di applicare la norma-$\infty$, infatti da alcune osservazioni, derivanti da prove svolte su Matlab, l'autovalore $\lambda_1$ (ovvero il più grande) è il più lento a convergere. Una volta verificata la convergenza di $\lambda_1$ anche gli altri autovalori si saranno stabilizzati su un valore preciso.

Si applica il metodo QR alla matrice $C$ (1.9).

In figura 2.1 vediamo raffigurato lo spettro di $C$ e i valori assunti dalla diagonale di $R$ durante la convergenza.

La convergenza del singolo autovalore si legge in verticale, gli autovalori sono ordinati da $\lambda_1$ a $\lambda_n$ sull'asse delle ascisse. L'asse delle ordinate da indicazione sul modulo. Dove i pallini sono più fitti e vicini ai valori reali avremo una convergenza più rapida. Vediamo come gli autovalori all'estremità inferiore dello spettro raggiungano più rapidamente il valore corretto.

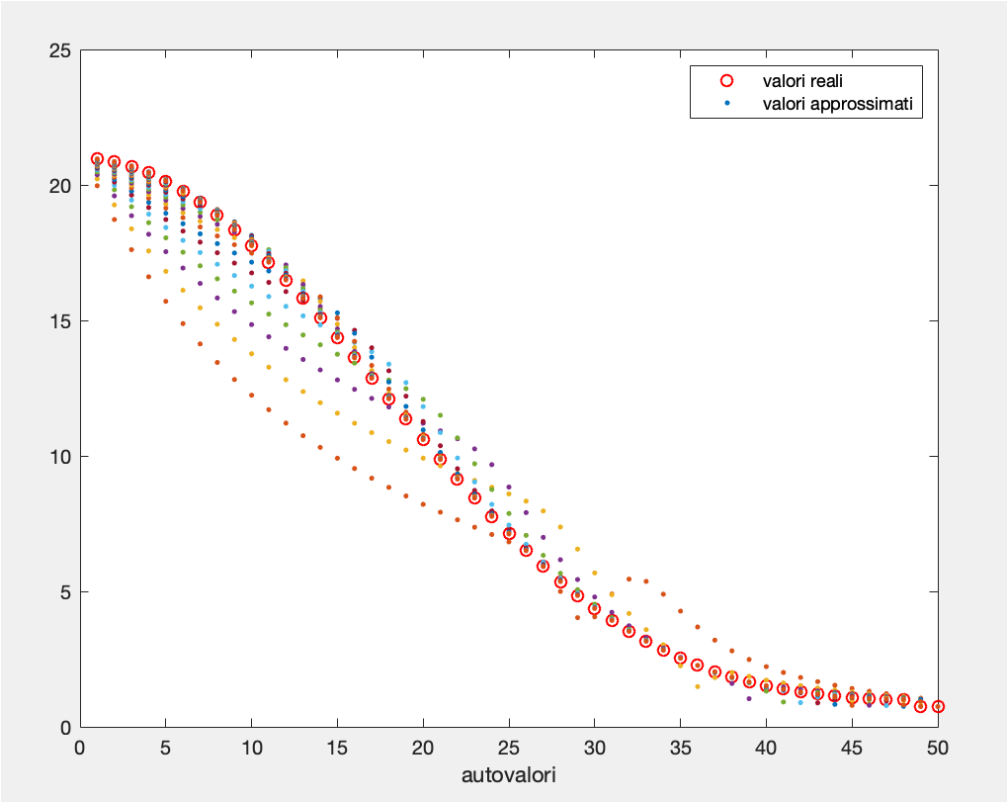
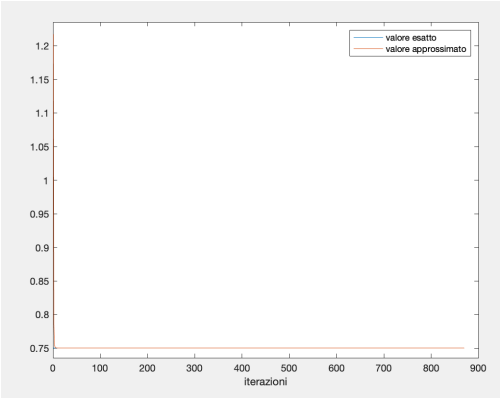
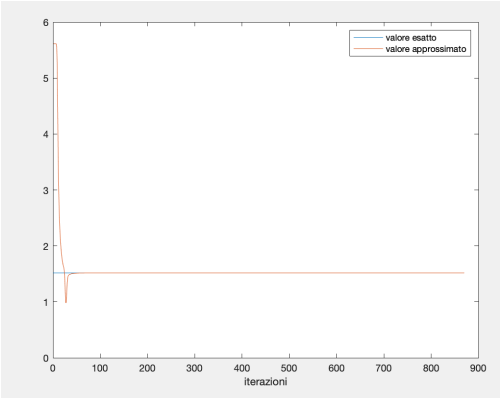


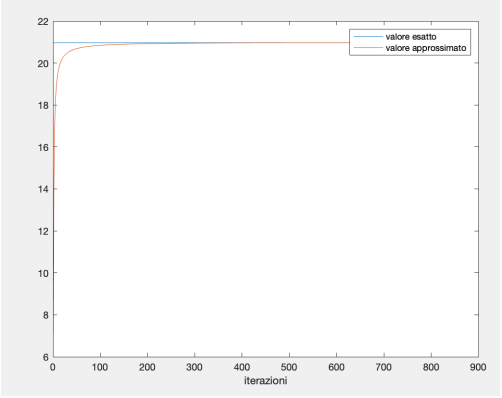

Figura 2.1: Visualizzazione dei valori assunti da ogni autovalore di $C$ durante la convergenza del metodo QR



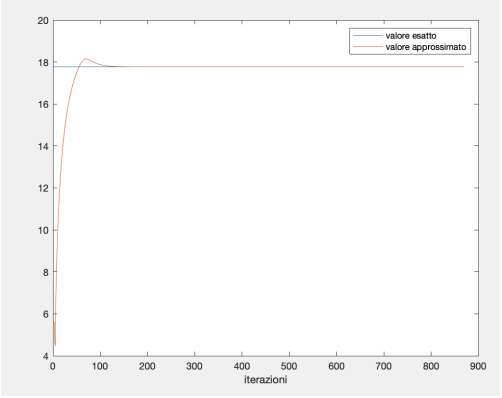

(a) $\lambda_{50}$




(b) $\lambda_{40}$




(c) $\lambda_1$




(d) $\lambda_{10}$

Figura 2.2: Tendenza al valore esatto al progredire delle iterazioni del metodo QR per quattro autovalori di $C$

Si scelgono quattro autovalori di $C$ e si riportano in figura 2.2 gli andamenti dei valori calcolati fino al raggiungimento della convergenza. Anche in questa visualizzazione vediamo come $\lambda_{50}$ e $\lambda_{40}$ si assestino più rapidamente vicino ai loro valori corretti.

La tolleranza utilizzata in questo caso è $10^{-8}$

## 2.2 Metodo di Lanczos

La convergenza del metodo QR in genere è veloce ma, come si è appena descritto, il costo computazionale del metodo è molto alto, a maggior ragione se la matrice A oggetto di analisi non è sparsa. Man mano che le dimensioni di A crescono diventa proibitivo calcolarne gli autovalori. Il metodo di Lanczos, tramite un'operazione di similitudine, permette di trasformare una qualsiasi matrice A simmetrica e definita positiva in una matrice T tridiagonale.

La nuova matrice T avrà solamente 3 valori non nulli per colonna e questo renderà più rapida ed efficiente ogni iterazione del metodo QR.

Ipotizziamo di generare in $\mathbb{R}^n$, una successione di vettori ortonormali a partire da un vettore $\vec{v}_1$ arbitrario di norma euclidea unitaria seguendo una procedura di ortogonalizzazione di Gram-Schmidt.
Il vettore $k+1esimo$ sarà trovato con la seguente relazione:

$$\vec{v}_{k+1} = A\vec{v}_k - \sum_{j=1}^{k} h_{j,k}\vec{v}_j \tag{2.13}$$

Il nuovo vettore generato con la (2.13) andrà normalizzato. Inoltre, in linea teorica, il vettore $n+1esimo$ sarà il vettore nullo perchè contemporaneamente ortogonale a $n$ vettori in uno spazio vettoriale $\mathbb{R}^n$.

Essendo i vettori della successione ortogonali fra loro, il prodotto scalare dovrà essere nullo.

$$\vec{v}_j^T \vec{v}_{k+1} = \vec{v}_j A\vec{v}_k - h_{j,k}\vec{v}_j^T\vec{v}_j = 0 \tag{2.14}$$

Sapendo che $\vec{v}_j^T\vec{v}_j = 1$, posso quindi ricavare $h_{j,k}$ come:

$$h_{j,k} = \vec{v}_j^T A\vec{v}_k \tag{2.15}$$

Si dimostra che $h_{j,k} = 0 \quad \forall \quad j \leq k-2$

$$h_{j,k} = \vec{v}_j^T A\vec{v}_k = \vec{v}_k^T A\vec{v}_j = \vec{v}_k^T\left(\parallel \vec{v}_{j+1} \parallel_2 \vec{v}_{j+1} + \sum_{i=1}^{j} h_{i,j}\vec{v}_i\right) = 0 \quad per\ j \leq k-2 \tag{2.16}$$

Infatti riscrivendo la (2.15) osservo che, per via dell'ortogonalità dei vettori della successione, il risultato della (2.16) sarà sempre zero per $j \leq k-2$.

Pertanto gli unici valori non nulli si troveranno per $h_{k,k}$, $h_{k-1,k}$ e $h_{k,k+1}$ (in seguito chiamati $\alpha_k$, $\beta_k$ e $\beta_{k+1}$)

I termini sulla diagonale principale si calcolano come:

$$h_{k,k} = \vec{v}_k^T A\vec{v}_k \tag{2.17}$$

I termini sulla sovradiagonale e sottodiagonale si calcolano mediante una norma-2 come mostrato nella seguente equazione:

$$h_{k,k+1} = \vec{v}_k^T A\vec{v}_{k+1} = \vec{v}_k^T(\parallel \vec{v}_{k+1} \parallel_2 \vec{v}_{k+1} + h_{k,k}\vec{v}_k + h_{k-1,k}\vec{v}_{k-1}) = \parallel \vec{v}_{k+1} \parallel_2 \tag{2.18}$$

Con gli $n$ vettori generati con la (2.13) costruisco una matrice $V_n$ le cui colonne saranno gli stessi vettori, mentre con i termini $\alpha_k$, $\beta_k$ e $\beta_{k+1}$ costruisco la matrice $T_n$. Ottengo che:

$$V_n T_n = A V_n \tag{2.19}$$

Sapendo che $V_n$ è ortogonale per costruzione riscrivo la (2.19) come:

$$T_n = V_n^T A V_n \tag{2.20}$$

Dalla (2.20) vedo chiaramente che la matrice $T_n$ è ottenuta con un'operazione di similitudine dalla matrice A di partenza, di conseguenza $T_n$ avrà mantenuto gli autovalori di A. Posso ora applicare il metodo QR a $T_n$ che essendo tridiagonale assicura un minor costo computazionale.

**Algorithm 6** Metodo di Lanczos

**procedure** LANCZOSMETHOD(A,it)
    ▷ Preparo valori di partenza
    $T = zeros(it)$
    $[n] = size(A, 1)$
    $\vec{v}_1 = zeros(n)$; $\vec{v}_1(1) = 1$;
    $\beta = 0, \vec{v}_0 = zeros(n), \quad \alpha = \vec{v}_1' A \vec{v}_1$;
    $T_{1,1} = \alpha$

    **for** $k = 1 \to it - 1$ **do**
        $\vec{v}_2 = A\vec{v}_1 - \alpha\vec{v}_1 - \beta\vec{v}_0$;
        $\beta = norm(\vec{v}_2)$; ▷ Aggiorno $\beta$
        $\vec{v}_2 = \vec{v}_2/\beta$; ▷ Normalizzo $\vec{v}_2$
        $\alpha = \vec{v}_2' A \vec{v}_2$; ▷ Aggiorno $\alpha$
        ▷ inserisco i valori di $\alpha$ e $\beta$ nella matrice $T$
        $T_{k+1,k+1} = \alpha$
        $T_{k,k+1} = T_{k+1,k} = \beta$;
        $\vec{v}_0 = \vec{v}_1 \quad \vec{v}_1 = \vec{v}_2$
    **end for**
    $T = sparse(T)$;
**end procedure**

### 2.2.1 Problema degli autovalori migranti

Il metodo di Lanczos può essere utilizzato per determinare anche solamente le estremità dello spettro. Immaginiamo di considerare $m < n$ e di fermarci alla $m - esima$ iterazione dell'algoritmo 6, quello che avremmo ottenuto sarà una matrice $T_m$ simmetrica e definita positiva.

$T_m$ essendo un minore principale di $T_n$ avrà come autovalori un sottoinsieme degli autovalori di $A$

In questi casi si verifica che l'insieme degli autovalori di $T_m$ contiene un sottoinsieme degli autovalori estremi di $A$. Quindi se l'interesse applicativo si concentra sulle estremità dello spettro di $A$ è possibile operare nel seguente modo.

Si considera nuovamente la matrice $C$ introdotta nel capitolo 1.

In tabella 2.1 vengono riportati gli autovalori estremi ottenuti dalle matrici $T_{30}$,$T_{40}$ e $T_{50}$ generate variando in input, sull'algoritmo 6, il valore di $it$. Con

$m = 30$ e con $m = 40$ si possono distinguere parte degli autovalori di $C$, una corrispondenza migliore si ha per gli autovalori più grandi.

Dai risultati ottenuti con $m = 50$ ci si aspetterebbe di ottenere lo spettro completo di $C$, però quello che succede è che compaiono alcuni autovalori che non hanno nulla a che vedere con $C$. Questi ultimi sono autovalori spuri causati dagli errori di arrotondamento che fanno perdere via via l'ortogonalità dei vettori $\vec{v}_k$. Vengono chiamati anche autovalori migranti (sottolineati nelle tabelle) perché tendono a cambiare mano a mano che $m$ viene aumentato, mentre gli autovalori veri di $A$ tendono a rimanere stabili. La loro presenza rende impossibile determinare l'insieme completo di autovalori di $C$ applicando Lanczos per $m = n$.

| $\lambda(C)$ | $\lambda(T_{50})$ | $\lambda(T_{40})$ | $\lambda(T_{30})$ |
|---|---|---|---|
| 20.965219707 | 20.965219568 | 20.965211576 | 20.964850280 |
| 20.861189485 | 20.861189621 | 20.861195604 | 20.847251837 |
| | <u>20.840579938</u> | | |
| 20.688837421 | 20.688837423 | 20.688839335 | 20.662000527 |
| 20.449697430 | 20.449697430 | 20.449697454 | 20.434855112 |
| | <u>20.201513065</u> | | |
| 20.145890108 | 20.145891060 | 20.145889305 | |
| 19.780096315 | 19.780096316 | 19.780096255 | 19.779752521 |
| 19.355523856 | 19.355523856 | 19.355522882 | 19.184498746 |
| 18.875867748 | 18.875867748 | 18.875867240 | 18.735230095 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 1.205184513 | 1.205178388 | 1.185148608 | |
| 1.138087560 | | | |
| 1.086010240 | 1.086035705 | | 1.107679830 |
| 1.047323300 | 1.047456089 | 1.068353481 | |
| 1.020691316 | 1.020767737 | | |
| 1.005121067 | 1.005124790 | 1.007494219 | 1.011950337 |
| 0.750000000 | 0.750000108 | | |
| 0.749999999 | | | |

Tabella 2.1: Autovalori estremi delle matrici $T_{30}$,$T_{40}$ e $T_{50}$ confrontati con gli autovalori di $C$

La successione di vettori $\vec{v}_k$ (2.13) in aritmetica esatta dovrà fermarsi al passo $n + 1$ generando il vettore nullo in quanto lo spazio vettoriale in cui si genera la successione è $\mathbb{R}^n$.

Implementando l'algoritmo 6 su Matlab è possibile continuare a generare vettori con la (2.13) anche dopo il passo $n$. Questo accade per via dell'accumularsi di errori di arrotondamento. È possibile proseguire con le iterazioni anche per $m > n$. Si creano quindi, a partire da $C$, le matrici $T_{60}$ e $T_{80}$.

In questo caso necessariamente saranno presenti ulteriori autovalori spuri, ma come vediamo dalla tabella 2.2 la totalità degli autovalori estremi di $C$ sono rappresentati in modo molto accurato in $T_{60}$ e $T_{80}$. Notiamo inoltre che si generano delle molteplicità fittizie, ad esempio $\lambda_1$ in $T_{80}$ ha molteplicità algebrica pari a due.

In genere non sappiamo quali sono gli autovalori che stiamo cercando. Per distinguere gli autovalori spuri da quelli corretti è sufficiente cambiare il vettore iniziale $\vec{v}_1$ nell'algoritmo 6. Come è facile immaginare gli autovalori corretti

saranno indipendenti dal vettore iniziale e resteranno fissi, tuttavia quelli spuri saranno necessariamente dipendenti da $\vec{v}_1$.

Generando con il metodo di Lanczos due matrici $T_m$, variando $\vec{v}_1$ e applicando successivamente il metodo QR sarà possibile, semplicemente confrontando le diagonali delle due matrici $R$, risalire allo spettro reale di $A$.

In tabella 2.3 vengono confrontati i risultati ottenuti generando due diverse matrici $T_{60}$ a partire da $C$. Gli autovalori corretti trovati sono precisi e non variano mentre quelli migranti sono diversi e perciò facilmente individuabili.

| $\lambda(C)$ | $\lambda(T_{80})$ | $\lambda(T_{60})$ |
|---|---|---|
| 20.965219707 | 20.965219707 | 20.965219707 |
| | 20.965219706 | |
| 20.861189485 | 20.861189485 | 20.861189485 |
| | 20.861185306 | 20.862554350 |
| | | 20.702974679 |
| 20.688837421 | 20.688837419 | 20.688837421 |
| | 20.688841566 | |
| 20.449697430 | 20.449697430 | 20.449697430 |
| | 20.449697413 | 20.448310905 |
| 20.145890108 | 20.145890108 | 20.145890108 |
| | 20.145890083 | |
| | | 20.058007058 |
| 19.780096315 | 19.780096315 | 19.780096315 |
| | 19.780096299 | |
| 19.355523856 | 19.355523856 | 19.355523856 |
| | 19.355523790 | |
| | | 19.348888466 |
| 18.875867748 | 18.875867748 | 18.875867748 |
| ⋮ | ⋮ | ⋮ |
| 1.138087560 | 1.138087560 | 1.138087560 |
| 1.086010240 | 1.086010240 | 1.086010240 |
| 1.047323300 | 1.047323300 | 1.047323300 |
| 1.020691316 | 1.020691316 | 1.020691316 |
| 1.005121067 | 1.005121067 | 1.005121067 |
| 0.750000000 | 0.750000000 | |
| 0.749999999 | 0.749999999 | 0.749999999 |

Tabella 2.2: Confronto tra gli autovalori estremi di $C$ con quelli delle matrici $T_{80}$ e $T_{60}$

| $\lambda(C)$ | $\lambda(T_{60})(a)$ | $\lambda(T_{60})(b)$ |
|---|---|---|
| 20.965219707 | 20.965219707 | 20.965219707 |
| | 20.960596476 | |
| 20.861189485 | 20.861189485 | 20.861189485 |
| | | 20.862554350 |
| | <u>20.839708382</u> | <u>20.702974679</u> |
| 20.688837421 | 20.688839528 | 20.688837421 |
| 20.449697430 | 20.449697430 | 20.449697430 |
| | | <u>20.448310905</u> |
| 20.145890108 | 20.145890108 | 20.145890108 |
| | <u>20.092000353</u> | <u>20.058007058</u> |
| 19.780096315 | 19.780096315 | 19.780096315 |
| | <u>19.523513969</u> | <u>19.348888466</u> |
| 19.355523856 | 19.355.523.990 | 19.355523856 |
| 18.875867748 | 18.875867748 | 18.875867748 |
| ⋮ | ⋮ | ⋮ |

Tabella 2.3: Influenza di $\vec{v}_1$ sugli autovalori di $T_{60}$. Gli autovalori migranti sono sottolineati

# Capitolo 3

# Applicazione degli algoritmi ad una matrice test

Come già detto nell'introduzione, la ricerca degli autovalori di matrici SDP ha un riscontro pratico in molti problemi di ingegneria.

Una matrice SDP, ad esempio, può generarsi dalla soluzione agli elementi finiti (FEM) di equazioni differenziali alle derivate parziali (PDE).

Queste equazioni sono importanti nell'ambito dell'ingegneria civile perché regolano diversi processi fisici di interesse. Esse sono alla base dello studio della deformazione di una membrana elastica oppure l'andamento del potenziale idraulico nel flusso di un fluido attraverso un mezzo poroso.

In questo caso, l'equazione che permette la soluzione in stazionario di questi problemi è l'equazione differenziale di Poisson.

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f$$

In seguito viene illustrato come a partire da quest'ultima equazione si costruisce la matrice di rigidezza $H$. Ad essa verranno applicati i metodi descritti nei precedenti capitoli al fine di determinare le principali caratteristiche del suo spettro.

## 3.1 Soluzione agli elementi finiti dell'equazione di Poisson

Partiamo dall'equazione di Poisson:

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f \tag{3.1}$$

$$\begin{cases} u(x,y) = \overline{u}(x,y) & \forall (x,y) \in \Omega_D \\ \frac{\partial u}{\partial n}(x,y) = q(x,y) & \forall (x,y) \in \Omega_N \end{cases} \tag{3.2}$$

Definito un dominio $\Omega$ e le condizioni al contorno (3.2), l'interesse è quello di trovare una soluzione all'equazione (3.1). Si definisce uno spazio funzionale, chiamato spazio trial, $S_n = \xi_1, \xi_2, \cdots, \xi_n$. $\hat{u}_n$ è una combinazione lineare delle

funzioni appartenenti a $S_n$ ed approssima la soluzione esatta $u$.

$$\hat{u}_n(x,y) = \sum_{j=1}^{n} \alpha_j \xi_j(x,y) \tag{3.3}$$

L'applicazione del metodo variazionale di Galerkin permette di trovare l'approssimazione migliore di $\hat{u}_n$ all'interno di $S_n$ trovando i coefficienti $\alpha$ presenti nell'equazione (3.3) tramite l'imposizione della seguente condizione:

$$\sum_{i=1}^{n} \int\int_{\Omega} \left( f - \frac{\partial^2 \hat{u}_n}{\partial x^2} + \frac{\partial^2 \hat{u}_n}{\partial y^2} \right) \xi_i d\Omega = 0 \tag{3.4}$$

ovvero imponendo l'ortogonalità tra il residuo, individuato come $f - \nabla^2 u$, e uno spazio funzionale, chiamato spazio test, che coincide con $S_n$. Applicando il lemma di Green, la (3.4) diventa:

$$-\sum_{i=1}^{n} \int\int_{\Omega} \left( \frac{\partial \hat{u}}{\partial x}\frac{\partial \xi_i}{\partial x} + \frac{\partial \hat{u}}{\partial y}\frac{\partial \xi_i}{\partial y} \right) d\Omega + \int\int_{\partial\Omega_N} \xi_i \left( \frac{\partial \hat{u}_n}{\partial x} n_x + \frac{\partial \hat{u}_n}{\partial y} n_y \right) dS = \int\int_{\Omega} f \xi_i d\Omega \tag{3.5}$$

Inserendo la definizione di $\hat{u}_n$ otteniamo:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \int\int_{\Omega} \left( \frac{\partial \xi_j}{\partial x}\frac{\partial \xi_i}{\partial x} + \frac{\partial \xi_j}{\partial y}\frac{\partial \xi_i}{\partial y} \right) \alpha_j d\Omega = -\int\int_{\Omega} f \xi_i d\Omega + \int\int_{\partial\Omega_N} \xi_i q \, dS \tag{3.6}$$

Riscrivendo la (3.6), individuiamo in $H$ la matrice di rigidezza, in $\vec{\alpha}$ i coefficienti della combinazione lineare (3.3) ed in $\vec{f}$ il termine noto.

$$\mathbf{H}\vec{\alpha} = \vec{f} \tag{3.7}$$

Al fine di definire le funzioni $\xi_i$ discretizziamo il dominio $\Omega = [(x,y) : -1 \leq x \leq 1, -1 \leq y \leq 1]$ in elementi finiti triangolari. Ad ogni nodo $i$ o vertice degli elementi è associata una funzione $\xi_i$. Più la discretrizzazione è fitta più grandi saranno le dimensioni di $H$. Le funzioni $\xi_i$ sono a supporto locale, ovvero sono non nulle solamente in un intorno del rispettivo nodo $i$. Ciò significa che la matrice di rigidezza $H$ generata sarà sparsa. Il generico elemento di $H$ sarà così determinato:

$$h_{i,j} = \sum_{e} h_{i,j}^{(e)} = \sum_{e} \int\int_{\Omega^{(e)}} \left( \frac{\partial \xi_j^{(e)}}{\partial x}\frac{\partial \xi_i^{(e)}}{\partial y} + \frac{\partial \xi_j^{(e)}}{\partial x}\frac{\partial \xi_i^{(e)}}{\partial y} \right) d\Omega^{(e)} \tag{3.8}$$

Per ultima cosa dovranno essere imposte le condizioni al contorno di Dirichlet (3.2) La matrice $H$ così generata sarà per costruzione simmetrica e definita positiva. Suddividendo $\Omega$ in 10240 elementi finiti triangolari e 5249 nodi (figura 3.1), otteniamo una matrice di rigidezza $H$ di ordine $n = 5249$ su cui possiamo applicare gli algoritmi descritti in precedenza.

In figura 3.2 troviamo la rappresentazione degli autovalori di H calcolati con la funzione *eig* di Matlab. Il cluster di autovalori pari a 3 è dovuto all'imposizione delle condizioni al contorno di Dirichlet.

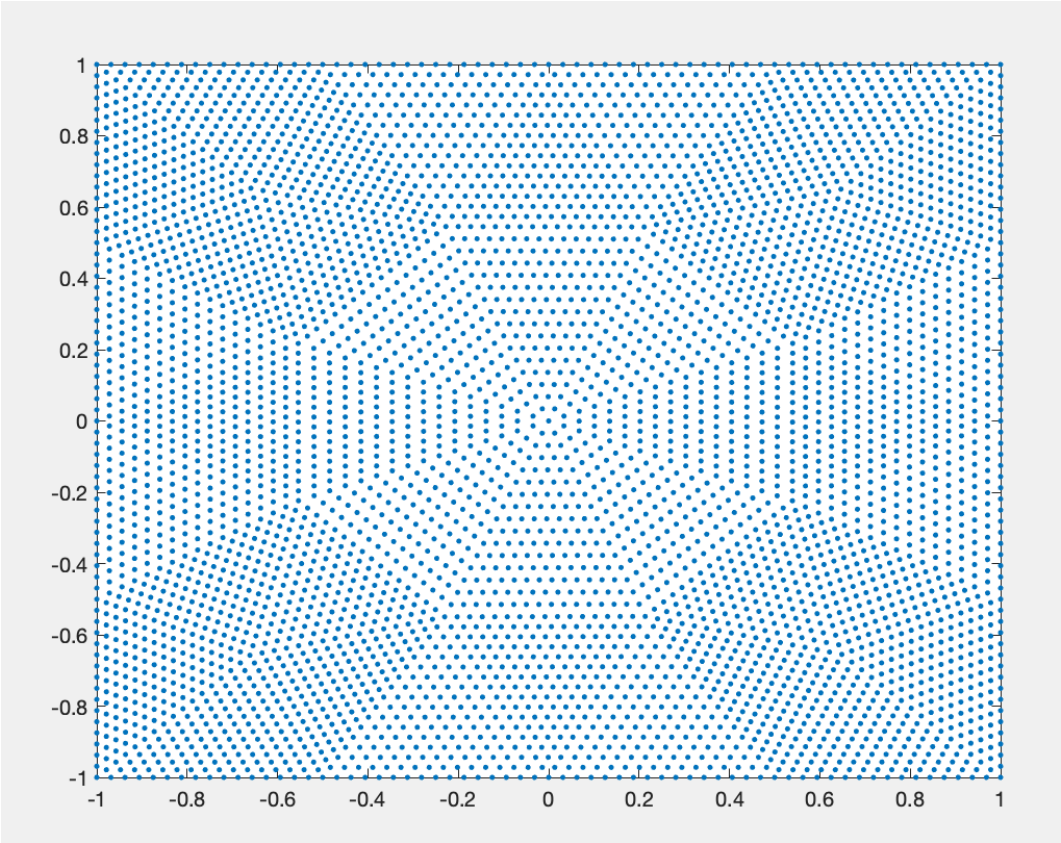



Figura 3.1: Discretizzazione dominio $\Omega$ in 5249 nodi.

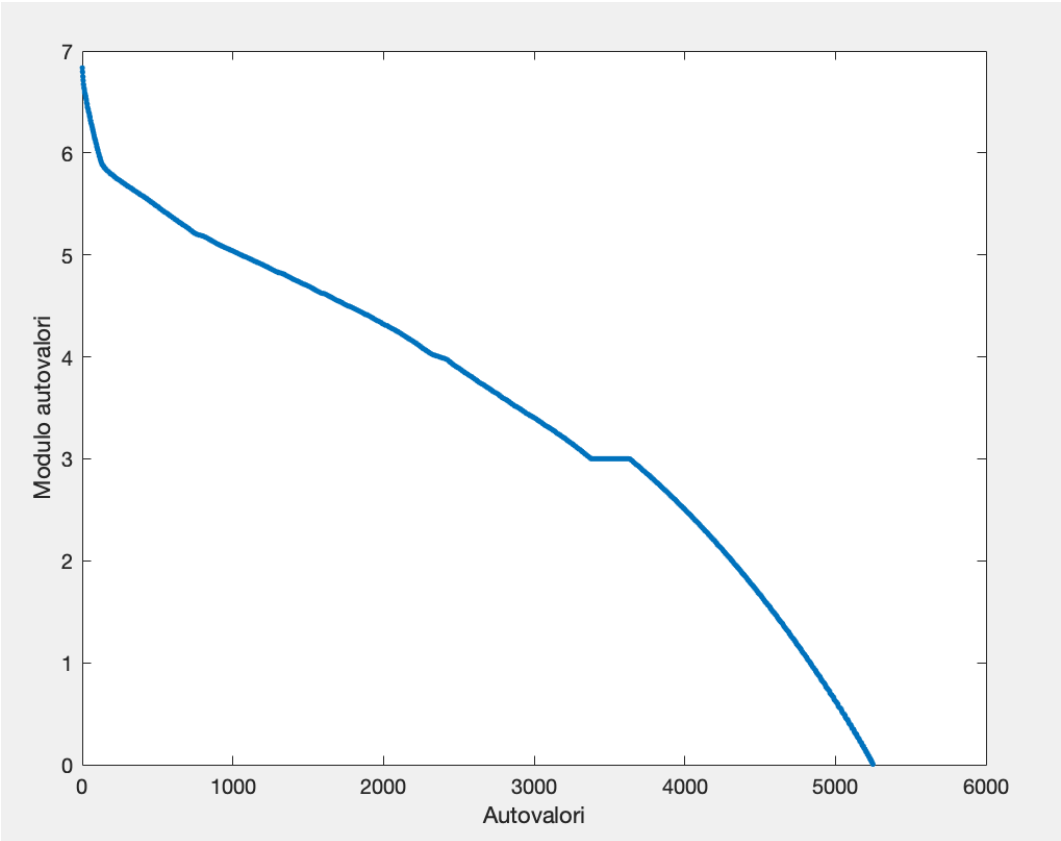
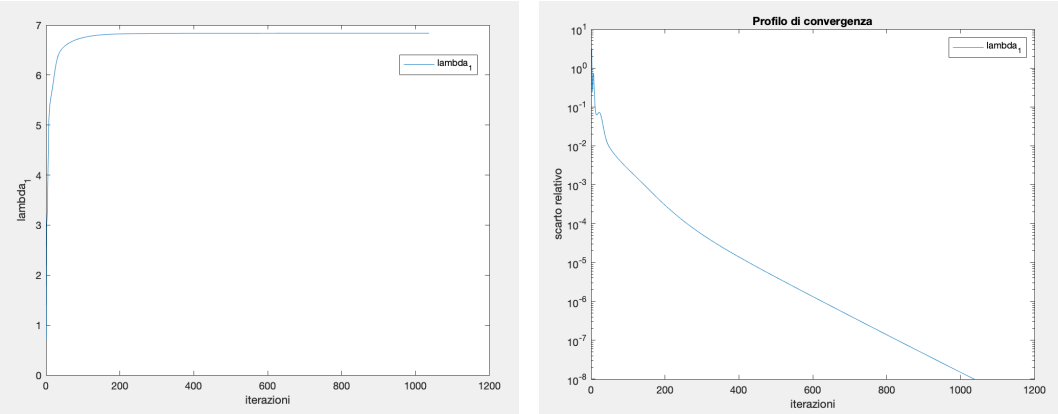



Figura 3.2: Rappresentazione grafica dello spettro della matrice $H$

Si determina il valore di $\lambda_1$ applicando il metodo delle potenze.

| | valore approssimato | valore reale | iterazioni |
|---|---|---|---|
| $\lambda_1$ | 6.834666078 | 6.834666732 | 1037 |

Tabella 3.1: Risultati numerici dell'applicazione del metodo delle potenze




(a) Valori assunti da $\lambda_1$

(b) Profilo di convergenza

Figura 3.3: Risultati ottenuti applicando ad $H$ il metodo delle potenze. $\vec{z}_0$ random e toll=$10^{-8}$

Si determina il valore di $\lambda_n$ applicando il gradiente coniugato applicato al quoziente di Rayleigh.

Al variare del precondizionatore troviamo i seguenti risultati:

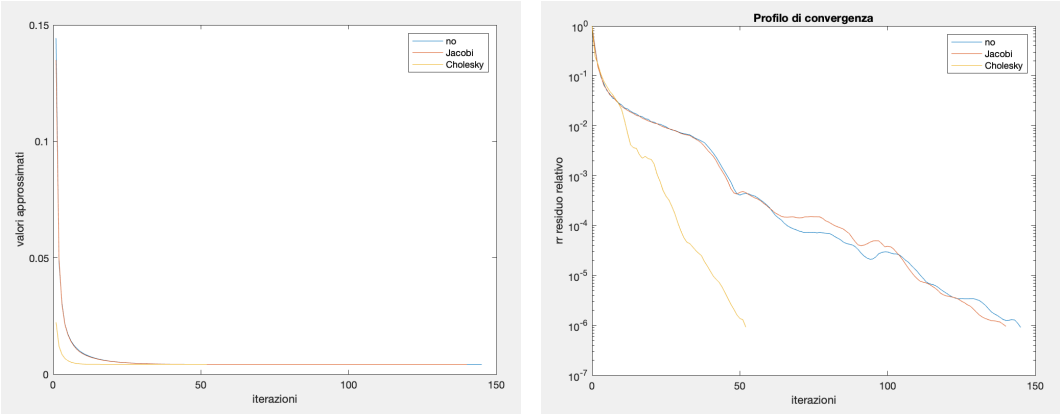
| Valore approssimato | Iterazioni | Precondizionatore |
|---|---|---|
| 0.004279327 | 145 | $K^{-1} = I$ |
| 0.004279327 | 140 | $K^{-1} = F^{-1}$ |
| 0.004279327 | 52 | $K^{-1} = (\tilde{L}\tilde{L}^T)^{-1}$ |

Tabella 3.2: Risultati del calcolo di $\lambda_n$ al variare del precondizionatore

In questo caso, $K^{-1} = (\tilde{L}\tilde{L}^T)^{-1}$ permette di ottenere i risultati migliori in termini di convergenza, giustificando il suo maggior costo computazionale.

L'efficacia del precondizionatore di Jacobi è limitata, infatti rispetto allo schema non modificato l'accelerazione ricavata è minima. L'effetto di $K^{-1} = F^{-1}$ dipende dalla matrice a cui è applicato, se quest'ultima non è fortemente diagonalmente dominante la performance diminuisce.

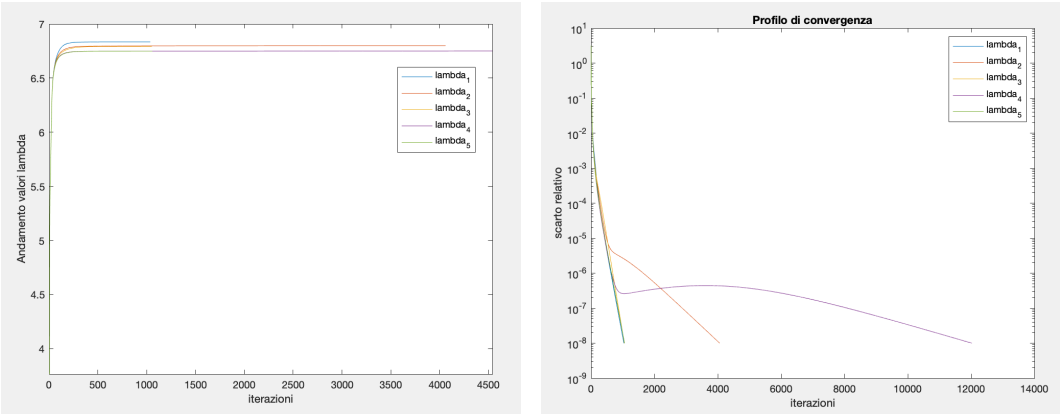
In figura 3.4 si riportano graficamente i profili di convergenza.




(a) Valori assunti da $\lambda_n$ (b) Diagrammi di convergenza

Figura 3.4: Risultati ottenuti applicando il GC al quoziente di Rayleigh al variare dei precondizionatori. $\vec{z}_0$ random e tolleranza pari a $= 10^{-6}$

Al fine di determinare più autovalori estremi di $H$ si deflaziona prima il metodo delle potenze e successivamente il gradiente coniugato. Si determinano i primi cinque autovalori di $H$.




(a) Valori assunti da $\lambda$ (b) Profili di convergenza

Figura 3.5: Risultati ottenuti deflazionando la matrice $H$ con il metodo delle potenze

Come ci aspettiamo gli autovalori più vicini al successivo convergono più lentamente. È il caso di $\lambda_2$ e $\lambda_4$.

| | valore approssimato | valore esatto | iterazioni |
|---|---|---|---|
| $\lambda_1$ | 6.834666078 | 6.834666732 | 1037 |
| $\lambda_2$ | 6.798588849 | 6.798592007 | 4061 |
| $\lambda_3$ | 6.791943008 | 6.791939763 | 1052 |
| $\lambda_4$ | 6.750916999 | 6.750928710 | 12023 |
| $\lambda_5$ | 6.748866443 | 6.748854826 | 1056 |

Tabella 3.3: Confronto tra i valori reali ed approssimati per gli autovalori più grandi di $H$

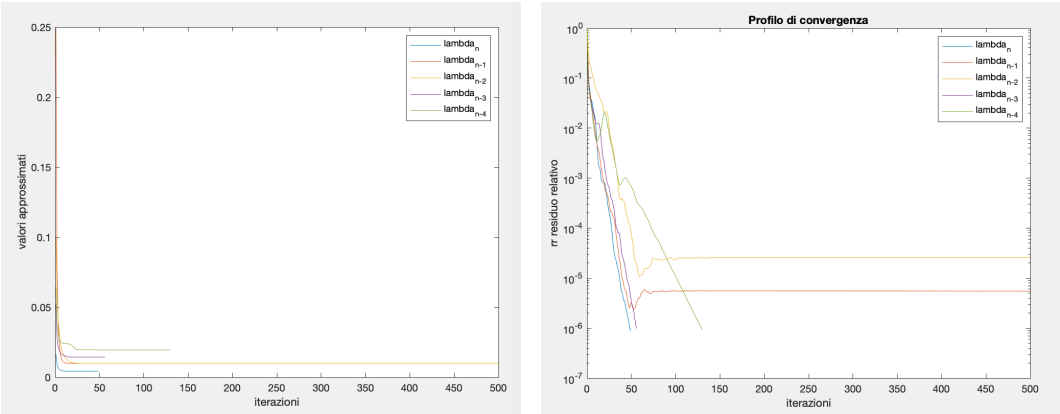
Per il metodo delle potenze conosciamo la velocità di convergenza teorica data da $2log(\lambda_2/\lambda_1)$, quest'ultima non è altro che la pendenza del profilo di convergenza. Nella seguente tabella vengono confrontate le pendenze teoriche

con le pendenze in condizioni asintotiche dei profili di convergenza in figura 3.5b

| | pendenza teorica | pendenza asintotica |
|---|---|---|
| $\lambda_1$ | - 4.596 $\times 10^{-3}$ | - 4.871 $\times 10^{-3}$ |
| $\lambda_2$ | - 8.503 $\times 10^{-4}$ | - 8.318 $\times 10^{-4}$ |
| $\lambda_3$ | - 5.260 $\times 10^{-3}$ | - 5.485 $\times 10^{-3}$ |
| $\lambda_4$ | - 2.669 $\times 10^{-4}$ | - 2.613 $\times 10^{-4}$ |

Tabella 3.4: Confronto tra la pendenza teorica ed effettiva del diagramma di convergenza del metodo delle potenze

Si determinano i cinque autovalori di modulo minore di $H$.




(a) Valori assunti da $\lambda$

(b) Profili di convergenza

Figura 3.6: Risultati ottenuti deflazionando la matrice $H$ con il GCM applicato al $q(\vec{z})$. Si utilizza il precondizionatore di Cholesky che sappiamo essere più efficiente.

Noto che per $\lambda_{n-1}$ e $\lambda_{n-2}$ il residuo relativo dopo un certo numero di iterazioni si stabilizza su un valore costante. Questo fenomeno di stagnazione è dovuto al fatto che il residuo per sua natura potrebbe non scendere sotto un certo valore a causa della precisione di macchina limitata. Questo accade in genere per gli autovalori multipli dove è più difficile separarli. Non è necessario proseguire ulteriormente con le iterazioni, i valori ottenuti saranno meno precisi.

| | valore approssimato | valore esatto | iterazioni |
|---|---|---|---|
| $\lambda_n$ | 0.004279327 | 0.004279297 | 49 |
| $\lambda_{n-1}$ | 0.009815002 | 0.009814524 | 500 |
| $\lambda_{n-2}$ | 0.009815312 | 0.009815778 | 500 |
| $\lambda_{n-3}$ | 0.014291919 | 0.014291954 | 42 |
| $\lambda_{n-4}$ | 0.019442287 | 0.019442281 | 130 |

Tabella 3.5: Confronto tra i valori reali ed approssimati per gli autovalori più piccoli di $H$

Infine, si creano le matrici $T_{500}$ e $T_{1000}$ con l'algoritmo di Lanczos (6) e successivamente si applica il metodo QR per determinarne gli autovalori.

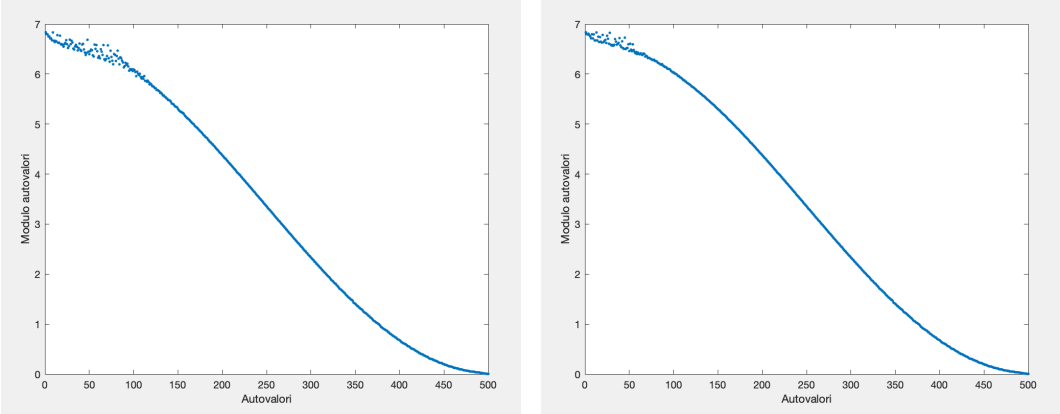
Osservando nelle figure 3.7 e 3.8 si vede come gli autovalori più piccoli si stabilizzino più rapidamente su un valore costante rispetto agli altri.

In entrambi i casi il numero di iterazioni massime impostato è stato raggiunto, questo perché parte degli autovalori di modulo maggiore non hanno raggiunto la convergenza. Si noti come esempio, in tabella 3.6, che la precisione delle approssimazioni di $\lambda_2$ o $\lambda_3$ sia inferiore rispetto a quelle di $\lambda_n$ o $\lambda_{n-1}$.

La convergenza del metodo QR dipende dal seguente rapporto:

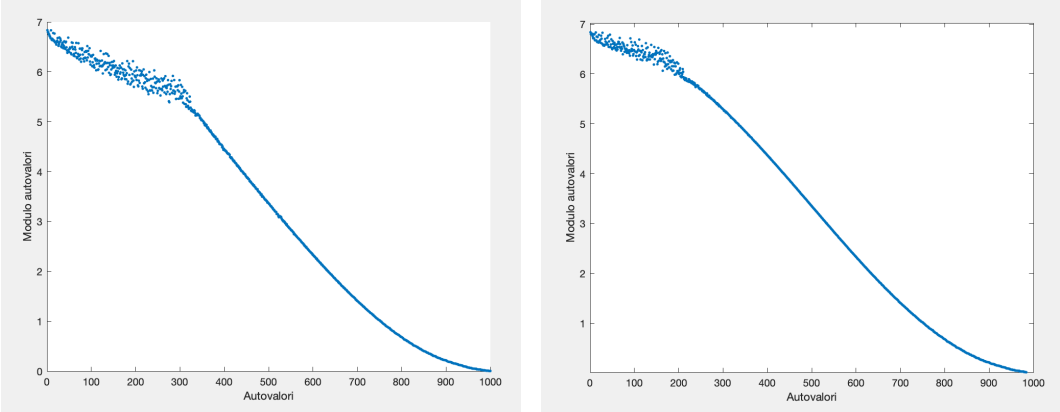
$$max_{1<i<n-1} \left| \frac{\lambda_{i+1}}{\lambda_i} \right| \tag{3.9}$$

per massimizzare 3.9 si usano delle tecniche di traslazione dello spettro, utilizzando il valore di un autovalore noto di $H$.




(a) 1500 iterazioni (b) 3000 iterazioni ($itmax$)

Figura 3.7: Confronto grafico degli autovalori della matrice $T_{500}$ dopo $n$ iterazioni con il metodo QR




(a) 1500 iterazioni (b) 3000 iterazioni ($itmax$)

Figura 3.8: Confronto grafico degli autovalori della matrice $T_{1000}$ dopo $n$ iterazioni con il metodo QR

Confrontando con la figura 3.2, l'andamento dello spettro dei due minori principali rispecchia grossomodo quello della matrice $H$, si perdono le molteplicità create dall'inserimento delle condizioni al contorno. Analizzando con più attenzione le estremità dello spettro, in tabella 3.6, vediamo come esse siano ben rappresentate anche per minori principali molto più piccoli di $H$, in questo caso $T_{1000}$ e $T_{500}$.

In $T_{1000}$ si creano diverse molteplicità fittizie, ad esempio su $\lambda_n$. Inoltre, confrontando le due matrici $T_{500}$ create a partire da due diversi vettori $\vec{v}_1$ è possibile individuare qualche autovalore migrante. Se l'interesse è determinare l'intero insieme $\lambda(H)$, non sarà sufficiente creare una matrice di dimensione $n = 5246$ con l'algoritmo di Lanczos, ma sarà necessario generarne una di ordine maggiore.

| $\lambda(H)$ | $\lambda(T_{1000})$ | $\lambda(T_{500})$ $(a)$ | $\lambda(T_{500})$ $(b)$ |
|---|---|---|---|
| 6.834666732 | 6.834666732 | 6.834666732 | 6.834666732 |
| | | <u>6.824919487</u> | <u>6.807832643</u> |
| | | | <u>6.830657701</u> |
| 6.798592007 | 6.798551277 | 6.798266978 | 6.798551198 |
| | | 6.798105007 | |
| | | | <u>6.795585806</u> |
| | 6.830545013 | | |
| | 6.808290848 | | |
| | 6.775704249 | | |
| 6.791939763 | 6.791980453 | 6.792264490 | 6.791980532 |
| | | | 6.791902467 |
| | 6.791901006 | | |
| | | <u>6.759437232</u> | <u>6.776159876</u> |
| 6.750928710 | 6.750227483 | | 6.750227897 |
| 6.748854826 | | | |
| 6.713039519 | 6.713186905 | 6.713024396 | 6.713174876 |
| | | | <u>6.712054099</u> |
| 6.709184641 | 6.711992833 | 6.709203180 | |
| 6.673326091 | 6.673275068 | | 6.67327501 |
| 6.672289591 | 6.672550186 | | 6.672589703 |
| | 6.671330202 | | |
| 6.667461459 | | | |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 0.0342411 | 0.0342402 | 0.0342404 | 0.0342409 |
| | 0.0342378 | | |
| 0.0342293 | 0.0342320 | 0.0342299 | 0.0342298 |
| 0.0241542 | 0.0241540 | 0.0241538 | 0.0241540 |
| | 0.0241540 | | |
| 0.0241532 | 0.0241533 | 0.0241536 | 0.0241534 |
| | 0.0241533 | | |
| 0.0203134 | 0.0203134 | 0.0203134 | 0.0203134 |
| | 0.0203134 | | |
| 0.0194423 | 0.0194423 | 0.0194422 | 0.0194422 |
| | 0.0194423 | 0.0158462 | |
| | 0.0194423 | | |
| 0.0142919 | 0.0142919 | 0.0142919 | 0.0142919 |
| | 0.0142919 | | |
| | 0.0142919 | | |
| | | | <u>0.0111295</u> |
| 0.0098158 | 0.0098158 | 0.0098157 | 0.0098157 |
| | 0.0098158 | | |
| 0.0098145 | 0.0098145 | 0.0098146 | 0.0098145 |
| | 0.0098145 | | |
| | 0.0098292 | | |
| | 0.0098160 | | |
| 0.0042793 | 0.0042793 | 0.0042793 | 0.0042793 |
| | 0.0042793 | 0.0042793 | 0.0042793 |
| | 0.0042793 | | |
| | 0.0042793 | | |

Tabella 3.6: Confronto tra gli autovalori estremi di $H$ con quelli delle matrici $T_{500}$ e $T_{1000}$. Gli autovalori migranti sono sottolineati.

# Capitolo 4

# Conclusioni

In questa tesi sono stati implementati i principali algoritmi numerici in grado di calcolare gli autovalori di matrici simmetriche definite positive.

Inizialmente gli algoritmi sono stati testati sulla matrice $C$ (1.9). Successivamente, nel capitolo 3, i metodi sono stati applicati alla matrice $H$ derivante da applicazioni pratiche.

I risultati ottenuti sono stati confrontati con gli autovalori esatti calcolati con la funzione *eig* di Matlab ed è stato possibile concludere che tutti i metodi permettono di ricavare con sufficiente precisione i dati di interesse.

Si nota come si ottengano valori approssimati più precisi per la parte inferiore dello spettro utilizzando il metodo di Lanczos. Risultati accurati si ottengono anche con l'utilizzo del gradiente coniugato applicato al quoziente di Rayleigh.

Dal momento che l'interesse ingegneristico quasi sempre ricade esclusivamente sugli autovalori più piccoli, disporre di metodi accurati per quei autovalori è fondamentale.

Eventuali sviluppi dello studio potrebbero essere rivolti verso l'implementazione di tecniche di traslazione dello spettro per il metodo QR.

# Bibliografia

[1] G. Gambolati, M. Ferronato. *Lezioni di metodi numerici per l'ingegneria.* Libreria progetto, Padova, 2018.

[2] M. Ferronato. *Esercitazione numerica: soluzione agli elementi finiti dell'equazione della diffusione.*