UNIVERSITÀ DEGLI STUDI DI CAGLIARI
FACOLTÀ DI SCIENZE
Corso di Laurea Magistrale in Matematica

Risoluzione di problemi agli autovalori tramite proiezioni in sottospazi di Krylov

Relatore:
Prof. Giuseppe Rodriguez

Tesi di:
Anna Concas

Anno Accademico 2014/2015

Indice

Introduzione **i**

1 Richiami di Algebra Lineare **1**
1.1 Autovettori e autovalori 1
1.2 Localizzazione degli autovalori 5
1.2.1 Teoremi di Gershgorin 6
1.3 Forme canoniche 10
1.4 Pseudospettri 15
1.4.1 Metodi di calcolo degli pseudospettri 18
1.5 Sistemi lineari e problemi ai minimi quadrati 19
1.5.1 Sistemi lineari e condizionamento 19
1.5.2 Problemi ai minimi quadrati 21

2 Algoritmi numerici **27**
2.1 Il metodo delle potenze 30
2.2 Metodo delle potenze inverse 33
2.3 L'algoritmo QR per gli autovalori 34
2.3.1 L'algoritmo QR classico 34
2.4 Passaggio in forma di Hessenberg 37
2.4.1 L'algoritmo QR con shift 40

3 Metodi di proiezione **43**
3.1 Gradiente e gradiente coniugato 44
3.2 Iterazioni in sottospazi di Krylov 50
3.2.1 Il metodo CG come metodo di Krylov 51
3.3 Il metodo di Arnoldi 53
3.3.1 CGS e MGS 53
3.3.2 L'iterazione di Arnoldi 57
3.3.3 Il metodo GMRES 59
3.3.4 Approssimazione degli autovalori 61
3.4 Il metodo di Lanczos 63

3.4.1 Il metodo MINRES 65
3.4.2 Applicazione al calcolo degli autovalori 66
3.5 Conseguenze del breakdown 69
3.6 Il metodo di Golub-Kahan 71
3.6.1 LSQR 73
3.6.2 Applicazione al calcolo dei valori singolari 75

4 Test numerici **79**
4.1 Approssimazione degli autovalori con Arnoldi 79
4.2 Approssimazione degli autovalori con Lanczos 82
4.3 Approssimazione dei valori singolari 84

5 Conclusioni **91**

Bibliografia **93**

Introduzione

I problemi agli autovalori si presentano in molti campi scientifici ed ingegneristici e quindi l'esigenza di calcolare alcuni o tutti gli autovalori e/o autovettori di una matrice emerge in problemi pratici in diversi settori delle scienze applicate. Nel libro di Y. Saad [11] sono citati alcuni classici esempi di problemi agli autovalori nel campo dell'ingegneria, della chimica e dell'economia. Un altro campo in cui autovalori e autovettori giocano un ruolo fondamentale è quello dei motori di ricerca sul Web e in particolare Google.

Quando si effettua una ricerca su Internet utilizzando un motore di ricerca vi è inizialmente una fase di elaborazione del testo in cui si ha lo scoopo di trovare tutte le pagine sul Web che contengono le parole digitate. Ovviamente a causa dell'enorme grandezza del Web è necessario, ricorrendo ad una misura di qualità, eliminare tutte le pagine meno interessanti. Google utilizza il concetto di *pagerank*come misura di qualità delle pagine Web. Questo concetto si basa sull'assunzione che il numero dei links alla o dalla pagina considerata, dia informazioni circa l'importanza della stessa. Quindi quando inseriamo una o più parole chiave per una ricerca, Google seleziona le pagine contenenti le parole nella sua banca dati e risponde alla richiesta presentando queste pagine in ordine di importanza, o autorevolezza. L'autorevolezza di una pagina p dipende da quante pagine la citano (ovvero contengono un link a p) e da quanto queste pagine sono a loro volta autorevoli. Questo concetto viene utilizzato per la selezione e l'ordinamento delle pagine mediante appunto un algoritmo PageRank, inventato alla fine degli anni '90 dai fondatori di Google, Sergey Brin e Lawrence Page [2]. L'algoritmo usato da Google è sicuramente molto complesso ed è ovviamente un segreto industriale anche se probabilmente il primo algoritmo utilizzato si basava sulla ricerca dell'autovettore principale (metodo delle potenze). Descriviamo brevemente le basi di un semplice algoritmo PageRank, per evidenziare che le tecniche di page ranking sono legate ad un problema agli autovalori e in particolare alla ricercsa degli autovalori di un'opportuna matrice.

Sia P l'insieme delle pagine presenti nel Web attinenti alle parole chiave inserite, n la sua cardinalità e sia i una particolare pagina Web. Denotiamo con

O_i l'insieme delle pagine a cui la pagina i è collegata ovvero gli "outlinks". Il numero degli outlinks è N_i. Denotiamo invece con I_i l'insieme degli "inlinks" ovvero le pagine che hanno un outlink alla i-esima. Facciamo l'ipotesi semplificativa che tutte le pagine di P contengano almeno un link ad un'altra pagina di P stesso. Sotto questa ipotesi possiamo definire l'autorevolezza x_i della i-esima pagina mediante la formula

$$x_i = \sum_{j \in I_i} \frac{x_j}{N_j}. \tag{1}$$

Con la precedente si afferma il principio che una pagina che fa riferimento alla i-esima conta di più se contiene pochi links, fra cui quello che ci interessa, perché è una pagina "selettiva", mentre conta meno se contiene molti links. Preso un vettore iniziale $x^{(0)}$ possiamo considerare l'iterazione

$$x_i^{(k+1)} = \sum_{j \in I_i} \frac{x_j^{(k)}}{N_j}, \quad k = 0, 1, \ldots. \tag{2}$$

La formula 1 non è utilizzabile direttamente per calcolare l'importanza di una singola pagina. Basti pensare al caso non infrequente che due pagine, per esempio la k-esima e la j-esima, si citino a vicenda; in questa situazione la formula diventa inutilizzabile perchè per calcolare x_k devo conoscere x_j e viceversa. Osserviamo però che essa può essere utilizzata per calcolare l'autorevolezza di tutte le n pagine web che stiamo considerando. Scrivendo infatti simultaneamente le n relazioni e indicando con $\mathbf{x}$ il vettore $(x_1, ..., x_n)^T$ si ottiene un sistema della forma

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

In tal modo ci si riconduce ad un problema agli autovalori per una matrice Q matrice quadrata di ordine n rappresentante il grafo di Internet definita come

$$Q_{ij} = \begin{cases} 1/N_j & \text{se esiste un link tra la pagina i e la pagina j} \\ 0 & \text{altrimenti} \end{cases}$$

Da ciò segue che l'iterazione (2) in forma matriciale è equivalente a

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

che rappresenta il metodo delle potenze per il calcolo dell'autovettore principale ossia quello associato all'autovalore di massimo modulo. Una trattazione

completa dell'utilizzo del metodo delle potenze per il calcolo del page ranking si può trovare in [3]. In questa tesi vengono presi in esame problemi agli autovalori e metodi iterativi per la loro risoluzione. In particolare vengono esaminati i metodi di proiezione in sottospazi di Krylov e considerate implementazioni in Matlab dei metodi di Arnoldi e Lanczos per l'approssimazione di autovalori di matrici non Hermitiane ed Hermitiane rispettivamente. Tra i metodi di Krylov viene poi preso in esame il metodo di Golub-Kahan una cui implementazione viene applicata all'approssimazione dei valori singolari di una matrice rettangolare non Hermitiana. Viene quindi presa in considerazione la decomposizione ai valori singolari di una matrice (SVD) le cui proprietà di approssimazione possono essere utilizzate per mostrare l'equivalenza tra la SVD e la *principal component analysis (PCA)*, una tecnica per la semplificazione dei dati il cui scopo è quello di ridurre il numero elevato di variabili rappresentanti altrettante caratteristiche del fenomeno analizzato. La SVD e in particolare il metodo di bidiagonalizzazione di Golub-Kahan vengono utilizzati anche nel *Text Mining*, termine con cui si denotano dei metodi per estrarre informazioni utili da una enorme e spesso priva di struttura collezione di testi.
La SVD o meglio la decomposizione ai valori singolari troncata (TSVD) è un *metodo di regolarizzazione*. Le tecniche di regolarizzazione vengono utilizzate nello studio di sistemi lineari mal posti

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

dove la matrice A dei coefficienti di ordine n è molto mal condizionata ossia il suo nemero di condizionamento $k(A) >> 1$.
Problemi di questo tipo detti **problemi inversi mal posti** si in incontrano spesso nella risoluzione di problemi reali, quali ad esempio, quelli che si affrontano nella elaborazione delle immagini digitali (rimozione della sfuocatura in un'immagine, ricostruzuione di immagini da proiezioni, etc). Un esempio tipico di problema inverso mal posto è rappresentato dalla soluzione numerica della discretizzazione di un'equazione integrale della forma

$$\int_a^b k(x,y)f(y)dy = g(x)$$

dove $f(\cdot)$ rappresenta la funzione incognita, $k(x,y)$ rappresneta il nucleo della funzione integrale e $g(\cdot)$ è la funzione misurata. I metodi di regolarizzazione procedono quindi al calcolo della soluzione di un sistema lineare "approssimante" quello di partenza mal condizionato e meglio condizionato rispetto a quest'ultimo e il primo passo del processo di regolarizzazione di un problema consiste nel calcolo dei valori singolari ottenuto mediante la decomposizione a valori singolari (SVD) della matrice dei coefficienti.

In relazione alle tecniche di regolarizzazione per problemi mal posti si veda [5].

Capitolo 1

Richiami di Algebra Lineare

1.1 Autovettori e autovalori

Sia A una matrice quadrata di ordine n ad entrate reali o complesse, supponiamo ad esempio che $A \in \mathbb{C}^{n\times n}$. Un vettore $\mathbf{x} \in \mathbb{C}^n$ non nullo si dice **autovettore** di A e uno scalare $\lambda \in \mathbb{C}$ è il corrispondente **autovalore** di A se

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

Questa relazione mette in evidenza il fatto che l'azione di una matrice A su un sottospazio S di $\mathbb{C}^n$ potrebbe imitare la moltiplicazione scalare. Quando questo capita, il particolare sottospazio S viene detto autospazio e ogni elemento non nullo $\mathbf{x} \in S$ è un autovettore.
Riscrivendo la precedente relazione nella forma

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

osserviamo che affinché questo sistema lineare omogeneo ammetta una soluzione non nulla, è necessario che il suo determinante

$$p_A(\lambda) = \det(A - \lambda I)$$

si annulli. Essendo $p_A(\lambda)$ un polinomio di grado n in λ, detto **polinomio caratteristico** della matrice A, dal Teorema fondamentale dell'Algebra segue che esistono, nel campo complesso, n autovalori di A che possono essere calcolati determinando le radici di $p_A(\lambda)$.
Per ciascun autovalore λ_k, $k = 1, \ldots, n$, una soluzione non nulla del sistema singolare omogeneo

$$(A - \lambda_k I)\mathbf{x} = 0$$

fornisce il corrispondente autovettore che risulta determinato a meno di una costante moltiplicativa non nulla; quindi se $\mathbf{x}$ è un autovettore di A anche $\alpha\mathbf{x}$, con $\alpha \neq 0$, è un autovettore di A corrispondente allo stesso autovalore.
L'insieme di tutti gli autovalori di una matrice A viene detto **spettro** di A, risulta essere un sottoinsieme di $\mathbb{C}$ denotato con $\sigma(A)$. Si definisce **raggio spettrale** di una matrice A e si denota con $\rho(A)$ il massimo dei moduli dei suoi autovalori

$$\rho(A) = \max_{k=1,\dots,n} |\lambda_k|.$$

Determinante e traccia. Ricordiamo che la *traccia* di una matrice $A \in \mathbb{C}^{n\times n}$ è la somma dei suoi elementi diagonali

$$\text{trace}(A) = \sum_{i=1}^{n} a_{ii}.$$

Osserviamo che sia la traccia che il determinante di una matrice sono legati ai suoi autovalori. Infatti consideriamo il

Teorema 1.1. *Data una matrice $A \in \mathbb{C}^{n\times n}$, il suo determinante* $\det(A)$ *e la sua traccia* $\text{trace}(A)$ *sono uguali, rispettivamente, al prodotto e alla somma degli autovalori di A contati con la rispettiva molteplicità algebrica:*

$$\det(A) = \prod_{i=1}^{n} \lambda_i, \qquad \text{trace}(A) = \sum_{i=1}^{n} \lambda_i.$$

Dimostrazione. Consideriamo il polinomio caratteristico di A $p_A(z) = \det(A - zI)$ le cui radici sono gli autovalori λ_i di A. Pertanto dal Teorema Fondamentale dell'Algebra possiamo scrivere

$$p_A(z) = (z-\lambda_1)\cdots(z-\lambda_n) = \prod_{i=1}^{n}(z-\lambda_i). \tag{1.1}$$

Pertanto calcolando il determinante di A otteniamo

$$\det(A) = (-1)^n \det(-A) = (-1)^n p_A(0) = \prod_{i=1}^{n} \lambda_i$$

e resta cosi provata la prima formula.
Per quanto riguarda la seconda formula si ha che dalla definizione di polinomio caratteristico $p_A(z) = \det(A - zI)$, segue che il coefficiente del termine z^{n-1} di p_A è l'opposto della somma degli elementi diagonali di A ovvero $-\,\text{trace}(A)$. Dalla 1.1 segue che tale coefficiente è anche pari a $-\sum_{i=1}^{n}\lambda_i$ e quindi $\text{trace}(A) = \sum_{i=1}^{n}\lambda_i$. □

Definizione 1.1 (Molteplicità geometrica)**.** Abbiamo accennato al fatto che l'insieme degli autovettori corrispondenti ad un autovalore, insieme al vettore nullo, forma un sottospazio di $\mathbb{C}^n$ noto come autospazio relativo all'autovalore considerato. Se λ è un autovalore di A possiamo denotare con E_λ il corrispondente autospazio la cui dimensione può essere interpretata come il massimo numero di autovettori linearmente indipendenti relativi allo stesso autovalore λ. Questo numero rappresenta la molteplicità geometrica di λ.

Definizione 1.2 (Molteplicità algebrica)**.** Si definisce molteplicità algebrica di un autovalore λ di A la sua molteplicità come radice del polinomio caratteristico di A. Un autovalore viene detto semplice se la sua molteplicità algebrica è 1.

Come abbiamo già accennato, il polinomio caratteristico di una matrice A fornisce un semplice modo per calcolare il numero degli autovalori di A. Infatti si ha il

Teorema 1.2. *Se $A \in \mathbb{C}^{n\times n}$, allora A ha n autovalori contati con la rispettiva molteplicità algebrica. In particolare se le radici di p_A sono semplici allora A ha n autovalori distinti.*

Possiamo osservare che in particolare, la molteplicità algebrica di un autovalore è in generale grande almeno quanto la sua molteplicità geometrica ossia

$$\text{molteplicità geometrica} \leq \text{molteplicità algebrica}.$$

Per provare tale relazione tra le molteplicità geometrica e algebrica occorre considerare una particolare relazione d'uguaglianza tra matrici e fare riferimento quindi alle trasformazioni di similitudine.

Trasformazioni di similitudine e matrici simili. Se $X \in \mathbb{C}^{n\times n}$ è non singolare, allora l'applicazione $A \longmapsto X^{-1}AX$ viene detta trasformazione di similitudine di A.
Due matrici A e B si dicono *simili* se esiste una trasformazione di similitudine che le lega, ossia se esiste una matrice non singolare $X \in \mathbb{C}^{n\times n}$ tale che

$$B = X^{-1}AX.$$

Le matrici simili hanno una importante proprietà come mostrato dal seguente

Teorema 1.3. *Due matrici simili A e B hanno lo stesso polinomio caratteristico e quindi autovalori coincidenti,con le rispettive molteplicità geometrica e algebrica.*

Dimostrazione. Essendo, per ipotesi, A e B simili esiste una matrice X non singolare tale che $B = X^{-1}AX$. La dimostrazione del fatto che i polinomi caratteristici di A e B coincidano segue dalla definizione e dalle proprietà del determinante:

$$\begin{aligned} p_B(z) = p_{X^{-1}AX}(z) &= \det(X^{-1}AX - zI) = \det(X^{-1}(A - zI)X) \\ &= \det(X^{-1})\det(A - zI)\det(X) = \det(A - zI) = p_A(z). \end{aligned}$$

Dalla corrispondenza dei polinomi caratteristici segue che essi hanno le stesse radici e quindi gli autovalori di A e B coincidono così come le loro rispettive molteplicità algebriche. Per provare poi la coincidenza delle molteplicità geometriche, è sufficiente provare che se E_λ è un autospazio di A relativo ad un autovalore λ allora $X^{-1}E_\lambda$ è un autospazio per $X^{-1}AX$ e viceversa. □

Osserviamo che la matrice X rappresenta la trasformazione lineare del cambiamento di base che fa passare dagli autovettori di B agli autovettori di A.
Richiamiamo la seguente definizione che sarà utile in tutta la tesi

Definizione 1.3 (Matrice aggiunta)**.** Data $A \in \mathbb{C}^{m\times n}$, la matrice aggiunta A^* si ottiene scambiando le righe di A con le colonne e coniugandone gli elementi cioè

$$(A^*)_{ij} = \bar{a}_{ji}.$$

Nel caso in cui la matrice A sia reale , l'aggiunta coincide con la trasposta

$$(A^T)_{ij} = a_{ji}.$$

Tornando alla relazione che lega molteplicità algebrica e geometrica siamo ora in grado di provare il seguente

Teorema 1.4. *La molteplicità algebrica di un autovalore λ di una matrice A è grande almeno quanto la sua molteplicità geometrica.*

Dimostrazione. Supponiamo che n sia la molteplicità geometrica di λ autovalore di A. È possibile costruire una matrice $m \times n$ $\hat{V}$ le cui n colonne costituiscano una base ortonormale per l'autospazio $\{x : Ax = \lambda x\}$.
Estendendo poi $\hat{V}$ ad una matrice quadrata V unitaria ($V^*V = VV^* = I$), otteniamo V^*AV nella forma

$$B = V^*AV = \begin{bmatrix} \lambda I & C \\ \mathbf{0} & D \end{bmatrix}$$

dove I è l'identità di ordine n, C è di ordine $n \times (m-n)$ e D è una matrice $(m-n)\times(m-n)$. Dalla definizione di determinante segue che $\det(B - zI) = \det(\lambda I - zI)\det(D - zI) = (\lambda - z)^n \det(D - zI)$. Pertanto la molteplicità algebrica di λ come autovalore di B è almeno n. □

Un autovalore la cui molteplicità algebrica supera la sua molteplicità geometrica e quindi tale per cui nella relazione precedente vale il minore stretto, viene detto *autovalore difettivo*.
Una matrice che ha almeno un autovalore difettivo è detta *difettiva*. Ogni matrice diagonale è non difettiva in quanto per matrici di questo tipo sia la molteplicità geometrica che quella algebrica di un autovalore, coincidono col numero di volte che esso compare nella diagonale.

1.2 Localizzazione degli autovalori

In generale il calcolo numerico degli autovalori e autovettori è un problema delicato sia perché computazionalmente oneroso sia perché in alcuni casi può essere estremamente instabile conducendo ad una forte propagazione degli errori e quindi ad approssimazioni inaccurate.
In alcuni problemi non è necessario conoscere esattamente tutti gli autovalori, ma ci si può accontentare ad esempio di conoscere un'approssimazione solo di alcuni autovalori o di localizzare gli autovalori ossia determinare una regione del piano complesso che contenga tutti gli autovalori.
Quindi spesso ci si può limitare all'individuazione di un sottoinsieme del piano complesso $\mathbb{C}$ in cui siano contenuti tutti gli autovalori o che non contenga alcun autovalore. I risultati di questo tipo sono detti **teoremi di localizzazione** e un primo loro esempio è rappresentato dal seguente teorema che afferma che tutti gli autovalori di una matrice A sono contenuti in un cerchio centrato nell'origine e avente raggio pari ad una qualsiasi norma matriciale consistente di A.
Prima di enununciare il teorema richiamiamo la

Definizione 1.4. Una norma matriciale si dice **consistente** con le norme vettoriali $\|\cdot\|_a$ di $\mathbb{R}^n$ e $\|\cdot\|_b$ di $\mathbb{R}^m$ se

$$\|A\mathbf{x}\|_b \leqslant \|A\|\|\mathbf{x}\|_a, \quad \forall A \in \mathbb{R}^{m\times n}, \forall \mathbf{x} \in \mathbb{R}^n.$$

Tornando al suddetto teorema che evidenzia il legame tra le norma matriciali consistenti e il raggio spettrale di una matrice, si ha

Teorema 1.5. *Data $\|\cdot\|$ una norma consistente, per ogni matrice quadrata A si ha*

$$\rho(A) \leq \|A\|.$$

Dimostrazione. Per ogni autovalore λ di A esiste un autovettore $\mathbf{v}$ tale che $A\mathbf{v} = \lambda\mathbf{v}$. Dalla consistenza e dalla proprietà delle norme, segue che

$$|\lambda|\|\mathbf{v}\| = \|\lambda\mathbf{v}\| = \|A\mathbf{v}\| \leq \|A\| \cdot \|\mathbf{v}\|,$$

da cui, essendo $\mathbf{v} \neq 0$ per definizione di autovettore segue che $|\lambda| \leq \|A\|$ per ogni autovalore λ e quindi la tesi. □

1.2.1 Teoremi di Gershgorin

Tra i teoremi di localizzazione si hanno i **teoremi dei cerchi di Gershgorin** che consentono di identificare le zone contenenti gli autovalori in maniera più rigorosa. Prima di considerare tali teoremi richiamiamo le seguenti definizioni:

Definizione 1.5. Definiamo **cerchi riga** gli insiemi

$$R_i = \Big\{ z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \Big\}, \quad i = 1, \ldots, n.$$

Definizione 1.6. Definiamo **cerchi colonna** gli insiemi

$$C_j = \Big\{ z \in \mathbb{C} : |z - a_{jj}| \leq \sum_{\substack{i=1 \\ i \neq j}}^{n} |a_{ij}| \Big\}, \quad j = 1, \ldots, n.$$

Teorema 1.6 (Primo teorema di Gershgorin)**.** *Data una matrice quadrata A di ordine n, gli autovalori di A sono contenuti nell'unione dei cerchi riga ossia*

$$\sigma(A) \subseteq \bigcup_{i=1}^{n} R_i.$$

Dimostrazione. Indicando con $(\lambda, \mathbf{x})$ una coppia costituita da un autovalore e dal corrispondente autovettore di A, si ha che $A\mathbf{x} = \lambda\mathbf{x}$ ossia

$$\sum_{j=1}^{n} a_{ij} x_j = \lambda x_i, \quad i = 1, \ldots, n.$$

La i-esima componente della precedente equazione può essere scritta nella forma

$$\lambda x_i - a_{ii} x_i = \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j. \tag{1.2}$$

Supponendo che i sia la componente di $\mathbf{x}$ di massimo modulo, cioè

$$|x_i| = \max_{j=1,\ldots,n} |x_j|,$$

dalla (1.2) otteniamo

$$|\lambda - a_{ii}| \cdot |x_i| = |\lambda x_i - a_{ii}x_i| = \Big|\sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij}x_j\Big|$$
$$\leq \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}| \cdot |x_j| \leq |x_i| \cdot \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}|.$$

Dato che $\mathbf{x} \neq \mathbf{0}$ per definizione di autovettore, possiamo dividere entrambi i membri della precedente diseguaglianza per $|x_i|$ ottenendo

$$|\lambda - a_{ii}| \leq \sum_{\substack{j=1\\ j\neq i}}^{n} |a_{ij}|,$$

ossia che l'autovalore λ appartiene al cerchio riga R_i. Dalla generalità dell'autovalore considerato segue la tesi. □

Un risultato analogo vale anche nel caso in cui si considerino i cerchi colonna, cioè

Corollario 1.7. *Assegnata una matrice quadrata A di ordine n, si ha*

$$\sigma(A) \subseteq \bigcup_{j=1}^{n} C_j.$$

Dimostrazione. La dimostrazione è analoga a quella del teorema precedente, applicata a A^T considerando che $\sigma(A) = \sigma(A^T)$. □

Dai precedenti risultati segue immediatamente il

Corollario 1.8. *Indicando con S_R e S_C l'unione, rispettivamente, dei cerchi riga e dei cerchi colonna ossia*

$$S_R = \bigcup_{i=1}^{n} R_i, \quad S_C = \bigcup_{j=1}^{n} C_j,$$

allora risulta che

$$\sigma(A) \subseteq S_R \cap S_C.$$

Consideriamo ora il

Teorema 1.9 (Secondo teorema di Gershgorin)**.** *Se l'insieme S_R è dato dall'unione disgiunta dei due sottoinsiemi*

$$S_1 = \bigcup_{i=1}^{k} R_i, \quad S_2 = \bigcup_{i=k+1}^{n} R_i, \quad S_1 \cap S_2 = \emptyset,$$

allora S_1 conterrà esattamente k autovalori di A contati con le rispettive molteplicità e S_2 conterrà i restanti $n-k$.

Dimostrazione. Consideriamo

$$A_0 = diag(a_{11}, \ldots, a_{nn})$$

la matrice diagonale costituita dagli elementi della diagonale principale della matrice A di partenza e sia $R = A - A_0$.
Prendiamo in considerazione la seguente matrice, dipendente dal parametro reale t

$$A_t = A_0 - tR, \quad 0 \leq t \leq 1.$$

Per $t = 0$ otteniamo la matrice A_0, mentre se fissiamo $t = 1$ otteniamo la matrice A da cui siamo partiti.
Gli autovalori della matrice diagonale A_0 coincidono con i suoi elementi diagonali, pertanto è immediato osservare che i suoi primi k autovalori appartengono all'insieme S_1 essendo i centri dei primi k cerchi riga mentre i restanti $n-k$ appartengono alla regione S_2. Essendo gli autovalori di una matrice delle funzioni continue delle sue componenti, al variare del parametro t tra 0 e 1 gli autovalori della matrice A_t si spostano con continuità nel piano complesso.
Poiché per ipotesi $S_1 \cap S_2 = \emptyset$, nessuno dei primi k autovalori contenuti inizialmente in S_1 può trovarsi nella regione S_2 e gli autovalori non possono passare da una regione all'altra. Otteniamo così la tesi. □

Prima di considerare il terzo teorema di Gershgorin è necessario fare alcuni richiami.

Definizione 1.7. Si definisce **matrice di permutazione** una matrice P che si ottiene permutando tra loro le righe di una matrice identità; osserviamo che tutte le matrici di questo tipo sono ortogonali.

Definizione 1.8. Si dice che una matrice A è **riducibile** se esiste una matrice di permutazione P tale che

$$PAP^T = \begin{bmatrix} B_{11} & B_{12} \\ 0 & B_{22} \end{bmatrix},$$

dove B_{11} e B_{22} sono matrici quadrate di dimensione k e $n-k$ rispettivamente. Diremo che una matrice è **irriducibile** se non è riducibile.

Osservazione 1.9.1. La riducibilità è una proprietà che risulta computazionalmente vantaggiosa nel calcolo degli autovalori di una matrice.
Infatti il fatto che A sia una matrice riducibile ossia $PAP^T = B$, implica che A e B siano simili essendo la matrice di permutazione P ortogonale; pertanto A e B hanno gli stessi autovalori.
Essendo B una matrice triangolare a blocchi osserviamo che vale la relazione

$$det(B) = det(B_{11}) \cdot det(B_{22}),$$

passando al polinomio caratteristico di B si ha

$$det(B - \lambda I) = det(B_{11} - \lambda I) \cdot det(B_{22} - \lambda I).$$

Da ciò si evince che il problema del calcolo degli autovalori di B e quindi della matrice A di partenza, si decompone in due problemi di dimensione inferiore.

Possiamo ora considerare il

Teorema 1.10 (Terzo teorema di Gershgorin o Teorema di Taussky)**.** *Data A una matrice irriducibile, se un suo autovalore appartiene alla frontiera di S_R allora esso deve appartenere alla frontiera di ciascuno dei cerchi riga R_i.*

Dimostrazione. Consideriamo $(\lambda, \mathbf{x})$ una coppia costituita da un autovalore e dal corrispondente autovettore e supponiamo che λ appartenga alla frontiera dell'unione dei cerchi riga S_R. Nella dimostrazione del Primo teorema di Gershgorin abbiamo visto che, supponendo che i sia l'indice corrispondente alla componente di massimo modulo di $\mathbf{x}$ ossia $|x_i| = \max_{j=1,\dots,n} |x_j|$, è possibile dimostrare che $\lambda \in R_i$ poichéè

$$|\lambda - a_{ii}| \cdot |x_i| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \cdot |x_j| \leq |x_i| \cdot \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|.$$

Poiché per ipotesi $\lambda \in \partial S_R$, necessariamente λ appartiene a ∂R_i e nelle precedenti due maggiorazioni deve valere l'uguale. Quindi deve aversi che $|x_i| = |x_j|$ per tutti i j tali che $a_{ij} \neq 0$.
Essendo A irriducibile esiste almeno un indice k tale che $a_{ik} \neq 0$ e risulta che

$$|x_k| = |x_i| = \max_{j=1,\dots,n} |x_j|.$$

È possibile ripetere il ragionamento per la k-esima riga di A e mostrare quindi che $\lambda \in R_k$ e cioè che $\lambda \in \partial R_k$. Per l'irriducibilità di A il ragionamento può essere ripetuto per tutti gli indici concludendo che

$$\lambda \in \partial R_i, \quad i = 1, \dots, n.$$

□

1.3 Forme canoniche

Definizione 1.9. Una matrice A di dimensione n si dice **diagonalizzabile** se esiste una matrice X non singolare tale che

$$X^{-1}AX = D = diag(\lambda_1, \ldots, \lambda_n). \tag{1.3}$$

Dalla definizione segue quindi che ogni matrice diagonalizzabile è simile ad una matrice diagonale. Quando la matrice X è unitaria cioè tale che $X^* = X^{-1}$, la matrice A si dirà **unitariamente diagonalizzabile**.
Osserviamo che non tutte le matrici sono diagonalizzabili; infatti si ha il seguente

Teorema 1.11. *Una matrice A di dimensione n è diagonalizzabile se e solo se ammette n autovettori indipendenti.*

Dimostrazione. La (1.3) può essere riscritta come

$$AX = XD, \tag{1.4}$$

considerando $X = [x_1 \ldots x_n]$ matrice avente gli autovettori di A in colonna, in maniera estesa la precendente diviene

$$A[x_1 \ldots x_n] = [x_1 \ldots x_n] \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix}$$

ossia

$$A\mathbf{x}_i = \lambda_i \mathbf{x}_i, \quad i = 1, \ldots, n.$$

Se gli autovettori $\{\mathbf{x}_i\}$ di A sono linearmente indipendenti si ha che la matrice X è invertibile e quindi si può moltiplicare la (1.4) per X^{-1} ottenendo la relazione

$$X^{-1}AX = D = diag(\lambda_1, \ldots, \lambda_n)$$

che esprime la diagonizzabilità di A e quindi la tesi.
Il viceversa è immediato. □

La (1.3) che può anche essere riscritta come

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

viene detta **fattorizzazione spettrale** della matrice A. Abbiamo osservato che non sempre esiste, ma nel caso in cui sia disponibile può essere utilizzata

per la risoluzione di un sistema lineare $Ax = b$ anche se tuttavia essa è, in generale, complessa da ottenere.
Un'altra fattorizzazione matriciale è la **decomposizione** (o forma canonica) **di Schur**. Essa risulta molto utile in analisi numerica in quanto ogni matrice A, incluse quelle difettive, può essere fattorizzata come

$$A = UTU^*,$$

dove U è unitaria e T è triangolare superiore. Osserviamo che essendo A e T simili, gli autovalori di A necessariamente appaiono nella diagonale di T.
Possiamo considerare il

Teorema 1.12 (Teorema di Schur)**.** *Per ogni matrice complessa A di dimensione n esiste una matrice complessa unitaria U non univocamente determinata, tale che U^*AU sia triangolare superiore ossia della forma*

$$U^*AU = T = \begin{bmatrix} \lambda_1 & * & \dots & * \\ & \lambda_2 & \ddots & \vdots \\ & & \ddots & * \\ & & & \lambda_n \end{bmatrix},$$

dove i $\lambda_i, i = 1, \dots, n$ sono gli autovalori di A e T non è univocamente determinata.

Dimostrazione. La dimostrazione procede per induzione sulla dimensione n. La tesi è banale per $n = 1$. Procedendo quindi per induzione, supponiamo che il risultato sia vero per $n - 1$ e consideriamo una generica matrice complessa A di dimensione n.
La matrice considerata possiede almeno un autovettore $\mathbf{u}$ associato ad un autovalore λ; senza ledere la generalità possiamo supporre che $\|u\|_2 = 1$. È possibile trovare una matrice V di ordine $n \times (n-1)$ tale che la matrice $n \times n$ $W = [\mathbf{u}, V]$ sia unitaria, inserendo u in un insieme di vettori ortonormali.
Abbiamo dunque che $AW = [\lambda\mathbf{u}, AV]$ e risulta quindi

$$W^*AW = \begin{bmatrix} u^* \\ V* \end{bmatrix}, \qquad [\lambda\mathbf{u}, AV] = \begin{bmatrix} \lambda & \mathbf{u}^*AV \\ 0 & V^*AV \end{bmatrix} \tag{1.5}$$

Utilizzando ora l'ipotesi induttiva sulla matrice $B = V^*AV$ di ordine $(n-1)$, dovrà esistere una matrice $(n-1) \times (n-1)$ unitaria U_1 tale che $U_1^*AU_1 = T_1$ sia triangolare superiore.
Consideriamo ora la matrice

$$\tilde{U}_1 = \begin{pmatrix} 1 & 0 \\ 0 & U_1 \end{pmatrix}$$

e moltiplicando entrambi i membri della 1.5 a sinistra per $\tilde{U}_1^*$ e a destra per $\tilde{U}_1$, otteniamo che la matrice risultante sarà triangolare superiore. Quindi il risultato è valido per la matrice A con $U = \tilde{U}_1 W$ che è una matrice $n \times n$ unitaria. □

Come si evince, il Teorema di Schur afferma che ogni matrice, anche reale, è unitariamente simile ad una matrice triangolare.
Grazie a questo risultato è possibile dimostrare che le matrici Hermitiane possiedono un'importante proprietà come segue dal

Teorema 1.13. *Ogni matrice Hermitiana è unitariamente diagonalizzabile e i suoi autovalori sono reali.*

Dimostrazione. Dal Teorema di Schur segue che esiste una matrice unitaria U tale che $U^*AU = T$; essendo A Hermitiana e quindi coincidente con la sua aggiunta si ha

$$T^* = (U^*AU)^* = U^*AU = T.$$

Dal fatto che coincide con la sua aggiunta, la matrice triangolare T deve essere necessariamente diagonale e i suoi elementi diagonali, autovalori di A, sono reali. □

Osservazione 1.13.1. Una matrice unitariamente diagonalizzabile è tale che i suoi autovettori sono ortonormali e viceversa.
Infatti ponendo

$$U = [\mathbf{v}_1 \dots \mathbf{v}_n],$$

con $A\mathbf{v}_i = \lambda_i \mathbf{v}_i$ essendo U unitaria si ha

$$U^*U = I \iff \mathbf{v}^*{}_i \mathbf{v}_j = \delta_{ij}.$$

Le matrici Hermitiane non sono le uniche ad essere unitariamente diagonalizzabili, ma rientrano in questa classe anche le matrici unitarie e quelle circolanti per citarne qualcuna. In generale le matrici unitariamente diagonalizzabili possono essere caratterizzate sfruttando le matrici normali.

Definizione 1.10. Una matrice A si dice **normale** se commuta con la sua aggiunta, ossia se

$$AA^* = A^*A.$$

Possiamo pertanto considerare il seguente risultato:

Teorema 1.14. *Una matrice è unitariamente diagonalizzabile se e solo se è normale.*

Dimostrazione. La condizione necessaria è immediata, infatti una verifica diretta mostra che ogni matrice unitariamente diagonalizzabile commuta con la sua aggiunta e quindi è normale. Viceversa, sia A una matrice normale e consideriamo $U^*AU = T$ la sua forma canonica di Shur. Allora anche T è normale e, essendo triangolare superiore, esaminando le matrici T^*T e TT^* si evince che essa è necessariamente diagonale. Da ciò segue che A è unitariamente diagonalizzabile. □

Prendiamo ora in considerazione la **forma normale di Jordan**, una più generale fattorizzazione che considera autovalori e autovettori di una matrice.

Teorema 1.15 (Teorema di Jordan)**.** *Per ogni matrice A di ordine n, esiste una matrice non singolare X tale che*

$$X^{-1}AX = J = diag(J_{k_1}(\lambda_1), J_{k_2}(\lambda_2), \ldots, J_{k_l}(\lambda_l)), \tag{1.6}$$

*dove $k_1 + k_2 + \ldots + k_l = n$ e gli elementi $J_{k_i}(\lambda_i)$ sono matrici dette **blocchi di Jordan** della matrice A e sono tali che*

$$J_k(\lambda) = \begin{bmatrix} \lambda & 1 & & \\ & \lambda & \ddots & \\ & & \ddots & 1 \\ & & & \lambda \end{bmatrix} \in \mathbb{R}^{k\times k}$$

se $k > 1$, mentre se $k = 1$ si ha $J_1(\lambda) = \lambda$.

Se la matrice A ha autovalori distinti $\lambda_1, \lambda_2, \ldots, \lambda_n$, il numero dei blocchi di Jordan associati a ciascuno di essi è unico così come la dimensione di ciascun blocco che in tal caso sarà 1 e quindi la matrice A risulta diagonalizzabile. Osserviamo che invece non è unica la matrice non singolare X e l'ordine dei blocchi che compongono la matrice di Jordan J.
Nel caso in cui la matrice A abbia un autovalore difettivo , almeno uno dei blocchi di Jordan corrispondenti ad esso ha dimensione superiore a 1 mentre si ha che tutti i blocchi di Jordan associati ad un autovalore non difettivo coincidono con l'autovalore stesso. Da questo fatto si evince che una matrice è pertanto diagonalizzabile se e solo se è non difettiva.
La fattorizzazione 1.6 è detta appunto **forma normale di Jordan** della matrice A. Dal teorema segue che essa esiste per ogni matrice A ma in generale risulta complicato calcolarla e quindi non è molto utilizzata nelle applicazioni, al contrario della forma canonica di Shur. Risulta però molto importante da un punto di vista teorico perché rappresenta un legame generale tra un matrice e i suoi autovalori e autovettori e quindi è uno strumento fondamentale per lo studio della matrici difettive.

Un'altra fattorizzazione matriciale che vedremo essere legata alla fattorizzazione spettrale, è la **decomposizione ai valori singolari** (SVD). Data una matrice $A \in \mathbb{C}^{m\times n}$, non necessariamente a rango pieno, una decomposizione ai valori singolari di A è una fattorizzazione del tipo

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

dove $U \in \mathbb{C}^{m\times m}$ e $V \in \mathbb{C}^{n\times n}$ sono unitarie mentre $\Sigma \in \mathbb{C}^{m\times n}$ è diagonale con elementi σ_j non negativi e tali che $\sigma_1 \geq \sigma_2 \geq \cdots \sigma_p \geq 0$ con $p = \min(m, n)$. Una dimostrazione del fatto che ogni matrice $A \in \mathbb{C}^{m\times n}$ può essere fattorizzata in tal modo, si trova in [3].
Le colonne di U e V sono detti *vettori singolari*, mentre gli elementi σ_i di Σ sono detti *valori singolari* di A.
La SVD ha dei fondamentali collegamenti con degli argomenti di Algebra Lineare. Per i successivi teoremi che consideriamo supponiamo che A sia una matrice rettangolare di dimensione $m \times n$, $p = \min(m, n)$ e sia $r \leq p$ il numero dei valori singolari non nulli di A.

Teorema 1.16. *Il rango della matrice A è pari a r, numero dei suoi valori singolari non nulli.*

Dimostrazione. Considerando che il rango di una matrice diagonale è pari al numero delle sue entrate non nulle, osserviamo che nella decomposizione $A = U\Sigma V^*$ le matrici unitarie U e V sono a rango pieno. Di conseguenza si ha che

$$\operatorname{rank}(A) = \operatorname{rank}(\Sigma) = r$$

□

Teorema 1.17. *I valori singolari non nulli di A sono le radici quadrate degli autovalori non nulli di A^*A o di AA^* (che sono matrici aventi gli stessi autovalori non nulli).*

Dimostrazione. Considerando la SVD di A calcoliamo

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

da cui segue che A^*A e $\Sigma^*\Sigma$ sono simili e quindi hanno gli stessi n autovalori. Considerando che gli autovalori della matrice diagonale $\Sigma^*\Sigma$ sono $\sigma_1^2, \sigma_2^2, \ldots, \sigma_p^2$, con $n - p$ autovalori nulli aggiuntivi nel caso in cui $n > p$, otteniamo la tesi. Effettuando un simile calcolo si dimostra lo stesso per gli m autovalori di AA^*. □

Nel caso particolare di una matrice Hermitiana, si ha il seguente

Teorema 1.18. *Se $A = A^*$, allora i valori singolari di A sono i valori assoluti degli autovalori di A.*

Ricordando che la norma Euclidea di una matrice A è data da

$$\|A\|_2 = \sqrt{(A^*A)},$$

possiamo considerare il

Teorema 1.19. *La norma-2 di una matrice A è*

$$\|A\|_2 = \sigma_1.$$

Dimostrazione. Senza perdita di generalità possiamo supporre che A sia di ordine $m \times n$, con $m \geq n$ e sia $A = U\Sigma V^*$ la sua decomposizione ai valori singolari. Considerando che la norma-2 è invariante per trasformazioni ortogonali/unitarie si ha

$$\|A\|_2 = \|\Sigma\|_2.$$

Poiché la norma Euclidea di una matrice diagonale è pari al valore assoluto del suo elemento maggiore, si ha

$$\|\Sigma\|_2 = \max_{i=1,\ldots,p} |\sigma_i| = \sigma_1,$$

da cui segue la tesi. □

1.4 Pseudospettri

Sullo studio di problemi agli autovalori si basa l'analisi di modelli lineari in vari campi della matematica e ingegneria. Si osserva che questa analisi è sicuramente soddisfacente nel caso particolare di matrici che possiedono una base ortonormale di autovettori e, recentemente, si è avuta una maggiore consapevolezza del fatto che in caso di matrici "non-normali", cioé che non possiedono una base ortogonale di autovettori, si deve procede ad un'analisi con maggiore cautela. Osserviamo che la proprietà di non-normalità può essere legata ad un comportamento transitorio della matrice, che differisce completamente da quello asintotico che si evince dallo studio degli autovalori. Allo scopo di studiare la non-normalità e analizzarne gli effetti, sono stati suggeriti vari strumenti tra cui classici strumenti di teoria delle matrici come il rango, l'angolo tra sottospazi invarianti e il numero di condizionamento del problema agli autovalori considerato. Un ulteriore strumento che si è dimostrato utile in vari casi, è quello degli **pseudospettri**. Esistono 4 definizioni

equivalenti di pseudospettro di una matrice A che supponiamo sia quadrata di ordine n.
Presa quindi $A \in \mathbb{C}^{n\times n}$, ricordando la definizione di spettro di A come insieme dei suoi autovalori osserviamo che gli autovalori di A soddisfano appunto la definizione

$$\sigma(A) = \{z \in \mathbb{C} : \det(zI - A) = 0\},$$

o equivalentemente lo spettro di A può essere quindi definito come l' insieme dei punti del piano complesso in cui la matrice $(zI - A)$ è singolare e quindi in cui $(A - zI)^{-1}$ è indefinita.
Se z è un autovalore di A per convenzione si pone la $\|(zI - A)^{-1}\|_2$ pari ad infinito. La prima definizione di pseudospettro nasce dall'ipotesi che tale norma possa essere invece finita, seppur molto grande.

Definizione 1.11 (Prima definizione di pseudospettro)**.** Si definisce *epsilon-pseudospettro* l'insieme

$$\Lambda_\varepsilon(A) = \{z \in \mathbb{C} : \|(A - zI)^{-1}\| \geq \varepsilon^{-1}\}.$$

L'epsilon-speudospettro può essere anche definito in termini di autovalori di matrici perturbate, pertanto si ottiene la seconda definizione equivalente:

Definizione 1.12 (Seconda definizione di pseudospettro)**.** L'epsilon-speudospettro di una matrice A può essere definito come l'insieme

$$\Lambda_\varepsilon(A) = \{z \in \mathbb{C} : z \in \sigma(A + E)\},$$

dove E è una qualche matrice di perturbazione tale che $\|E\| \leq \varepsilon$.

Se z appartiene all' epsilon-pseudospettro,allora viene detto *epsilon-pseudo-autovalore* di A. Come nel caso degli autovalori, anche a ciascun pseudo-autovalore è possibile associare, in modo non univoco in generale, uno pseudo-autovettore. Considerando queste quantità si arriva alla terza definizione di pseudo-spettro.

Definizione 1.13 (Terza definizione di pseudospettro)**.**

$$\Lambda_\varepsilon(A) = \{z \in \mathbb{C} : \|(A - zI)\mathbf{v}\| \leq \varepsilon\},$$

per qualche vettore $\mathbf{v} \in \mathbb{C}^n$ di norma unitaria.

L'equivalenza di queste tre definizioni è espressa dal seguente

Teorema 1.20. *Supponiamo che $\|\cdot\|$ sia una norma matriciale indotta da una norma vettoriale; allora le seguenti tre definizioni di pseudospettro sono equivalenti*

1. $\Lambda_\varepsilon(A) = \{z \in \mathbb{C} : \|(A - zI)^{-1}\| \geq \varepsilon^{-1}\}$;
2. $\Lambda_\varepsilon(A) = \{z \in \mathbb{C} : z \in \sigma(A + E) \quad \textit{per qualche } E \textit{ tale che} \quad \|E\| \leq \varepsilon\}$;
3. $\Lambda_\varepsilon(A) = \{z \in \mathbb{C} : \exists \mathbf{v} \in \mathbb{C}^n \quad \textit{con} \quad \|\mathbf{v}\| = 1 \mid \|(A - zI)\mathbf{v}\| \leq \varepsilon\}$
   Se $\|\cdot\|$ è la norma euclidea, si ha una quarta definizione di pseudospettro equivalente
4. $\Lambda_\varepsilon(A) = \{z \in \mathbb{C} : \sigma_{\min}(A - zI) \leq \varepsilon\}$,

dove $\sigma_{\min}(\cdot)$ denota il più piccolo valore singolare della matrice considerata.

Dimostrazione. Osserviamo innanzitutto che se $z \in \sigma(A)$, allora z soddisfa tutte le caratteristiche di appartenenza all'epsilon-pseudospettro $\Lambda_\varepsilon(A)$. Supponiamo quindi che $z \notin \sigma(A)$.
(1) $\Longrightarrow$ (3): supponendo quindi che $\|(A - zI)^{-1}\| \geq \varepsilon^{-1}$, esisterà un vettore $\mathbf{u} \in \mathbb{C}^n$ tale che

$$\|(A - zI)^{-1}\mathbf{u}\|/\|\mathbf{u}\| = \|(A - zI)^{-1}\|.$$

Definiamo $\hat{\mathbf{v}} = (A - zI)^{-1}\mathbf{u}$, in modo che si abbia

$$\varepsilon^{-1} \leq \frac{\|(A - zI)^{-1}\mathbf{u}\|}{\|\mathbf{u}\|} = \frac{\|\hat{\mathbf{v}}\|}{\|(A - zI)\hat{\mathbf{v}}\|}.$$

Resta così determinato un vettore $\mathbf{v} = \hat{\mathbf{v}}/\|\hat{\mathbf{v}}\|$ di norma unitaria che soddisfa la diseguaglianza $\|(A - zI)\mathbf{v}\| \leq \varepsilon$.
(3) $\Longrightarrow$ (2): supponiamo che esista un vettore $\mathbf{v} \in \mathbb{C}^n$ tale che $\|\mathbf{v}\| = 1$. Sia $\mathbf{u} \in \mathbb{C}^n$ un vettore unitario che soddisfi la relazione

$$(A - zI)\mathbf{v} = \hat{\varepsilon}\mathbf{u}, \quad \hat{\varepsilon} \leq \varepsilon.$$

Si può dimostrare, utilizzando la teorie delle norme duali, l'esistenza di un vettore $\mathbf{w}$ di norma unitaria e tale che $\mathbf{w}^*\mathbf{v} = 1$.
Considerando quindi un vettore $\mathbf{w}$ di questo tipo possiamo scrivere $z\mathbf{v} = A\mathbf{v} - \hat{\varepsilon}\mathbf{u}\mathbf{w}^*\mathbf{v} = (A - \hat{\varepsilon}\mathbf{u}\mathbf{w}^*)\mathbf{v}$ che implica che l'appartenenza di z a $\sigma(A+E)$ dove la matrice di perturbazione $E = \hat{\varepsilon}\mathbf{u}\mathbf{w}^*$ è tale che $\|E\| \leq \varepsilon$.
(2) $\Longrightarrow$ (1): per ipotesi si ha che $z \in \sigma(A + E)$, per una qualche matrice E tale che $\|E\| \leq \varepsilon$. Esiste allora un vettore unitario $\mathbf{v} \in \mathbb{C}^n$ tale che $(A + E)\mathbf{v} = z\mathbf{v}$. Riscrivendo questa relazione come $\mathbf{v} = (A - zI)^{-1}E\mathbf{v}$, passando alle norme si ha

$$1 = \|\mathbf{v}\| = \|(A - zI)^{-1}E\mathbf{v}\| \leq \|(A - zI)^{-1}\|\|E\| \leq \varepsilon\|(A - zI)^{-1}\|,$$

che implica che $\|(A - zI)^{-1}\| \geq \varepsilon^{-1}$. Non ci resta ora che provare che la (1) implica la (4) per concludere la dimostrazione.
(1) $\Longrightarrow$ (4): se la norma matriciale considerata è la norma-2, l'implicazione segue dalla caratterizzazione della norma euclidea dell'inversa di una matrice come il suo più piccolo valore singolare (vedi Teorema (1.19)). □

1.4.1 Metodi di calcolo degli pseudospettri

Sebbene gli pseudospettri vennero definiti in via teorica intorno al 1975, il loro primo calcolo per una matrice 3×3 venne pubblicato nel 1987. A partire poi dal 1993, calcoli di pseudospettri di matrici di dimensione 64 vennero applicati allo studio del comportamento diegli operatori differenziali della fluidomeccanica [9]. Dal 2001, si trovano esempi di calcolo di pseudospettri per matrici di dimensioni dell'ordine di centinaia di migliaia [16]. In tempi recenti le tecniche per il calcolo degli pseudospettri sono migliorate. Tutti gli sforzi nel rendere le tecniche maggiormente avnzate sono diretti al calcolo dello pseudospettro in norma-2 che, come si è visto, può essere caratterizzato in termini del più piccolo valore singolare di $A - zI$

$$\Lambda_\varepsilon(A) = \{z \in \mathbb{C} : \sigma_{\min}(A - zI) \leq \varepsilon\}.$$

La maggior parte degli algoritmi oer il calcolo degli pseudospettri rientra nelle due seguenti classi:

- **Algoritmi griglia** che considerano una regione nel piano complesso, la ricoprono con un reticolo e calcolano il più piccolo valore singolare di $zI - A$ per ogni z nella gliglia considerata;
- **Algoritmi di Path-following** che consistono nel determinare un punto sulla frontiera dello pseudospettro che si vuole calcolare e partendo da esso una curva nel piano complesso sulla quale il minimo valore singolare di $zI - A$ si mantiene costante [7].

Gli algorirmi della prima tipologia, che rappresentano i metodi standard, affrontano diversi aspetti. Occorre innanzitutto scegliere una regione del piano complesso da studiare, e successivamente considerare un gran numero di punti nel reticolo per delineare adeguatamente il profilo dello pseudospettro considerato. A tal proposito sono stati sviluppati nuovi approcci e tecniche per eliminare in maniera veloce, alcuni dei punti del reticolo che risultano poco interessanti.
Anche gli algoritmi di Path-following hanno delle limitazioni. Innanzitutto si deve trovare un punto sulla frontiera dello pseudospettro che si vuole

determinare; se lo pseudospettro è sconnesso, si deve determinare ogni componente connessa e sono richieste molte esecuzioni nel caso in cui si voglia trovare lo pseudospettro per parecchi differenti valori di ε.
Entrambe le tipologie di algoritmi richiedono la valutazione del minimo valore singolare di $zI - A$ al variare di z. Sono stati fatti considerevoli progressi nel rendere questo processo il più efficiente possibile. Quando A è una matrice densa, Lui ha evidenziato il vantaggio di ridurre inizialmente A in forma di Hessenberg o in forma triangolare prima di calcolare σ_{min} di $zI - A$ [6]. Una tecnica di accelerazione, molto importante in alcune applicazioni, consiste nel proiettare ortogonalmente A su un opportuno sottospazio in modo da ridurre la dimensione del problema nel caso in cui A sia molto grande e si voglia determinare solo lo pseudospettro in una porzione del piano complesso; si vedano [6], [9], [16], [12].
Tra i software sviluppati il più noto è **EigTool**, un pacchetto gratuito di MATLAB per il calcolo degli pseudospettri di matrici dense e sparse. Fornisce inoltre un'interfaccia grafica per la routine `eigs` di MATLAB per problemi agli autovalori di grandi dimensioni. EigTool fu sviluppato negli anni 1999 - 2002 da Thomas G. Wright presso l'Università di Oxford sotto la guida di Nick Trefethen. Nelle Figure 1.1-1.2-1.3 vengono mostrati alcuni esempi di pseudospettri che possono trovarsi in [4], ottenuti utilizzando il pacchetto EigTool. I valori sulla barra dei colori sono riportati secondo una scala logaritmica in base 10(epsilon).

1.5 Sistemi lineari e problemi ai minimi quadrati

1.5.1 Sistemi lineari e condizionamento

Consideriamo un sistema lineare di n equazioni in n incognite che in notazione matriciale possiamo esprimere come

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

dove $A = (a_{ij})_{i,j=1}^{n}$ è la matrice dei coefficienti, $\mathbf{b} = (b_1, \ldots, b_n)^T$ è il vettore dei termini noti e $\mathbf{x} = (x_1, \ldots, x_n)^T$ la soluzione.
Da un fondamentale teorema di Algebra Lineare segue che un sistema lineare è ben posto ossia ammette una ed una sola soluzione che dipende con continuità dai dati, se e solo se è verificata una delle seguenti proprietà equivalenti

- $\det(A) \neq 0$;

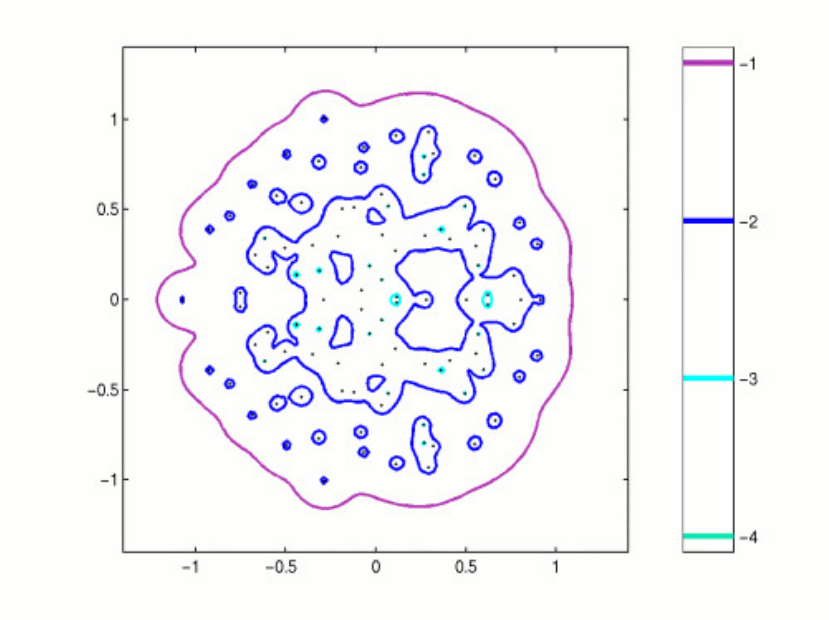

Figura 1.1: *Pseudospettro di una matrice random di dimensione N=200, le cui entrate sono state ottenute da una distribuzione normale con media 0 e varianza 1/N. I punti neri rappresentano gli autovalori. Questo esempio fu illustrato per la prima volta in [13].*

- A ha rango pieno n;
- il sistema omogeneo $A\mathbf{x} = 0$ ammette solo la soluzione banale $\mathbf{x} = 0$.

I sistemi lineari possono essere risolti tramite i *metodi diretti* oppure tramite i *metodi iterativi*; in ogni caso occorre considerare il *condizionamento* del problema che fornisce una misura quantitativa di come la soluzione venga influenzata da una perturbazione dei dati.
Richiamiamo la seguente

Definizione 1.14. Sia δy una perturbazione dei dati y di un problema e sia δx la corrispondente perturbazione sulla soluzione x del problema considerato. Considerata $\|\cdot\|$ una qualsiasi norma vettoriale si ha che il **numero di condizionamento assoluto** $K = K(y)$ è definito dalla relazione

$$\|\delta x\| \leq K\|\delta y\|,$$

mentre il **numero di condizionamento relativo** $k = k(y)$ verifica la disuguaglianze

$$\frac{\|\delta x\|}{\|x\|} \leq k\frac{\|\delta y\|}{\|y\|}.$$

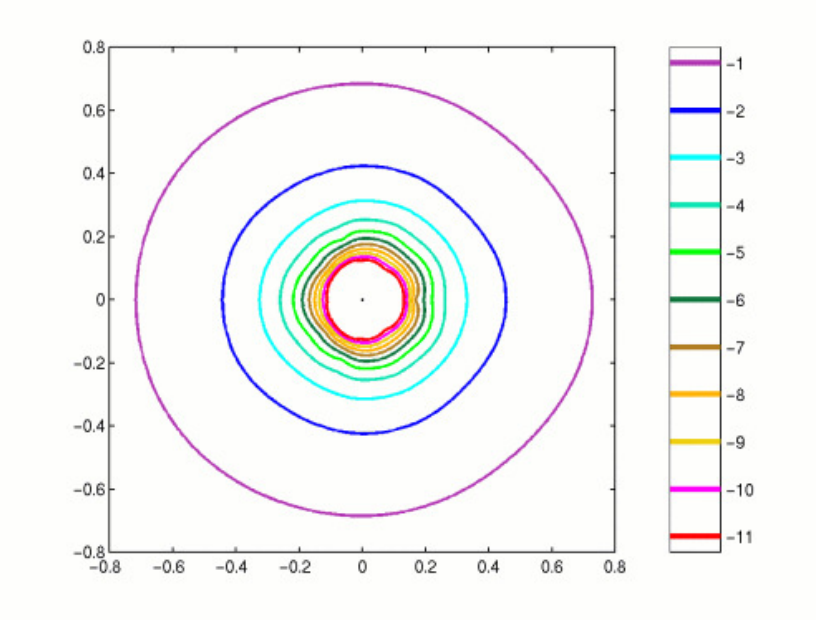

Figura 1.2: *Pseudospettro di una una matrice random triangolare inferiore con entrate reali, di dimensione N=100.*

Il condizionamento di un problema misura, quindi, quanto un errore sui dati influenzi i risultati.
Considerando A matrice dei coefficienti di un sistema lineare di n equazioni in n incognite consideriamo la seguente

Definizione 1.15. Si definisce **numero di condizionamento** di una matrice A in relazione alla risoluzione di un sistema lineare $A\mathbf{x} = \mathbf{b}$, la quantità

$$k(A) = \|A\|\|A^{-1}\|. \tag{1.7}$$

Si vedrà successivamente che il numero di condizionamento di una matrice rettangolare, è legato alla sua SVD.
Osserviamo che il numero di condizionamento dipende dalla norma matriciale adottata e, in particolare, si utilizza un pedice nel caso in cui si voglia specificarla come in $k_2(A)$ o $k_\infty(A)$.

1.5.2 Problemi ai minimi quadrati

Sinora abbiamo considerato sistemi lineari di n equazioni in n incognite, ma nelle applicazioni si trovano frequentemente sistemi lineari con un numero di equazioni differente dal numero delle incognite ossia sistemi $A\mathbf{x} = \mathbf{b}$ dove A

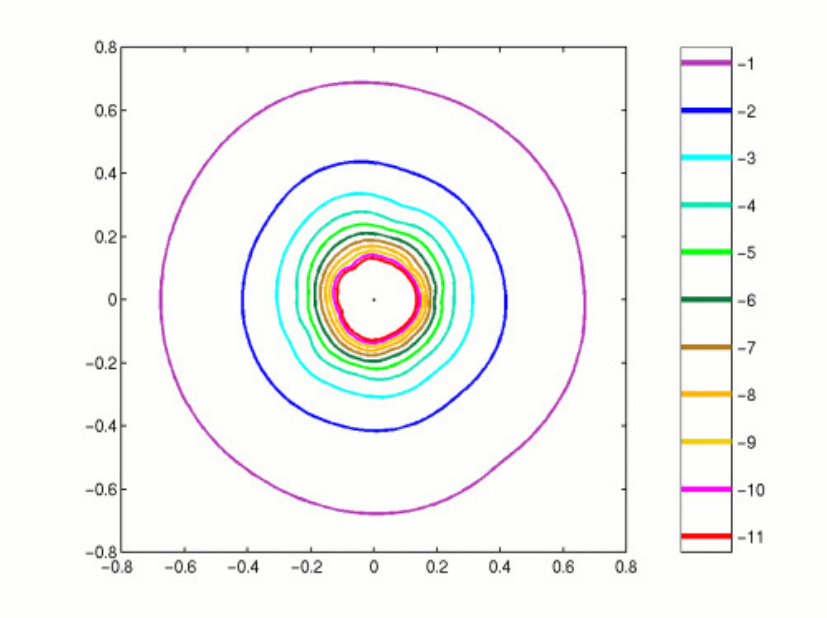

Figura 1.3: *Pseudospettro di una una matrice random triangolare inferiore con entrate complesse, di dimensione N=100.*

è una matrice rettangolare $m \times n$, $\mathbf{b} \in \mathbb{R}^m$ e $\mathbf{x} \in \mathbb{R}^n$. Supponendo che la matrice dei coefficienti sia a rango pieno, si possono avere due casi

- $m > n$ sistema **sovradeterminato** in cui si hanno più equazioni che incognite per cui potrebbe non esistere soluzione;

- $m < n$ sistema **sottodeterminato** in cui vi sono più incognite che equazioni e quindi potrebbe ammettere infinite soluzioni.

In entrambi i casi si hanno dei problemi mal posti, cioè non aventi un'unica soluzione che dipende con continuità dai dati, e quindi essi non ammettono una soluzione in senso classico. Prendiamo in considerazione solo il caso $m > n$ osservando che nel caso di un sistema sottodeterminato si ha una situazione maggiormente complicata dovuta al fatto che potrebbero esserci infinite soluzioni.
L'idea alla base dei minimi quadrati è quella di "risolvere" un sistema sovradeterminato minimizzando la norma-2 del residuo; quindi un **problema ai minimi quadrati** in generale può essere formulato come un problema di minimizzazione del tipo

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

dove $A \in \mathbb{R}^{m\times n}$ con $m > n$ e $\mathbf{b} \in \mathbb{R}^m$. Nel caso in cui il minimo precedente risulta essere nullo, si ha che il sistema ammette una soluzione in senso classico. In caso contrario si otterrà la *soluzione nel senso dei minimi quadrati* del sistema lineare sovradeterminato $A\mathbf{x} = \mathbf{b}$.

Metodo delle equazioni normali Un modo classico per la risoluzione di un problema ai minimi quadrati è quello che opera risolvendo il *sistema normale* associato al problema sovradeterminato considerato, dato da

$$A^T A\mathbf{x} = A^T\mathbf{b} \tag{1.8}$$

Se A è a rango pieno, la matrice $A^T A \in \mathbb{R}^{n\times n}$ è invertibile e quindi il sistema normale ammette un'unica soluzione nel senso dei minimi quadrati

$$\mathbf{x} = \left(A^T A\right)^{-1} A^T\mathbf{b}.$$

Nel caso in cui A abbia rango inferiore a A, si ha che $A^T A$ è singolare ma il sistema resta consistente e tra gli infiniti vettori che lo soddisfano si sceglie come soluzione $\mathbf{x}$ quella dotata di minima norma euclidea.
La matrice

$$A^\dagger = \left(A^T A\right)^{-1} A^T \in \mathbb{R}^{n\times m}$$

viene detta **pseudo-inversa** di A o inversa di Moore-Penrose; risulta essere un'inversa sinistra di A ma non inversa destra.
Nel caso in cui A è a rango pieno la matrice $A^T A$ sarà simmetrica e definita positiva e quindi per risolvere il sistema normale (1.8) si può far ricorso alla *fattorizzazione di Cholesky* di tale matrice. Questo metodo, per risolvere il sistema normale considerato, costruisce la fattorizzazione $A^T A = R^T R$ con R matrice triangolare superiore di ordine n riducendo la (1.8) al sistema

$$R^T R\mathbf{x} = A^T\mathbf{b}.$$

Pertanto la soluzione $\mathbf{x}$ può essere calcolata risolvendo in successione i due sistemi triangolari inferiore e superiore rispettivamente

$$\begin{cases} R^T\mathbf{y} = A^T\mathbf{b} \\ R\mathbf{x} = \mathbf{y} \end{cases}$$

Un altro procedimento utilizzato per la risoluzione di un problema ai minimi quadrati utilizza la fattorizzazione QR della matrice dei coefficienti del sistema.

Problemi ai minimi quadrati tramite la SVD I problemi ai minimi quadrati possono essere risolti tramite la decomposizione ai valori singolari. Consideriamo il sistema reale sovradeterminato $A\mathbf{x} = \mathbf{b}$ con matrice dei coefficienti di ordine $m \times n$, con $m > n$, a rango pieno n. Consideriamo la SVD ridotta di A data da

$$A = \hat{U}\hat{\Sigma}V^T$$

dove $\hat{U} \in \mathbb{R}^{m\times n}$ è ortogonale, Σ è una matrice diagonale di ordine n con entrate reali positive e $V \in \mathbb{R}^{n\times n}$ è ortogonale. Vale il seguente

Teorema 1.21. *Sia $A \in \mathbb{R}^{m\times n}$ a rango pieno n e sia $A = \hat{U}\hat{\Sigma}V^T$ la sua decomposizione ai valori singolari ridotta. Si ha allora che il problema ai minimi quadrati $\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|_2$ ammette un'unica soluzione data da*

$$\mathbf{x} = V\hat{\Sigma}^{-1}\hat{U}^T\mathbf{b} = \sum_{i=1}^{n} \frac{u_i^T\mathbf{b}}{\sigma_i} v_i.$$

Questo metodo per la risoluzione di un problema ai minimi quadrati tramite la SVD può essere schematizzato come illustrato nell' Algoritmo 1.

Algoritmo 1 Minimi quadrati con SVD

1: **Input:** $A \in \mathbb{R}^{m\times n}$ $(m > n)$, $\mathbf{b} \in \mathbb{R}^m$.
2: calcolare la SVD ridotta $A = \hat{U}\hat{\Sigma}V^T$
3: calcolare il vettore $\hat{U}^T\mathbf{b}$
4: risolvere il sistema diagonale $\hat{\Sigma}\mathbf{y} = \hat{U}^T\mathbf{b}$
5: calcolare $\mathbf{x} = V\mathbf{y}$
6: **Output:** soluzione $\mathbf{x}$ nel senso dei minimi quadrati
7: del sistema sovradeterminato $A\mathbf{x} = \mathbf{b}$

Numero di condizionamento Abbiamo già considerato la definizione di numero di condizionamento di una matrice quadrata relativamente alla risoluzione di un sistema lineare. Nel caso di una matrice rettangolare $A \in \mathbb{R}^{m\times n}$ $(m \geq n)$ a rango pieno n, il suo numero di condizionamento è definito in termini della psedo-inversa ossia $\mathrm{k}(A) = \|A\|\|A^\dagger\|$. Quindi poiché si tratta del numero di condizionamento di una matrice in relazione ad un sistema lineare sovradeterminato, la norma considerata sarà la norma-2 e si ha che il numero di condizionamento di A può essere espresso utilizzando la SVD e in particolare i valori singolari di A come

$$\mathrm{k}(A) = \frac{\sigma_1}{\sigma_n}.$$

Nel caso in cui A sia quadrata e non singolare la precedente definizione si riduce alla (1.7).
Il numero di condizionamento in norma-2 di un problema ai minimi quadrati $\min_{\mathbf{x}\in\mathbb{R}^n} \|A\mathbf{x} - \mathbf{b}\|_2$ che misura quantitativamente quanto la soluzione del problema venga influenzata da una perturbazione della matrice $A \in \mathbb{R}^{m\times n}$, è dato da

$$k_{LS} = k(A) + \frac{k(A)^2 \tan\theta}{\eta}$$

dove $k(A) = \|A\|\|A^\dagger\|$ è il numero di condizionamento della matrice rettangolare A, $\eta = \frac{\|A\|\|\mathbf{x}\|}{\|A\mathbf{x}\|} \in [1, k(A)]$ e

$$\theta = \arccos\frac{\|A\mathbf{x}\|}{\mathbf{b}} \in [0, \pi/2]$$

è l'angolo formato tra il vettore $\mathbf{b}$ e la sua proiezione $A\mathbf{x}$ sul sottospazio rank(A) [14].

Capitolo 2

Algoritmi numerici per il calcolo degli autovalori

Nonostante autovalori e autovettori abbiano delle semplici definizioni, il miglior modo per calcolarli non è ovvio.
Il loro calcolo è un problema non lineare e non viene affrontato attraverso il metodo classico, ossia determinando gli autovalori come zeri del polinomio caratteristico in quanto il problema di determinare le radici di un polinomio può essere difficile da risolvere. Osserviamo che così come un problema agli autovalori può essere ridotto ad un problema di determinare le radici di un polinomio, viceversa il calcolo delle radici di equazioni algebriche può basarsi sulla risoluzione di un problema agli autovalori. Infatti dato un polinomio di grado n

$$p_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$$

è sempre possibile renderlo monico, dividendolo per il coefficiente a_n del termine di grado massimo. Si ottiene quindi il polinomio con coefficiente di grado massimo unitario e avente gli stessi zeri di $p_n(x)$

$$\tilde{p}_n(x) = x^n - b_{n-1}x^{n-1} - \cdots - b_1 x - b_0$$

dove $b_i = -\frac{a_i}{a_n}$, $i = 0, 1, \ldots, n-1$. Al polinomio monico $\tilde{p}_n(x)$ è possibile associare la matrice

$$A_n = \begin{bmatrix} b_{n-1} & b_{n-2} & \cdots & b_1 & b_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & \ddots & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 0 \end{bmatrix}$$

detta **matrice compagna** per la quale vale la relazione

$$\tilde{p}_n(\lambda) = \det(A_n - \lambda I).$$

Dalla precedente relazione si evince che $\tilde{p}_n(\lambda)$ è il polinomio caratteristico di A_n e quindi gli autovalori della matrice compagna coincidono con gli zeri di $\tilde{p}_n(x)$ e quindi con quelli del polinomio iniziale $p_n(x)$. Questo evidenzia quindi la difficoltà che insorge nell'utilizzo del metodo standard per la ricerca degli autovalori dato che è ben noto che non esiste alcuna formula che esprima le radici di un polinomio arbitrario, noti i suoi coefficienti. Questo risultato, che rientra nella *Teoria di Galois*, esprime il fatto che non esistano formule per le radici di un'equazione algebrica di grado superiore al quarto. In particolare si ha il seguente teorema, dimostrato nel 1824 dal matematico norvegese Abel (1802-1829).

Teorema 2.1. *Per ogni $m \geq 5$, esiste un polinomio $p_m(z)$ di grado m con coefficienti razionali avente una radice reale r che non può essere ottenuta per radicali ossia che non può essere espressa in funzione dei coefficienti di $p_m(z)$ tramite una formula in cui si utilizzino esclusivamente le quattro operazioni elementari e l'estrazione di radice.*

Questo teorema implica che, anche se si lavorasse in aritmetica esatta, non potrebbe esistere alcun algoritmo in grado di determinare le radici di un polinomio arbitrario in un numero finito di passi e quindi la stessa difficoltà si riscontra nel calcolo degli autovalori di una matrice.
Pertanto sono stati sviluppati dei metodi iterativi che consentono di calcolare autovalori e autovettori di una matrice in sostituzione al metodo standard.
Si è già osservato che, in generale, il calcolo numerico degli autovalori e autovettori è un problema delicato sia perché computazionalmente oneroso sia perché in alcuni casi può essere estremamente instabile conducendo ad una forte propagazione degli errori e quindi ad approssimazioni inaccurate.
In alcuni problemi non è necessario conoscere esattamente tutti gli autovalori, ma ci si può accontentare di una risposta parziale come localizzare gli autovalori ossia determinare una regione del piano complesso che contenga gli autovalori, determinare degli insiemi che contengano ognuno un singolo autovalore (separazione degli autovalori), calcolare gli autovalori estremali cioè quelli di modulo massimo e minimo, migliorare la stima di un autovalore oppure dato un autovalore determinare il corrispondente autovettore.
Consideriamo ora uno dei più importanti risultati ai fini dello studio della stabilità del calcolo degli autovalori, valido solo nel caso di matrici diagonalizzabili.

Teorema 2.2 (Teorema di Bauer-Fike)**.** *Supponiamo che $A \in \mathbb{C}^{n\times n}$ sia diagonalizzabile, ossia che esista una matrice non singolare X tale che*

$$X^{-1}AX = D = \mathrm{diag}(\lambda_1, \ldots, \lambda_n).$$

Fissata arbitrariamente una matrice di perturbazione $E \in \mathbb{C}^{m\times m}$, supponiamo che μ sia un autovalore della matrice perturbata $A + E$. Allora risulta che

$$min_{\lambda\in\sigma(A)}|\lambda - \mu| \leq k_2(X) \cdot \|E\|_2,$$

dove con $\sigma(A)$ denotiamo lo spettro della matrice A e con k_2 il numero di condizionamento in norma 2.

Il Teorema di Bauer-Fike afferma che, se la matrice A di partenza è diagonalizzabile, il condizionamento $k_2(X)$ della matrice degli autovettori rappresenta il numero di condizionamento del problema agli autovalori che misura quanto una perturbazione dei dati possa influenzare la soluzione. Osserviamo che tale teorema mostra che ogni autovalore della matrice perturbata $A + E$ giace in almeno uno degli m dischi circolari del piano complesso di raggio $k_2(X) \cdot \|E\|_2$, centrati negli autovalori di A.

Dimostrazione. Se μ è un autovalore di A, la dimostrazione è immediata. Supponiamo pertanto che $\mu \notin \sigma(A)$. Essendo μ per ipotesi un autovalore della matrice perturbata $A + E$, la matrice $A + E - \mu I$ è singolare e di conseguenza lo sarà anche

$$X^{-1}(A + E - \mu I)X = D + X^{-1}EX - \mu I.$$

Dovrà quindi esistere un vettore $\mathbf{z} \neq 0$ tale che

$$(D + X^{-1}EX - \mu I)\mathbf{z} = 0$$

da cui segue che, essendo $D - \mu I$ invertibile, la precedente possa scriversi come

$$\big(I - (D - \mu I)^{-1}X^{-1}EX\big)\mathbf{z} = 0.$$

dalla precedente otteniamo che il vettore $\mathbf{z}$ verifica l'uguaglianza

$$\mathbf{z} = -(D - \mu I)^{-1}X^{-1}EX\mathbf{z}$$

e, passando alla norma Euclidea, la diseguaglianza

$$\|\mathbf{z}\|_2 \leq \|(D - \mu I)^{-1}\|_2 \cdot k_2(X) \cdot \|E\|_2 \cdot \|\mathbf{z}\|_2.$$

Sfruttando il fatto che se una matrice B è Hermitiana risulta che $\|B\|_2 = \rho(B)$ otteniamo

$$\|(D-\mu I)^{-1}\|_2 = |\lambda max((D-\mu I)^{-1})| = \frac{1}{|\lambda_{min}(D-\mu I)|}$$
$$= \frac{1}{\min_{\lambda\in\sigma(A)}|\lambda-\mu|},$$

da cui segue la diseguaglianza

$$\min_{\lambda\in\sigma(A)}|\lambda-\mu| = \frac{1}{\|(D-\mu I)^{-1}\|_2} \leq k_2(X)\cdot\|E\|_2$$

che implica la tesi. □

Osserviamo che il problema agli autovalori risulta ben condizionato nel caso di matrici unitariamente diagonalizzabili e in particolare lo è per matrici Hermitiane. Infatti se A è unitariamente diagonalizzabile, si ha $Q^TAQ = D$ con Q matrice ortogonale e quindi tale che $k_2(Q) = 1$ si ha

$$\min_{\lambda\in\sigma(A)}|\lambda-\mu| \leq \|E\|_2.$$

Consideriamo ora dei classici algoritmi per la risoluzione di un problema agli autovalori.

2.1 Il metodo delle potenze

Il metodo delle potenze è un metodo iterativo per la risoluzione di un problema agli autovalori; può essere utilizzato per calcolare l'autovalore di massimo modulo di una matrice di grandi dimensioni, per migliorare la stima di un autovalore e anche per determinare l'autovettore associato ad un dato autovalore.
Questo metodo consente di approssimare l'autovalore λ_1 di modulo massimo di una matrice A, che supponiamo sia di ordine n, e l'autovettore corrispondente ad esso. Esso converge quando sono verificate le seguenti ipotesi:

1. A è diagonalizzabile;

2. il vettore iniziale $x^{(0)}$ ha una componente non nulla lungo l'autovettore $\mathbf{v}_1$ corrispondente all'autovalore di massimo modulo λ_1;

3. l'autovalore di modulo massimo è separato dagli altri, ossia risulta

$$|\lambda_1| > |\lambda_i|, \quad i = 1, \ldots, n.$$

Consideriamo quindi una matrice A di ordine n diagonalizzabile e per convenzione supponiamo che i suoi autovalori siano ordinati in modo decrescente relativamente al loro modulo, cioè

$$|\lambda_1| \geq |\lambda_2| \geq \ldots |\lambda_n|,$$

supponiamo cioè che l'autovalore di modulo massimo λ_1 abbia molteplicità algebrica unitaria e che non esistano altri autovalori con lo stesso modulo. Il metodo delle potenze consiste nel fissare un vettore iniziale $\mathbf{x}^{(0)}$ e calcolare iterativamente una successione di vettori il cui termine k-esimo è

$$\mathbf{x}^{(k)} = A^k \mathbf{x}^{(0)};$$

tale successione converge all'autovettore principale della matrice A ossia all'autovettore $\mathbf{v}_1$ associato all'autovalore di massimo modulo. Esaminiamo il comportamento per $k \to \infty$ della successione $\mathbf{x}^{(k)}$ e quindi studiamo la convergenza del metodo sotto le ipotesi fatte. Per l'ipotesi 1 la matrice A è diagonalizzabile e quindi per la condizione necessaria e sufficiente espressa dal teorema (1.11) si ha che esiste una base di autovettori e quindi il vettore iniziale $\mathbf{x}^{(0)}$ può essere espresso come loro combinazione lineare

$$\mathbf{x}_0 = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i, \tag{2.1}$$

dove i $\mathbf{v}_i$ sono gli autovettori di A cioè $A\mathbf{v}_i = \lambda_i \mathbf{v}_i$. Considerando il k-esimo termine della successione degli iterati, tramite la (2.1) otteniamo

$$\mathbf{x}_k = A^k \mathbf{x}_0 = \sum_{i=1}^{n} \alpha_i A^k \mathbf{v}_i = \sum_{i=1}^{n} \alpha_i \lambda_i^k \mathbf{v}_i.$$

Sfruttando la seconda ipotesi si ha che $\alpha_1 \neq 0$, pertanto la precedente uguaglianza diviene

$$\mathbf{x}_k = \alpha_1 \lambda_1^k \mathbf{v}_1 + \sum_{i=2}^{n} \alpha_i \lambda_i^k \mathbf{v}_i = \alpha_1 \lambda_1^k \left(\mathbf{v}_1 + \sum_{i=2}^{n} \frac{\alpha_i}{\alpha_1} \left(\frac{\lambda_i}{\lambda_1} \right)^k \mathbf{v}_i \right). \tag{2.2}$$

Grazie alla terza ipotesi, il fattore $\left(\frac{\lambda_i}{\lambda_1} \right)^k$ converge a zero per $k \to \infty$ e quindi otteniamo che il vettore $\mathbf{x}_k$ tende ad essere parallelo all'autovettore

principale $\mathbf{v}_1$. In realtà, poiché un autovettore è univocamente determinato a meno di una costante moltiplicativa, la successione degli iterati converge all'autovettore $\mathbf{v}_1$ associato a λ_1.
Allo scopo di aumentare la stabilità del calcolo e diminuire il costo computazionale di ogni iterazione, potremmo valutare $\mathbf{x}^k$ a partire dal vettore ottenuto nell'iterazione precedente tramite la relazione

$$\mathbf{x}^{(k)} = A\mathbf{x}^{(k-1)}.$$

Nonostante questo procedimento sia utile da un punto di vista implementativo, può presentare un problema numerico. Infatti dalla (2.2), osserviamo che la norma di $\mathbf{x}^{(k)}$ converge a zero quando $|\lambda_1| < 1$ e si va incontro ad un fenomeno di *underflow* mentre diverge all'infinito quando $|\lambda_1| > 1$ comportando in tal caso un fenomeno di *overflow*.
Per evitare questi problemi numerici, è necessario normalizzare il vettore $\mathbf{x}^{(k)}$ ad ogni iterazione per evitare che la sua norma cresca o descresca eccessivamente considerando la schema

$$\mathbf{x}^{(k)} = A\mathbf{q}^{(k-1)},$$

$$\mathbf{q}^{(k)} = \frac{\mathbf{x}^{(k)}}{\|\mathbf{x}^{(k)}\|}.$$

Quindi ad ogni iterazione valutiamo il vettore $\mathbf{x}^{(k)}$ a partire dal vettore calcolato all'iterazione precedente normalizzato.
L'autovalore di massimo modulo λ_1 può essere approssimato ad ogni passo tramite il quoziente di Rayleigh, ovvero come

$$\lambda_1^{(k)} = \frac{(\mathbf{q}^{(k)})^T A\mathbf{q}^{(k)}}{(\mathbf{q}^{(k)})^T\mathbf{q}^{(k)}}.$$

Occorre poi fissare un numero massimo di iterazioni N ed una certa tolleranza $\tau > 0$ per considerare un criterio di stop del tipo

$$|\lambda_1^{(k)} - \lambda_1^{(k-1)}| < \tau \cdot |\lambda_1^{(k)}| \quad \text{oppure} \quad k > N.$$

Il metodo delle potenze, considerando la normalizzazione in norma 2, può essere quindi schematizzato come nell'Algortimo 2.
Il metodo delle potenze ha comunque un utilizzo limitato per alcune ragioni. Innanzitutto tramite esso è possibile approssimare solo l'autovalore di massimo modulo di una matrice e l'autovettore ad esso corrispondente, in secondo luogo la convergenza risulta lineare e dipende dal fatto che l'autovalore di

Algoritmo 2 Metodo delle potenze con normalizzazione in norma-2

Input: matrice diagonalizzabile $A \in \mathbb{R}^{n\times n}$, vettore iniziale $\mathbf{x}^{(0)} \in \mathbb{R}^n$,
tolleranza τ e numero massimo di iterazioni N.
$k = 0$, $\mathbf{q}^{(0)} = \frac{\mathbf{x}^{(0)}}{\|\mathbf{x}^{(0)}\|_2}$, $\lambda^{(0)} = 0$
while $|\lambda^{(k)} - \lambda^{(k-1)}| < \tau \cdot ||\lambda^{(k)}|$ and $k \leq N$
  $k = k + 1$
  $\mathbf{x}^{(k)} = A\mathbf{q}^{(k-1)}$
  $\mathbf{q}^{(k)} = \frac{\mathbf{x}_{(k)}}{\|\mathbf{x}_{(k)}\|_2}$
  $\lambda^{(k)} = (\mathbf{q}^{(k)})^T A\mathbf{q}^{(k)}$
end while
Output: autovalore di modulo massimo λ_1 e
il corrispondente autovettore normalizzato q_1 .

modulo massimo sia significativamente più grande degli altri. Infatti in presenza di due autovalori dello stesso ordine di grandezza, la convergenza del metodo potrebbe essere molto lenta.
Osserviamo che il metodo delle potenze può essere modificato in modo da approssimare l'autovalore di modulo minimo di una matrice non singolare; in questo caso parliamo di **metodo delle potenze inverse**.

2.2 Metodo delle potenze inverse

Per ogni $\mu \in \mathbb{R}$ che non sia un autovalore della matrice A, gli autovettori di $(A-\mu I)^{-1}$ coincidono con gli autovettori di A e gli autovalori relativi ad essi sono $(\lambda_j - \mu)^{-1}$ dove i λ_j sono gli autovalori di A per $j = 1, \ldots, n$. Questo suggerisce il fatto che, se μ è vicino ad un autovalore λ_J di A allora $(\lambda_J-\mu)^{-1}$ risulterebbe più grande di $(\lambda_j-\mu)^{-1}$ per tutti i $j \neq J$. Quindi se applicassimo il metodo delle potenze alla matrice $(A-\mu I)^{-1}$,sotto le opportune ipotesi, il processo convergerebbe all'autovettore $\mathbf{v}_J$ corrispondente all'autovalore λ_J.
Su questa idea si basa il metodo delle potenze inverse che quindi applicato alla matrice $(A - \mu I)$, consente di determinare l'autovalore della matrice A più prossimo ad un valore prefissato μ.
Abbiamo osservato in precedenza che il metodo delle potenze inverse è una modifica del metodo delle potenze, con lo scopo di determinare l'autovalore di minimo modulo di una matrice che sia non singolare.

Infatti considerando A invertibile si ha che se $\sigma(A) = \{\lambda_1, \ldots, \lambda_n\}$, gli autovalori dell'inversa di A sono i reciproci degli autovalori di A ossia $\sigma(A^{-1}) = \{\lambda_1^{-1}, \ldots, \lambda_n^{-1}\}$; pertanto è possibile l'applicare l'algoritmo alla matrice A^{-1} per approssimare il suo autovalore di massimo modulo λ_n^{-1}. Analogamente a quanto detto in precedenza, il metodo converge sotto le ipotesi che A sia diagonalizzabile, il vettore iniziale $\mathbf{x}^{(0)}$ abbia componente non nulla rispetto all'autovettore $\mathbf{v}_n$ corrispondente all'autovalore di minimo modulo λ_n e risulti

$$\frac{1}{|\lambda_n|} > \frac{1}{|\lambda_i|}, \quad i = 1, \ldots, n-1.$$

Poiché calcolare l'inversa di una matrice ha un elevato costo computazionale, la k-esima iterazione non verrà calcolata dalla formula $\mathbf{x}^{(k)} = A^{-1}\mathbf{x}^{(k-1)}$ ma risolvendo il sistema lineare

$$A\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)}.$$

Nel risolvere questo sistema lineare per ridurre il costo computazionale è conveniente fattorizzare la matrice A all'inizio del processo in modo da risolvere ad ogni passo due sistemi triangolari di più semplice risoluzione.
Se le ipotesi di convergenza sono verificate si ha che per $k \to \infty$ $\lambda^{(k)} \to \lambda_n^{-1}$.
Il metodo delle potenze inverse può essere schematizzato come indicato nell'Algoritmo 3.
Il metodo delle potenze, come già osservato, risulta efficiente per approssimare gli autovalori estremali di matrici di grandi dimensioni. Tuttavia questo procedimento calcola un autovalore per volta, mentre i metodi di Arnoldi e di Lanczos che risulta essere una variante del precedente per matrici simmetriche, determinano gli autovalori estremali in numero superiore a 1 simultaneamente. Nel prossimo capitolo prenderemo in esame i suddetti metodi che rientrano tra i metodi di proiezione in sottospazi di Krylov.

2.3 L'algoritmo QR per il calcolo degli autovalori

2.3.1 L'algoritmo QR classico

L'algoritmo QR è un metodo iterativo che permette di approssimare tutti gli autovalori di una matrice di dimensioni non elevate. Se la matrice non ha una forma particolare, il suo costo computazionale è elevato per questo motivo viene solitamente implementato con alcune tecniche che consentono di accelerarne la convergenza oltre che di ridurne il costo computazionale ad ogni iterazione.

Algoritmo 3 Metodo delle potenze inverse

Input: matrice non singolare e diagonalizzabile $A \in \mathbb{R}^{n\times n}$,
vettore iniziale $\mathbf{x}^{(0)} \in \mathbb{R}^n$, tolleranza τ e N.
$k = 0$, $\mathbf{q}^{(0)} = \frac{\mathbf{x}^{(0)}}{\|\mathbf{x}^{(0)}\|_2}$, $\lambda^{(0)} = 0$, fattorizzazione di A $A = LU$
(o $A = QR$, $PA = LU$,$A = R^T R$ etc.)
while $|\lambda^{(k)} - \lambda^{(k-1)}| < \tau \cdot |\lambda^{(k)}|$ and $k \leq N$
  $k = k + 1$
  risolvere $L\mathbf{y} = \mathbf{q}^{(k-1)}$
  risolvere $U\mathbf{x}^{(k)} = \mathbf{y}$
  $\mathbf{q}^{(k)} = \frac{\mathbf{x}_{(k)}}{\|\mathbf{x}_{(k)}\|_2}$
  $\lambda^{(k)} = (\mathbf{q}^{(k)})^T A\mathbf{q}^{(k)}$
end while
Output: autovalore di modulo minimo λ_n e
il corrispondente autovettore normalizzato q_n .

Infatti viene generalmente applicato dopo aver trasformato la matrice, mediante trasformazioni di Householder o di Givens, in una forma tridiagonale o di Hessenberg.
Il metodo QR, descritto da Francis nel 1961, utilizza la fattorizzazione QR di una matrice. A tal proposito ricordiamo che ogni matrice A di ordine $m \times n$ può essere fattorizzata nella forma

$$A = QR$$

dove Q è una matrice $m \times m$ ortogonale quindi tale che $Q^TQ = QQ^T = I$ mentre R è triangolare superiore con le stesse dimensioni di A.
L'algoritmo consiste nel, data una matrice quadrata A di ordine n, porre per inizializzare il metodo $A_0 = A$; nella generica iterazione si calcola la fattorizzazione QR della matrice $A_k \Rightarrow A_k = Q_kR_k$, usando ad esempio il metodo di Householder, e si calcola poi la matrice al passo successivo rimoltiplicando i fattori in ordine inverso $\Rightarrow A_{k+1} = R_kQ_k$.
Possiamo innanzitutto osservare che tutte le matrici della successione risultano unitariamente simili, infatti si ha

$$A_{k+1} = R_kQ_k = Q_k^{\mathrm{T}}Q_kR_kQ_k = Q_k^{\mathrm{T}}A_kQ_k$$

pertanto avranno tutte gli stessi autovalori (coincidenti appunto con quelli della matrice iniziale A).

Sotto l'ipotesi che gli autovalori di A, matrice arbitraria, siano distinti in modulo, la successione di matrici $\{A_k\}$ converge ad una matrice triangolare superiore

$$T = \begin{bmatrix} \lambda_1 & * & \dots & * \\ & \lambda_2 & \ddots & \vdots \\ & & \ddots & * \\ & & & \lambda_n \end{bmatrix}$$

che conterrà gli autovalori della matrice di partenza A, essendo simile a questa, nella diagonale principale non necessariamente ordinati per modulo decrescente.
Nel caso in cui la matrice di partenza sia Hermitiana, essendo $A_{k+1}^{\mathrm{T}} = Q_k^{\mathrm{T}} A_k^{\mathrm{T}} Q_k$, saranno simmetriche tutte le matrici della successione e quindi necessariamente la matrice T (a cui esse convergono) coinciderà con una matrice diagonale

$$T = D = \mathrm{diag}(\lambda_1, \dots, \lambda_n).$$

Osserviamo che la fattorizzazione QR fornisce anche un'approssimazione della forma canonica di Schur della matrice A; infatti si ha

$$A_{k+1} = Q_k^{\mathrm{T}} A_k Q_k = Q_k^{\mathrm{T}} Q_{k-1}^{\mathrm{T}} A_{k-1} Q_{k-1} Q_k = \dots$$
$$= Q_k^{\mathrm{T}} \dots Q_0^{\mathrm{T}} A Q_0 \dots Q_k = U_k^{\mathrm{T}} A U_k,$$

in cui abbiamo indicato con $U_k = Q_0 \dots Q_k$; pertanto risulta

$$\lim_{k\to\infty} A_k = T = U^{\mathrm{T}} A U, \quad \text{con} \quad U = \prod_{i=1}^{\infty} Q_i.$$

Nel caso in cui A sia Hermitiana otteniamo la sua fattorizzazione spettrale

$$A = U D U^T,$$

in cui U è la matrice le cui colonne sono gli autovettori di A e D è una matrice diagonale i cui elementi sono gli autovalori di A.
Abbiamo affermato in precedenza che il metodo risulta essere convergente sotto l'ipotesi che gli autovalori siano tutti distinti in modulo

$$|\lambda_1| > |\lambda_2| > \dots > |\lambda_n|;$$

tuttavia anche se questa ipotesi non fosse soddisfatta il metodo può essere modificato in modo tale da approssimare comunque gli autovalori.
Infatti supponendo che la matrice abbia due autovalori con stesso modulo $\lambda_1 = \overline{\lambda_2}$, in tal caso sarà presente un blocco diagonale 2×2 avente l'elemento sottodiagonale non nullo ma gli autovalori tenderanno a λ_1 e λ_2. In

un caso del genere la successione della matrici A_k non convergerà ad una matrice triangolare superiore ma sarà sufficiente calcolare separatamente gli autovalori dei blocchi diagonali che non si stabilizzano applicando l'algoritmo classico, ovvero calcolando gli zeri del polinomio caratteristico corrispondente che sarà di grado 2.
Osserviamo che il metodo QR applicato ad una matrice di ordine n ha ad ogni passo un costo computazionale dell'ordine di n^3 operazioni moltiplicative che sono quelle utilizzate per calcolare la fattorizzazione $A_k = Q_k R_k$ e per moltiplicare la matrice triangolare R_k per le matrici elementari della fattorizzazione.

2.4 Passaggio in forma di Hessenberg

L'algoritmo QR è il metodo più diffuso per il calcolo degli autovalori di matrici di dimensione non troppo elevata; per diminuire il numero di iterazioni e quindi la complessità computazionale, viene spesso applicato insieme a delle tecniche che ne accelerano appunto la convergenza.
Una tecnica spesso utilizzata consiste nel trasformare la matrice A, di cui vogliamo calcolare gli autovalori tramite l'algoritmo QR, in una matrice in forma di Hessenberg cioè avente la seguente struttura

$$\tilde{A} = \begin{bmatrix} * & \dots & \dots & * \\ * & \ddots & & \vdots \\ & \ddots & \ddots & \vdots \\ & & * & * \end{bmatrix}$$

in cui quindi $\tilde{a_{ij}} = 0$ per $i > j + 1$.
Questa trasformazione viene effettuata una sola volta perché il metodo QR, applicato a matrici in forma di Hessenberg produce matrici in forma di Hessenberg. Questa tecnica è molto utilizzata proprio per il fatto che la forma di Hessenberg è quindi invariante per trasformazioni QR, ovvero se la matrice A a cui si deve applicare l'algoritmo QR è in forma di Hessenberg avranno questa struttura anche tutte le matrici della successione generata dal metodo. Questo accelera la convergenza per il fatto che l' algoritmo in tal caso dovrà azzerare solo gli $n-1$ elementi sottodiagonali (per ottenere la matrice triangolare superiore T). Per azzerare i suddetti $n-1$ elementi sottodiagonali sono sufficienti $n-1$ trasformazioni di Givens e pertanto nel calcolare la fattorizzazione QR di una matrice in forma di Hessenberg si ha una notevole diminuzione del costo computazionale di ogni iterazione.
Per effettuare il passaggio in forma di Hessenberg sono sufficienti $n-2$ trasformazioni di Householder applicate contemporaneamente alla destra e alla

sinistra della matrice A così da conservare la similitudine. Prima di considerare la riduzione in forma di Hessenberg tramite trasformazioni di Householder richiamiamo la seguente

Definizione 2.1. (Matrici elementari di Householder)
Una matrice elementare di Householder reale H è una matrice della forma

$$H = I - 2\mathbf{w}\mathbf{w}^T$$

dove $\mathbf{w} \in \mathbb{R}^n$ ha norma euclidea unitaria; la matrice $2\mathbf{w}\mathbf{w}^T$ avente ordine n è di rango 1.

Ogni matrice di Householder gode delle seguenti proprietà

- è simmetrica cioè $H{=}H^T$. Infatti:

$$H^T = (I - 2\mathbf{w}\mathbf{w}^T)^T = I - 2((\mathbf{w}^T)^T\mathbf{w}^T) = I - 2\mathbf{w}\mathbf{w}^T = H$$

- è ortogonale cioè $H^TH = HH^{\mathrm{T}} = I$ e quindi $H^{\mathrm{T}} = H^{-1}$. Infatti sfruttando la simmetria di H si ha:

$$\begin{aligned} H^{\mathrm{T}}H = HH &= (I - 2\mathbf{w}\mathbf{w}^T)(I - 2\mathbf{w}\mathbf{w}^T) = I - 4\mathbf{w}\mathbf{w}^T + 4\mathbf{w}\mathbf{w}^T\mathbf{w}\mathbf{w}^T \\ &= I - 4\mathbf{w}\mathbf{w}^T + 4\mathbf{w}\mathbf{w}^T. \end{aligned}$$

Le matrici elementari di Householder hanno anche l'importante proprietà di annullare determinati elementi di una matrice o di un vettore e in questo consiste una trasformazione di Householder.
Vediamo quindi ora come può essere determinata la riduzione in forma di Hessenberg della matrice A di partenza e quindi la fattorizzazione

$$U^TAU = \tilde{A},$$

dove $\tilde{A}$ è una matrice di Hessenberg superiore e U è il prodotto di $n-2$ matrici elementari di Householder H_i per $i = 1, \ldots, n-2$. Osserviamo che il ruolo della generica matrice H_i è quello di annullare gli elementi della i-esima colonna che si trovano al di sotto della sottodiagonale.
Al primo passo denotiamo con $\tilde{\mathbf{a}}_1$ il vettore contenente gli $n-1$ elementi della prima colonna di $A^{(1)} = A$ aventi indice compreso tra 2 e n. Sia ora $\tilde{H}_1$ la matrice elementare di Householder di ordine $n-1$ tale che $\tilde{H}_1\tilde{\mathbf{a}}_1 = k_1\mathbf{e}_1$ e indichiamo con H_1 la matrice ortogonale che si ottiene orlando, con una

riga e una colonna della matrice identità, $\tilde{H}_1$ fino a raggiungere dimensione n come

$$H_1 = \begin{bmatrix} 1 & 0 & \dots & 0 \\ 0 & & & \\ \vdots & & \tilde{H}_1 & \\ 0 & & & \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{0}^{\mathrm{T}} \\ \mathbf{0} & \tilde{H}_1 \end{bmatrix}.$$

La matrice che si ottiene moltiplicando la matrice H_1 a destra e a sinistra della matrice A ossia applicando una trasformazione di Householder contemporaneamente a destra e sinistra sarà del tipo

$$A^{(2)} := H_1 A^{(1)} H_1 = \begin{bmatrix} * & * & \dots & * \\ k_1 & * & \dots & * \\ 0 & * & \dots & * \\ \vdots & \vdots & & \vdots \\ 0 & * & \dots & * \end{bmatrix}$$

e inoltre è simile ad A quindi possiede gli stessi autovalori.
Procedendo allo stesso modo indicando con $\tilde{H}_i$ la matrice elementare di Householder di ordine $n-i$ tale che $\tilde{H}_i \tilde{\mathbf{a}}_i = k_i \mathbf{e}_1$ e con H_i la matrice di dimensione n ottenuta orlando $\tilde{H}_i$, otteniamo

$$A^{(i+1)} := H_i A^{(i)} H_i = \begin{bmatrix} * & * & \dots & \dots & \dots & * \\ k_1 & * & \dots & * & & \\ & \ddots & \ddots & & & * \\ & & & * & \dots & * \\ & & & \vdots & & \vdots \\ & & & * & \dots & * \end{bmatrix}.$$

Iterando il procedimento , la matrice finale

$$\tilde{A} = A^{(n-1)} = H_{n-2} H_{n-1} \dots H_2 H_1 A H_1 H_2 \dots H_{n-1} H_{n-2} = U^T A U$$

dove $U = \prod_{i=1}^{n-2} H_i$, risulta in forma di Hessenberg e simile alla matrice iniziale A; ad essa verrà applicato l'algoritmo QR. Nell'implementazione, dopo aver ridotto in forma di Hessenberg e aver assegnato una certa tolleranza e il numero massimo di iterazioni, utilizziamo l'algoritmo QR per fattorizzare la matrice $\tilde{A}$ e come criterio di stop utilizziamo il seguente

$$|a_{i+1,1}| \leqslant \tau, i = 1, \dots, n-1,$$

verificando che gli elementi sottodiagonali siano inferiori in modulo alla tolleranza assegnata.

Il metodo QR applicato ad una matrice in forma di Hessenberg superiore ha, ad ogni passo, un costo computazionale di $2n^2$ operazioni moltiplicative. Nel caso in cui la matrice A sia Hermitiana, la procedura di riduzione in forma di Hessenberg tramite trasformazioni di Householder appena descritta comporta la riduzione di A ad una forma tridiagonale in assenza di errori di arrotondamento. Infatti se A è Hermitiana la matrice H^*AH sarà anch'essa Hermitiana oltre che essere in forma di Hessenberg e ogni matrice di Hessenberg che sia anche Hermitiana è necessariamente tridiagonale. Inoltre anche tutte le matrici $\{A_k\}$ della successione generata dal metodo, saranno Hermitiane e quindi tridiagonali. In questo caso il costo computazionale del metodo sarà, ad ogni passo, lineare in n.

2.4.1 L'algoritmo QR con shift

La velocità di convergenza del metodo QR dipende dai rapporti $|\frac{\lambda_i}{\lambda_j}|$ per $i > j$, e quindi per l'ipotesi che gli autovalori siano distinti in modulo dal numero

$$\max_{1 \leqslant i \leqslant n} \left| \frac{\lambda_{i+1}}{\lambda_i} \right|.$$

Se questo rapporto è prossimo a 1, la convergenza può essere lenta. In tal caso per accelerare la convergenza del metodo si utilizza una tecnica di traslazione dello spettro degli autovalori della matrice A di partenza, detta *di shift*. Supponendo che μ sia un numero che approssima un certo autovalore λ di A , ponendo $A_0 = A - \mu I$ possiamo schematizzare l'algoritmo QR con shift come

Algoritmo 4 Metodo QR con shift

1: **Input:** $A \in \mathbb{R}^{n \times n}$, μ
2: $A_0 = A - \mu I$
3: **for** $k = 0, 1, \ldots$ fino alla convergenza
4: $\quad A_k - \mu I = Q_k R_k$
5: $\quad A_{k+1} = R_k Q_k + \mu I$
6: **end for**
7: **Output:** matrice triangolare T avente nella diagonale
8: $\quad$ gli autovalori di A

Tenendo presente che gli autovalori della matrice $A - \mu I$ sono $\lambda_i - \mu$ e che la velocità di convergenza è legata al rapporto tra gli autovalori, è possibile sce-

gliere un parametro μ in modo tale da accelerare la convergenza del metodo QR con shift.
È conveniente scegliere per μ un valore che approssima l'autovalore λ_n; questo può essere ottenuto ad esempio applicando il metodo QR inizialmente senza shift per un certo numero l di iterazioni e poi scegliendo $\mu = a_{nn}^{(l)}$ per le successive iterazioni con shift. Poiché il parametro μ può essere anche modificato ad ogni iterazione, risulta maggiormente conveniente scegliere

$$\mu_k = a_{nn}^{(k)}, \quad k = 1, 2, \ldots.$$

Supponiamo ora che la matrice A sia reale e simmetrica di ordine n con autovalori reali λ_j; vogliamo considerare la convergenza delle matrici A_k della successione generata dal metodo, ad una forma diagonale.
Prima di procedere all'iterazione del metodo QR con shift, la matrice A può essere ridotta ad una forma tridiagonale da utilizzare quindi alla base del metodo. Poi al posto di A_k, ad ogni passo viene fattorizzata una matrice "shiftata" $A_k - \mu_k I$ dove μ_k è la stima di un qualche autovalore di A. Quando possibile e in particolare nel caso in cui venga determinato un autovalore, il problema viene ridotto considerando delle sottomatrici di A_k.
L'algoritmo QR con shift in questo caso può essere schematizzato come

Algoritmo 5 Metodo QR con shift

Input: $A \in \mathbb{R}^{n\times n}$ simmetrica, $\mu_0 = a_{nn}^{(0)}$, τ
$(Q_0)^T A_0 Q_0 = A$ con A_0 tridiagonalizzazione di A
for $k = 0, 1, \ldots$ fino alla convergenza
  $A_k - \mu_k I = Q_k R_k$
  $A_{k+1} = R_k Q_k + \mu_k I$
  if $a_{j,j+1}^{(k)} < \tau$
    $a_{j,j+1} = a_{j+1,j} = 0$ per ottenere

$$A_k = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}$$

  end if
  applicare algoritmo QR a A_1 e A_2
end for
Output: matrice triangolare T avente nella diagonale
    gli autovalori di A

Capitolo 3

Metodi di proiezione in sottospazi di Krylov

I metodi di proiezione in sottospazi di Krylov possono essere utilizzati sia per risolvere un sistema lineare di n equazioni in n incognite $A\mathbf{x} = \mathbf{b}$ sia per approssimare gli autovalori di A come vedremo in seguito.

I *metodi iterativi* per la risoluzione di un sistema lineare si contrappongono a quelli *diretti* che in generale risultano essere computazionalmente costosi. I metodi iterativi per risolvere $A\mathbf{x} = \mathbf{b}$ generano, a partire da un vettore iniziale $\mathbf{x}^{(0)}$, una successione di vettori $\mathbf{x}^{(k)}$ con $k = 0, 1, \ldots$, che converge alla soluzione $\mathbf{x}$ del sistema sotto opportune ipotesi. I metodi diretti valutano la soluzione in un numero finito di passi e gli errori presenti nei risultati discendono dall'aritmetica finita e dalla presenza di errori sui dati iniziali. Nei metodi iterativi invece, oltre agli errori sperimentali e di arrotondamento, si aggiungono anche gli errori di troncamento derivanti dal fatto che la soluzione cercata è un limite e quindi deve essere approssimata troncando la successione per un valore sufficientemente grande dell'indice k. Quindi in tali metodi si presenta il problema della determinazione di un opportuno criterio di arresto. Tuttavia i metodi iterativi risultano maggiormente convenienti rispetto a quelli diretti per matrici di grandi dimensioni specialmente nel caso in cui esse siano sparse o strutturate.

Infatti i metodi iterativi preservano la struttura della matrice sfruttandone quindi l'eventuale sparsità, mentre quelli diretti operano modificando la matrice dei coefficienti del sistema alterandone spesso la struttura e aumentando il numero degli elementi non nulli.

Tra i metodi iterativi rientrano il metodo di Jacobi, di Gauss-Seidel, il metodo del gradiente, il metodo del gradiente coniugato (CG) e i metodi di proiezione in sottospazi di Krylov alcuni dei quali vengono esaminati in questa tesi sia

per la risoluzione di un sistema lineare sia per l 'approssimazione di autovalori e valori singolari di una matrice.
Alla base dei metodi iterativi vi è comunque l'idea di sostituire il sistema lineare o il problema dato, con uno di più semplice risoluzione.

3.1 Il metodo del gradiente e del gradiente coniugato

In questa sezione viene esaminato il metodo del gradiente e in particolare la sua variante CG (conjugate gradient) che può essere vista come un metodo di proiezione in sottospazi di Krylov come mostreremo in seguito.
I metodi del gradiente e del gradiente coniugato sono applicabili alla risoluzione di un sistema lineare $A\mathbf{x} = \mathbf{b}$ nel caso in cui la matrice A dei coefficienti sia simmetrica e definita positiva cioè tale che $\mathbf{x}^T A\mathbf{x} > 0, \forall \mathbf{x} \in \mathbb{C}^n$.
Consideriamo per entrambi i metodi la seguente forma quadratica

$$\varphi(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b}$$

funzione che assume valore minimo nel punto in cui si annulla il suo gradiente

$$\nabla\varphi(\mathbf{x} = \frac{1}{2}(A + A^T)\mathbf{x} - \mathbf{b} = A\mathbf{x} - \mathbf{b} \tag{3.1}$$

che si ottiene sfruttando le formule

$$\nabla(\mathbf{b}^T\mathbf{x}) = \mathbf{b}, \quad \nabla(\mathbf{x}^T A\mathbf{x}) = 2A\mathbf{x}.$$

Si evince quindi che il problema di minimizzare la forma quadratica $\varphi(\mathbf{x})$ è equivalente alla risoluzione del problema $A\mathbf{x} = \mathbf{b}$. Questa soluzione può essere calcolata quindi minimizzando $\varphi(\mathbf{x})$ tramite un metodo iterativo non stazionario del tipo

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{d}^{(k)}$$

partendo da un vettore iniziale $\mathbf{x}^{(0)}$, lungo le direzioni di decrescita $\mathbf{d}^{(k)}$ e con passi di lunghezza α_k che sono dei *parametri di rilassamento* introdotti per accelerare la convergenza del metodo iterativo.
I metodi del gradiente e del gradiente coniugato si distinguono per la particolare scelta delle direzioni di discesa, mentre per quanto riguarda la lunghezza del passo α_k essa si ottiene determinando il minimo di $\varphi(\mathbf{x}^{(k+1)})$ rispetto alla variazione di $\alpha = \alpha_k$ e sarà

$$\alpha_k = \frac{\left(\mathbf{d}^{(k)}\right)^T \mathbf{r}^{(k)}}{\left(\mathbf{d}^{(k)}\right)^T A\mathbf{d}^{(k)}}, \tag{3.2}$$

dove con $\mathbf{r}^{(k)}$ denotiamo il residuo al passo k ossia $\mathbf{r}^{(k)} = \mathbf{b} - A\mathbf{x}^{(k)}$. Nel *metodo del gradiente* viene scelta la direzione $\mathbf{d}^{(k)}$ come quella di massima decrescita ossia come quella opposta alla direzione del gradiente della funzione φ nel punto $\mathbf{x}^{(k)}$; tale direzione coincide col vettore residuo al passo k in quanto dalla 3.1 si ottiene

$$\nabla\varphi(\mathbf{x}^{(k)}) = A\mathbf{x}^{(k)} - \mathbf{b} = -\mathbf{r}^{(k)}$$

Quindi scegliendo come direzione di decrescita il vettore residuo, l'iterazione assume la forma

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{r}^{(k)}$$

dove il valore della lunghezza del passo α_k si ottiene ponendo $\mathbf{d}^{(k)} = \mathbf{r}^{(k)}$ nella 3.2.
Osserviamo che per ridurre la complessità computazionale possiamo calcolare il residuo al passo $k+1$ sfruttando quello calcolato al passo k come

$$\mathbf{r}^{(k+1)} = \mathbf{b} - A\mathbf{x}^{(k+1)} = \mathbf{b} - A\mathbf{x}^{(k)} - \alpha_k A\mathbf{r}^{(k)} = \mathbf{r}^{(k)} - \alpha_k A\mathbf{r}^{(k)}.$$

Il metodo del gradiente può quindi essere schematizzato come mostrato nell'Algoritmo 6.

Algoritmo 6 Metodo del gradiente

Input: $A \in \mathbb{R}^{n\times n}$ simmetrica e definita positiva, $\mathbf{b} \in \mathbb{R}^n$,
$\mathbf{x}^{(0)}$ vettore iniziale
$\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
$k = 0$
for $k = 0, 1, \ldots$ fino alla convergenza
  $\mathbf{s} = A\mathbf{r}^{(k)}$
  $\alpha_k = \left(\mathbf{r}^{(k)}\right)^T \mathbf{r}^{(k)} / \left(\mathbf{r}^{(k)}\right)^T \mathbf{s}$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{r}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathbf{s}$
  $k = k + 1$
end for

Poiché nella pratica il metodo del gradiente è caratterizzato da una convergenza piuttosto lenta, è necessario applicarlo utilizzando un opportuno *precondizionatore* allo scopo di accelerare appunto la convergenza. Si ottiene così il **metodo del gradiente precondizionato** che si basa sull'utilizzo di

una matrice di precondizionamento P che approssimi A e che risulti essere facilmente invertibile da un punto di vista computazionale. La generica iterazione del metodo è della forma

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k P^{-1}\mathbf{r}^{(k)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{z}^{(k)},$$

dove si evince che ad ogni passo si deve risolvere il sistema $P\mathbf{z}^{(k)} = \mathbf{r}^{(k)}$ e tener conto del cambiamento della direzione di discesa (che sarà il residuo precondizionato) nel calcolo del passo α_k. Lo schema del metodo è illustrato nel seguente algoritmo.

Algoritmo 7 Metodo del gradiente precondizionato

Input: $A \in \mathbb{R}^{n\times n}$ simmetrica e definita positiva, $\mathbf{b} \in \mathbb{R}^n$,
  $\mathbf{x}^{(0)}$ vettore iniziale, $\mathbf{z}^{(0)}$, P
$\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
risolvere $P\mathbf{z}^{(0)} = \mathbf{r}^{(0)}$
$k = 0$
for $k = 0, 1, \ldots$ fino alla convergenza
  $\mathbf{s} = A\mathbf{z}^{(k)}$
  $\alpha_k = \left(\mathbf{z}^{(k)}\right)^T \mathbf{r}^{(k)} / \left(\mathbf{z}^{(k)}\right)^T \mathbf{s}$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{z}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathbf{s}$
  risolvere $P\mathbf{z}^{(k+1)} = \mathbf{r}^{(k+1)}$
  $k = k + 1$
end for

Si è accennato al fatto che il **metodo del gradiente coniugato** è una variante del metodo del gradiente, ottenuta effettuando una differente scelta delle direzioni di discesa $\mathbf{d}^{(k)}$. Grazie a questa scelta il metodo converge, in aritmetica finita, alla soluzione in un numero finito n di passi e pertanto può considerarsi un metodo diretto. Esso viene però utilizzato come metodo iterativo in quanto, se opportunamente precondizionato, fornisce un'accurata approssimazione della soluzione in un numero di iterazioni significativamente inferiore alla dimensione del problema. Analogamente al metodo del gradiente, la generica iterazione è della forma

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}$$

dove in particolare le direzioni di discesa $\mathbf{p}^{(k)}$ sono **A-coniugate** come vedremo.
Per la costruzione del metodo consideriamo i seguenti richiami.

Definizione 3.1. Un vettore $\mathbf{x}^{(k)}$ si definisce **ottimale** rispetto ad una direzione $\mathbf{p}$ se risulta

$$\varphi(\mathbf{x}^{(k)}) \leq \varphi(\mathbf{x}^{(k)} + \lambda\mathbf{p}) \quad \forall\lambda \in \mathbb{R}.$$

Nel caso in cui la precedente valga per ogni direzione $\mathbf{p}$ in un sottospazio V, si dirà che $\mathbf{x}^{(k)}$ è ottimale rispetto a V

Si ha poi un modo equivalente per esprimere l'ottimalità di un vettore rispetto ad una direzione come espresso dal seguente

Teorema 3.1. *Condizione necessaria e sufficiente affinché un vettore* $\mathbf{x}^{(k)}$ *sia ottimale rispetto a* $\mathbf{p}$ *è che la direzione* $\mathbf{p}$ *sia ortogonale al residuo al passo* k *ossia*

$$\mathbf{p}^T\mathbf{r}^{(k)} = 0.$$

Ricordiamo che nel metodo del gradiente il parametro α_k, lunghezza del passo, viene scelto come

$$\alpha_k = \frac{\left(\mathbf{r}^{(k)}\right)^T\mathbf{r}^{(k)}}{\left(\mathbf{r}^{(k)}\right)^T A\mathbf{r}^{(k)}}$$

e grazie a tale scelta si ha che il residuo $\mathbf{r}^{(k+1)}$ risulta ortogonale a $\mathbf{r}^{(k)}$ e quindi dal teorema precedente discende che l'iterato $\mathbf{x}^{(k+1)}$ è ottimale rispetto al residuo $\mathbf{r}^{(k)}$. Questa proprietà però non è più vera per i successivi elementi della successione degli iterati generata dal metodo. Pertanto nel metodo del gradiente coniugato viene operata una scelta più accurata della direzione di discesa proprio per far si che l'ottimalità del k-esimo vettore, rispetto ad una data direzione, sia mantenuta anche dai successivi iterati.
Consideriamo $\mathbf{p}$ e $\mathbf{q}$ due generiche direzioni; supponiamo che $\mathbf{x}^{(k)}$ sia ottimale rispetto a $\mathbf{p}$ e quindi che $\mathbf{p}^T\mathbf{r}^{(k)} = 0$ e poniamo

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

Imponendo che anche $\mathbf{x}^{(k+1)}$ sia ottimale rispetto a $\mathbf{p}$ otteniamo

$$0 = \mathbf{p}^T\mathbf{r}^{(k+1)} = \mathbf{p}^T(\mathbf{r}^{(k)} - A\mathbf{q}) = -\mathbf{p}^T A\mathbf{q},$$

ossia che le direzioni $\mathbf{p}$ e $\mathbf{q}$ devono essere A-ortogonali o **A-coniugate**.
Ponendo $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$, consideriamo direzioni del tipo

$$\mathbf{p}^{(k+1)} = \mathbf{r}^{(k+1)} - \beta_k\mathbf{p}^{(k)}.$$

Affinché tutti gli iterati siano ottimali rispetto a tali direzioni, si richiede che $\mathbf{p}^{(k+1)}$ e $\mathbf{p}^{(k)}$ siano A-coniugate ossia che

$$\left(\mathbf{p}^{(k)}\right)^T A\mathbf{p}^{(k+1)} = 0,$$

da cui si ottiene che

$$\beta_k = \frac{\left(\mathbf{p}^{(k)}\right)^T A\mathbf{r}^{(k+1)}}{\left(\mathbf{p}^{(k)}\right)^T A\mathbf{p}^{(k)}}.$$

Con questa scelta del paramentro β_k si ha che la direzione $\mathbf{p}^{(k+1)}$ risulta essere A-coniugata anche con le direzioni generate in precedenza k ossia

$$\left(\mathbf{p}^{(i)}\right)^T A\mathbf{p}^{(k+1)} = 0, \quad i = 0, 1, \ldots, k.$$

La generica iterazione sarà del tipo

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}$$

dove il parametro α_k viene scelto utilizzando la (3.2); il fatto che il vettore $\mathbf{x}^{(k+1)}$ sia ottimale rispetto alle direzioni generate precedentemente $\mathbf{p}^{(i)}$, $i = 0, 1, \ldots, k$, significa che lungo queste direzioni non è possibile diminuire ulteriormente la funzione obiettivo $\varphi(\mathbf{y})$ e per questo motivo il metodo converge alla soluzione esatta in un numero finito di passi.
L'implementazione del metodo del gradiente coniugato può essere schematizzata come illustrato nell'Algoritmo 8.
Vogliamo ora esaminare un risultato relativo alla velocità di convergenza; considerando che la matrice A è simmetrica e definita positiva, si può dimostrare che la relazione

$$\|\mathbf{y}\|_A = \sqrt{\mathbf{y}^T A\mathbf{y}}$$

definisce una norma per ogni $\mathbf{y} \in \mathbb{R}^n$. Si può quindi dimostrare il

Teorema 3.2. *Dato il sistema lineare $A\mathbf{x} = \mathbf{b}$, supponiamo che la matrice A dei coefficienti sia simmetrica e definita positiva. Allora*

$$\|\mathbf{e}^{(k)}\|_A \leq 2\left(\frac{\sqrt{k_2(A)} - 1}{\sqrt{k_2(A)} + 1}\right)^k \|\mathbf{e}^{(0)}\|_A$$

dove con $\mathbf{e}^{(k)} = \mathbf{x}^{(k)} - \mathbf{x}$ denotiamo l'errore al passo k e con $k_2(A) = |\lambda_{max}(A)|/|\lambda_{min}(A)|$ il numero di condizionamento in norma 2 della matrice simmetrica A.

Algoritmo 8 Metodo del gradiente coniugato

Input: $A \in \mathbb{R}^{n\times n}$ simmetrica e definita positiva, $\mathbf{b} \in \mathbb{R}^n$,
  $\mathbf{x}^{(0)}$ vettore iniziale
$\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
$\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$
$k = 0$
for $k = 0, 1, \ldots$ fino alla convergenza
  $\mathbf{s} = A\mathbf{p}^{(k)}$
  $\delta = \left(\mathbf{p}^{(k)}\right)^T \mathbf{s}$
  $\alpha_k = \left(\mathbf{p}^{(k)}\right)^T \mathbf{r}^{(k)} / \mathbf{s}$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathbf{s}$
  $\beta_k = \mathbf{s}^T \mathbf{r}^{(k+1)} / \delta$
  $\mathbf{p}^{(k+1)} = \mathbf{r}^{(k+1)} - \beta_k \mathbf{p}^{(k)}$
  $k = k + 1$
end for

Osserviamo che è possibile dimostrare quindi che il metodo del gradiente coniugato garantisce la risoluzione di un sistema lineare simmetrico e definito positivo, più rapidamente nel caso in cui gli autovalori di A sono raggruppati lontano dall'origine.
Analogamente a quanto visto per il metodo del gradiente, anche nel caso del metodo del gradiente coniugato è possibile utilizzare un opportuno precondizionatore in modo da accelerare la convergenza. In particolare precondizionando il residuo al passo k con una matrice P che sia invertibile con un basso costo computazionale e che sia tale che $P^{-1}A \simeq I$ ossia che approssimi A, otteniamo il **metodo del gradiente coniugato precondizionato** che viene schematizzato come illustrato nell'Algoritmo 9.

Algoritmo 9 Metodo del gradiente coniugato precondizionato

Input: $A \in \mathbb{R}^{n\times n}$ simmetrica e definita positiva, $\mathbf{b} \in \mathbb{R}^n$,
  $\mathbf{x}^{(0)}$ vettore iniziale, $\mathbf{z}^{(0)}$, P
$\mathbf{r}^{(0)} = \mathbf{b} - A\mathbf{x}^{(0)}$
risolvere $P\mathbf{z}^{(0)} = \mathbf{r}^{(0)}$
$k = 0$
for $k = 0, 1, \ldots$ fino alla convergenza
  $\mathbf{s} = A\mathbf{p}^{(k)}$
  $\delta = \left(\mathbf{p}^{(k)}\right)^T \mathbf{s}$
  $\alpha_k = \left(\mathbf{p}^{(k)}\right)^T \mathbf{r}^{(k)}/\delta$
  $\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \alpha_k \mathbf{p}^{(k)}$
  $\mathbf{r}^{(k+1)} = \mathbf{r}^{(k)} - \alpha_k \mathbf{s}$
  risolvere $P\mathbf{z}^{(k+1)} = \mathbf{r}^{(k+1)}$
  $\beta_k = \mathbf{s}^T \mathbf{z}^{(k+1)}/\delta$
  $\mathbf{p}^{(k+1)} = \mathbf{z}^{(k+1)} - \beta_k \mathbf{p}^{(k)}$
  $k = k + 1$
end for

3.2 Iterazioni in sottospazi di Krylov

Definizione 3.2. Sia $A \in \mathbb{R}^{n\times n}$ e $\mathbf{b} \in \mathbb{R}^n$. Per $r = 1, 2, \ldots$, si definisce **sottospazio di Krylov** con indice r generato da A e $\mathbf{b}$, l'insieme

$$\mathcal{K}_r := span(\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, \ldots, A^{r-1}\mathbf{b}) \subseteq \mathbb{R}^n.$$

I **metodi di proiezione in sottospazi di Krylov** consistono nel proiettare un problema di dimensione n in un sottospazio di Krylov di dimensione inferiore. Essi determinano quindi, per ogni r, una soluzione approssimata $\mathbf{x}^{(r)}$ del sistema lineare $A\mathbf{x} = \mathbf{b}$ che appartenga al sottospazio $\mathcal{K}_r$.
I metodi di Krylov possono essere distinti in 4 diverse classi [15] che conducono a differenti algoritmi e che rappresentano i diversi criteri di ottimalità tramite cui scegliere il vettore $\mathbf{x}^{(r)}$:

1. l'*approccio di Ritz-Galerkin*: costruire il vettore $\mathbf{x}^{(k)}$ tale che il residuo corrispondente sia ortogonale al sottospazio di Krylov con indice k cioè $\mathbf{b} - A\mathbf{x}^{(k)} \perp \mathcal{K}_k$;

2. l'*approccio del residuo di norma minima*: identificare l'$\mathbf{x}^{(k)}$ per cui la norma Euclidea del residuo $\|\mathbf{b} - A\mathbf{x}^{(k)}\|_2$ sia minima su $\mathcal{K}_k$;

3. l'*approccio di Petrov-Galerkin*: trovare l'$\mathbf{x}^{(k)}$ tale che il residuo $\mathbf{b}-A\mathbf{x}^{(k)}$ sia ortogonale ad un qualche sottospazio di dimensione k;

4. l'*approccio dell'errore di minima norma*: determinare l'$\mathbf{x}^{(k)} \in A^T\mathbf{K}_k(A^{\mathrm{T}}, \mathbf{b})$ (sottospazio di Krylov con indice k generato da A^T e da $\mathbf{b}$) per cui la norma Euclidea dell'errore $\|\mathbf{x}^{(k)} - \mathbf{x}\|_2$ sia minima.

Si è osservato che tali criteri di ottimalità conducono a differenti algoritmi; in particolare l'approccio di Ritz-Galerkin conduce ai noti metodi del gradiente coniugato e di Lanczos. Il secondo approccio invece porta ai metodi GMRES (Generalized Minimal Residuals) e MINRES (Minimal Residuals). Questi due approcci hanno però lo svantaggio di risultare molto lenti, soprattutto nel caso di sistemi non simmetrici, e computazionalmente costosi per determinare la soluzione approssimata. Tuttavia questo viene risolto considerando altri sottospazi per la condizione di ortogononalità (come nell'approccio di Petrov-Galerkin). Il terzo approccio conduce ai metodi Bi-CG (bi-Conjugate Gradient) e QMR (Quasi-Minimal Residual), mentre il quarto non è immediato ma per $A = A^T$ conduce al metodo SYMMLQ (Symmetric LQ). Nel caso non simmetrico tale approccio conduce invece al metodo GMERR (Generalized Minimal ERRor).
Per identificare l'approssimazione $\mathbf{x}^{(l)}$ della soluzione occorre determinare una base per il sottospazio di Krylov di indice l che possa essere estesa a sottospazi di dimensione crescente. Osserviamo che una base per il sottospazio di Krylov l-dimensionale $\mathcal{K}_l$ è

$$\{\mathbf{b}, A\mathbf{b}, A^2\mathbf{b}, \ldots, A^{l-1}\mathbf{b}\};$$

tale base risulta essere numericamente instabile perché al crescere di l i vettori $A^l\mathbf{b}$ tendono all'autovettore dominante (come segue dal metodo delle potenze) e pertanto tendono ad essere linearmente indipendenti. Si potrebbe pensare di considerare questa base non ortogonale e ortogonalizzarla successivamente , ma questo potrebbe portare ad un forte mal condizionamento. Si pone quindi il problema di sostituire tale base instabile con una base ortogonale o ortonormale che caratterizza il metodo di Krylov utilizzato.

3.2.1 Il metodo CG come metodo di Krylov

È possibile mostrare che il metodo del gradiente coniugato è caratterizzato da una iterazione in sottospazi di Krylov e quindi può essere considerato come un metodo di Krylov. Vale infatti il seguente

Teorema 3.3. *Supponiamo di applicare il metodo CG al sistema lineare $A\mathbf{x} = \mathbf{b}$, con A simmetrica e definita positiva. Se il vettore iniziale è $\mathbf{x}^{(0)} = 0$, allora il metodo procede finché non si annulla il residuo $\mathbf{r}^{(k)}$ e valgono le seguenti identità, per $l = 1, \ldots, k$, per i sottospazi di Krylov*

$$\begin{aligned} \mathcal{K}_l &:= \text{span}(\mathbf{b}, A\mathbf{b}, \ldots, A^{l-1}\mathbf{b}) = \text{span}(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(l)}) \\ &= \text{span}(\mathbf{p}^{(0)}, \ldots, \mathbf{p}^{(l-1)}) = \text{span}(\mathbf{r}^{(0)}, \ldots, \mathbf{r}^{(l-1)}). \end{aligned} \tag{3.3}$$

Inoltre si ha che, per ogni l, i residui sono ortogonali

$$\left(\mathbf{r}^{(l)}\right)^{\mathrm{T}} \mathbf{r}^{(j)} = 0 \quad j = 0, 1, \ldots, l-1 \tag{3.4}$$

e le direzioni di ricerca sono A-coniugate

$$\left(\mathbf{p}^{(l)}\right)^{\mathrm{T}} A\mathbf{p}^{(j)} = 0 \quad j = 0, 1, \ldots, l-1. \tag{3.5}$$

Dimostrazione. Diamo uno schema della dimostrazione che procede per induzione su l. Per $l = 0$ il risultato è banale poiché essendo $\mathbf{x}^{(0)} = 0$ si ha che $\mathbf{r}^{(0)} = \mathbf{p}^{(0)} = \mathbf{b}$ e $\mathbf{x}^{(1)} = \alpha_0 \mathbf{p}^{(0)}$.
Supponiamo che $l > 1$; dalla relazione $\mathbf{x}^{(l)} = \mathbf{x}^{(l-1)} + \alpha_{l-1}\mathbf{p}^{(l-1)}$ segue per induzione che $\mathbf{x}^{(l)} \in span(\mathbf{p}^{(0)}, \ldots, \mathbf{p}^{(l-1)})$. Dalla relazione $\mathbf{p}^{(l-1)} = \mathbf{r}^{(l-1)} + \beta_{l-2}\mathbf{p}^{(l-2)}$ segue che $\mathbf{r}^{(l-1)} \in span(\mathbf{p}^{(0)}, \ldots, \mathbf{p}^{(l-1)})$ e infine dal fatto che $\mathbf{r}^{(l-1)} = \mathbf{r}^{(l-2)} - \alpha_{l-2}A\mathbf{p}^{(l-2)}$ segue che $A^{l-1}\mathbf{b} \in span(\mathbf{r}^{(0)}, \ldots, \mathbf{r}^{(l-1)})$. Questo prova la 3.3.
L'ortogonalità dei residui e il fatto che le direzioni siano A-coniugate, possono essere provati sfruttando le formule considerate nella costruzione del metodo del gradiente coniugato. Una dimostrazione accurata delle formule (3.4) e (3.5) si può trovare in [14]. □

Come si evince da questo teorema, il metodo del gradiente coniugato sostituisce la base di $\mathcal{K}_l$ numericamente instabile $\{\mathbf{b}, A\mathbf{b}, \ldots, A^{l-1}\mathbf{b}\}$ con la base costituita dalle direzioni A-ortogonali $\{\mathbf{p}^{(0)}, \ldots, \mathbf{p}^{(l-1)}\}$.
Ricordiamo che sotto l'ipotesi che A sia simmetrica e definita positiva e quindi che gli autovalori di A siano tutti positivi o equivalentemente $\mathbf{x}^T A\mathbf{x} \geq 0$ per ogni vettore $\mathbf{x} \in \mathbb{R}^n$ non nullo, la funzione $\|\cdot\|_A$ definita da

$$\|\cdot\|_A = \sqrt{\mathbf{x}^T A\mathbf{x}}$$

è una norma su $\mathbb{R}^n$ detta *norma-A*. Si può quindi mostrare che la generica iterata $\mathbf{x}^{(l)}$ del metodo del gradiente coniugato, possiede un'importante proprietà di ottimalità:

Teorema 3.4. *Il vettore* $\mathbf{x}^{(l)}$ *minimizza nel sottospazio di Krylov* $\mathcal{K}_l$ *la norma-A dell'errore, ossia risulta che*

$$\mathbf{x}^{(l)} = \arg\min_{\mathbf{x}\in\mathcal{K}_l} \|\mathbf{x}^* - \mathbf{x}\|_A$$

dove con $\mathbf{x}^*$ *indichiamo la soluzione esatta del sistema lineare.*
Inoltre se $\mathbf{e}^{(l)} = \mathbf{x}^* - \mathbf{x}^{(l)}$ *allora si ha*

$$\|\mathbf{e}^{(l)}\|_A \leq \|\mathbf{e}^{(l-1)}\|_A, \quad \forall l \tag{3.6}$$

e per un $k \leq n$ *si ottiene* $\mathbf{e}^{(k)} = 0$.

Dimostrazione. Consideriamo $\mathbf{x} = \mathbf{x}^{(l)} - \mathbf{z}$ un generico vettore di $\mathcal{K}_l$ e il corrispondente errore $\mathbf{e} = \mathbf{x}^* - \mathbf{x} = \mathbf{e}^{(l)} + \mathbf{z}$. Calcolando il quadrato della norma-A di tale errore, troviamo

$$\begin{aligned} \|\mathbf{e}\|_A^2 &= (\mathbf{e}^{(l)} + \mathbf{z})^T A(\mathbf{e}^{(l)} + \mathbf{z}) = (\mathbf{e}^{(l)})^T A\mathbf{e}^{(l)} + \mathbf{z}^T A\mathbf{z} + 2\mathbf{z}^T A\mathbf{e}^{(l)} \\ &= (\mathbf{e}^{(l)})^T A\mathbf{e}^{(l)} + \mathbf{z}^T A\mathbf{z} + 2\mathbf{z}^T\mathbf{r}^{(l)} = (\mathbf{e}^{(l)})^T A\mathbf{e}^{(l)} + \mathbf{z}^T A\mathbf{z}, \end{aligned} \tag{3.7}$$

dove l'ultima uguaglianza segue dal fatto che il residuo $\mathbf{r}^{(l)}$ è ortogonale al sottospazio $\mathcal{K}_l$ a cui appartiene $\mathbf{z}$. Poiché $\mathbf{y}^T A\mathbf{y} \geq 0$ per ogni $\mathbf{y}$, il minimo di $\|\mathbf{e}\|_A$ si otterrà per $\mathbf{z} = 0$ e quindi per $\mathbf{x} = \mathbf{x}^{(l)}$.
La proprietà di monotonia espressa dalla (3.6) è una conseguenza del fatto che dall'inclusione $\mathcal{K}_{l-1} \subseteq \mathcal{K}_l$ segue che

$$\min_{\mathbf{x}\in\mathcal{K}_l} \|\mathbf{x}^* - \mathbf{x}\|_A \leq \min_{\mathbf{x}\in\mathcal{K}_{l-1}} \|\mathbf{x}^* - \mathbf{x}\|_A$$

e del fatto che un sottospazio di Krylov termina in al più n passi. □

3.3 Il metodo di Arnoldi

3.3.1 CGS e MGS

Consideriamo il processo di ortogonalizzazione di Gram-Schimdt nella sua versione classica (CGS) e modificata (MGS), che fornisce un metodo per il calcolo della fattorizzazione QR di una matrice costruendo i vettori $\mathbf{q}_1, \ldots, \mathbf{q}_n$ e le entrate r_{ij} tramite un processo di successive ortogonalizzazioni.

Classical Gram-Schmidt. Il processo di Gram-Schmidt nella sua versione classica, spesso indicato con l'acronimo **CGS**, consente di ortogonalizzare n vettori di $\mathbb{R}^m$ con $m \geq n$. Se immaginiamo tali vettori come colonne di una

matrice $A \in \mathbb{R}^{m\times n}$, questo metodo fornisce appunto una fattorizzazione QR di A calcolando la matrice

$$Q = [\mathbf{q}_1 \quad \mathbf{q}_2 \cdots \mathbf{q}_n]$$

tale che

$$\text{span}\{\mathbf{q}_1, \ldots, \mathbf{q}_n\} = \text{span}\{\mathbf{a}_1, \ldots, \mathbf{a}_n\} \quad \text{con} \quad \mathbf{q}_i^T \mathbf{q}_j = \delta_{ij}.$$

Al primo passo , facendo uso della norma euclidea, si pone

$$\mathbf{v}_1 = \mathbf{a}_1, \quad r_{11} = \|\mathbf{v}_1\| \quad \text{e} \quad \mathbf{q}_1 = \frac{\mathbf{v}_1}{r_{11}}.$$

Al generico passo k viene calcolato

$$\mathbf{v}_k = \mathbf{a}_k - \sum_{j=1}^{k-1} r_{jk}\mathbf{q}_j$$

con

$$r_{jk} = \mathbf{q}_j^T \mathbf{a}_k, \quad j = 1, \ldots, k-1,$$

e si ottiene

$$\mathbf{q}_k = \frac{\mathbf{v}_k}{r_{kk}}, \quad r_{kk} = \|\mathbf{v}_k\|.$$

Grazie alla scelta effettuata per i coefficienti r_{jk}, si ha che ciascun vettore $\mathbf{q}_k$ risulta ortonormale ai precedenti.
Dalle precedenti relazioni si può ricavare che

$$\mathbf{a}_k = \sum_{j=1}^{k} r_{jk}\mathbf{q}_j, \quad k = 1, \ldots, n,$$

e quindi che la matrice $A = [\mathbf{a}_1\mathbf{a}_2 \cdots \mathbf{a}_n]$ può essere rappresentata tramite il prodotto

$$[\mathbf{q}_1 \cdots \mathbf{q}_k \cdots \mathbf{q}_n] \begin{bmatrix} r_{11} & \cdots & r_{1k} & \cdots & r_{1n} \\ & \ddots & \vdots & & \vdots \\ & & r_{kk} & & \vdots \\ & & & \ddots & \vdots \\ & & & & r_{nn} \end{bmatrix}.$$

Quindi si evince che, effettuando l'ortonormalizzazione delle colonne di A l'algoritmo ne fornisce la fattorizzazione QR. Osserviamo che in questo procedimento il fattore R viene costruito per colonne e quando $n < m$ la matrice

Algoritmo 10 CGS

Input: $A \in \mathbb{R}^{m\times n}$
for $k = 1, \dots, n$
  $\mathbf{v}_k = \mathbf{a}_k$
  for $j = 1, \dots, k-1$
    $r_{jk} = \mathbf{q}_j^T \mathbf{a}_k$
    $\mathbf{v}_k = \mathbf{v}_k - r_{jk}\mathbf{q}_j$
  end for
  $r_{kk} = \|\mathbf{v}_k\|$
  $\mathbf{q}_k = \mathbf{v}_k / r_{kk}$
end for

Q pur essendo costituita da colonne ortonormali, non è ortogonale perché non è quadrata. Per ottenere una matrice ortogonale, essa viene completata aggiungendo $m - n$ vettori che completino la base ortonormale.
Il procedimento descritto può essere schematizzato come illustrato nell'Algoritmo 10
Questo algoritmo tuttavia è numericamente instabile e in particolare al crescere delle iterazioni si ha che le colonne di Q tendono a perdere la reciproca ortogonalità. Per questo motivo è stata studiata una differente versione del procedimento di Gram-Schmidt detta **MGS** (*Modified Gram-Schmidt*).

Modified Gram-Schmidt. Questa versione modificata dell'ortogonalizzazione di Gram-Schmidt permette anch'essa di ottenere la fattorizzazione QR di una matrice A ma la differenza consiste nel fatto che in questo caso la matrice triangolare superiore R verrà costruita per righe e non per colonne. Questo algoritmo genera una successione di $n + 1$ matrici tali che $A^{(1)} = A$ e $A^{(n+1)} = Q \in \mathbb{R}^{m\times n}$. La k-esima matrice della successione generata dal metodo sarà della forma

$$A^{(k)} = \begin{bmatrix} \mathbf{q}_1 & \cdots & \mathbf{q}_{k-1} & \mathbf{a}_k^{(k)} & \cdots & \mathbf{a}_n^{(k)} \end{bmatrix}$$

con

$$\begin{cases} \mathbf{q}_i^T \mathbf{q}_j = \delta_{ij}, & i, j = 1, \dots, k-1 \\ \mathbf{q}_i^T \mathbf{a}_j^{(k)} = 0, & i = 1, \dots, k-1, j = k, \dots, n. \end{cases}$$

Al generico passo k si pone

$$r_{kk} = \|\mathbf{a}_k^{(k)}\|,$$

$$\mathbf{q}_k = \frac{\mathbf{a}_k^{(k)}}{r_{kk}},$$

$$r_{kj} = \mathbf{q}_k^T \mathbf{a}_j^{(k)}, \qquad j = k+1, \ldots, n$$

$$\mathbf{a}_j^{(k+1)} = \mathbf{a}_j^{(k)} - r_{kj}\mathbf{q}_k, \quad j = k+1, \ldots, n.$$

In questo modo si generano due famiglie di vettori $\{\mathbf{q}_1, \ldots, \mathbf{q}_k\}$ e $\{\mathbf{a}_{k+1}^{(k+1)}, \ldots, \mathbf{a}_n^{(k+1)}\}$, costituite rispettivamente da vettori tra loro ortonormali e da vettori ortogonali a tutti gli elementi della prima. Dopo n passi la prima famiglia conterrà n vettori ortonormali mentre la seconda non conterrà più alcun elemento. Vediamo che anche con questo procedimento è possibile ottenere la fattorizzazione QR della matrice iniziale A, infatti dalla relazione

$$\mathbf{a}_j^{(k)} = \mathbf{a}_j^{(k+1)} + r_{kj}\mathbf{q}_k$$

si ottiene

$$\begin{aligned}\mathbf{a}_j = \mathbf{a}_j^{(1)} &= \mathbf{a}_j^{(2)} + r_{1j}\mathbf{q}_1 = \mathbf{a}_j^{(3)} + r_{1j}\mathbf{q}_1 + r_{2j}\mathbf{q}_2 = \ldots \\ &= \mathbf{a}_j^{(j)} + \sum_{i=1}^{j-1} r_{ij}\mathbf{q}_i.\end{aligned}$$

Questo procedimento può essere schematizzato come mostrato nell'Algoritmo 11.

Algoritmo 11 MGS

Input: $A \in \mathbb{R}^{m\times n}$
for $k = 1, \ldots, n$
  $\mathbf{v}_k = \mathbf{a}_k$
  $r_{kk} = \|\mathbf{v}_k\|$
  $\mathbf{q}_k = \mathbf{v}_k / r_{kk}$
  for $j = k+1, \ldots, n$
    $r_{kj} = \mathbf{q}_k^T \mathbf{v}_j$
    $\mathbf{v}_j = \mathbf{v}_j - r_{kj}\mathbf{q}_k$
  end for
end for

3.3.2 L'iterazione di Arnoldi

Il **metodo di Arnoldi** è un metodo di proiezione in sottospazi di Krylov per matrici non Hermitiane. Questa procedura fu introdotta nel 1951 come metodo per ridurre una matrice densa in forma di Hessenberg e vedremo che infatti essa permette di ottenere una fattorizzazione matriciale.
L'iterazione di Arnoldi consente di costruire una base $\{\mathbf{q}_1, \ldots, \mathbf{q}_l\}$ per il sottospazio di Krylov $\mathcal{K}_l$ costituita da vettori tra loro ortonormali, in sostituzione della base numericamente instabile $\{\mathbf{b}, A\mathbf{b}, \ldots, A^{l-1}\mathbf{b}\}$. Questo può essere fatto applicando sia il classico metodo di ortogonalizzazione di Gram-Schmidt sia la sua versione modificata MGS. In entrambi i casi sarà possibile ottenere una fattorizzazione di $A \in \mathbb{R}^{n\times n}$ della forma

$$AQ_l = Q_{l+1}\tilde{H}_l$$

dove $Q_l \in \mathbb{R}^{n\times l}$, $Q_{l+1} \in \mathbb{R}^{n\times(l+1)}$ e $\tilde{H}_l \in \mathbb{R}^{(l+1)\times l}$ è data da

$$\tilde{H}_l = \begin{bmatrix} h_{11} & h_{12} & \cdots & \cdots & h_{1l} \\ h_{21} & h_{22} & \cdots & \cdots & h_{2l} \\ & h_{32} & \ddots & & h_{3l} \\ & & \ddots & \ddots & \vdots \\ & & & h_{l,l-1} & h_{ll} \\ & & & & h_{l+1,l} \end{bmatrix}.$$

Il metodo di Arnoldi che abbiamo implementato in MATLAB è riportato nell'Algoritmo 12 in cui abbiamo considerato sia l'ortogonalizzazione tramite CGS sia quella tramite MGS.
Nella **linea 4** si costruisce la prima colonna della matrice Q che costituisce quindi il primo vettore della base ortonormale per il sottospazio di Krylov di dimensione k dove k è il numero di iterazioni dato come input.
Linee 5-25: l'algoritmo genera la matrice $Q_{k+1} \in \mathbb{R}^{n\times(k+1)}$ e la matrice $\tilde{H}_k \in \mathbb{R}^{(k+1)\times k}$ contenente al suo interno, come abbiamo visto, la matrice $H_k \in \mathbb{R}^{k\times k}$ in forma di Hessenberg.
Linee 7-15: si procede all'ortogonalizzazione dei vettori $\mathbf{v}_j$ man mano che essi vengono costruiti (**linea 4**), utilizzando una variabile logica *mgs* che posta inizialmente pari a 0 procede con il procedimento di Gram-Schmidt classico, altrimenti posta pari a 1 utilizza Gram-Schidt modificato.
Linee 16-20: richiamando una variabile logica *reorth* che fa parte delle opzioni date in input, come *mgs*, si effettua una riortogonalizzazione dei vettori $\mathbf{v}_j$ che normalizzati saranno le colonne della matrice Q. La riortogonalizzazione viene effettuata in quanto, al crescere del numero delle iterazioni, i vettori $\mathbf{v}_j$ potrebbero perdere la rispettiva ortogonalità.

Algoritmo 12 Decomposizione di Arnoldi

1: **Input:** $A \in \mathbb{R}^{n\times n}$,$\mathbf{b} \in \mathbb{R}^n$,
2: numero di iterazioni k, tolleranza τ, opzioni
3: $Q = O \in \mathbb{R}^{n\times(k+1)}$, $H = O \in \mathbb{R}^{(k+1)\times k}$
4: $\mathbf{q}_1 = \mathbf{b}/\|\mathbf{b}\|$
5: **for** $j = 1, \ldots, k$
6: $\mathbf{v} = A\mathbf{q}_j$
7: **if** mgs *// ortogonalizzazione tramite MGS*
8: **for** $i = 1, \ldots, j$
9: $h_{ij} = \mathbf{q}_i^T\mathbf{v}$
10: $\mathbf{v} = \mathbf{v} - h_{ij}\mathbf{q}_i$
11: **end for**
12: **else** *// ortogonalizzazione tramite CGS*
13: $h_{1:j,j} = \mathbf{q}_{(:,1:j)}^T\mathbf{v}$
14: $\mathbf{v} = \mathbf{v} - \mathbf{q}_{(:,1:j)}h_{1:j,j}$
15: **end if**
16: **if** reorth *// riortogonalizzazione delle colonne di Q*
17: **for** $i = 1, \ldots, j$
18: $\mathbf{v} = \mathbf{v} - (\mathbf{q}_i^T\mathbf{v})\mathbf{q}_i$
19: **end for**
20: **end if**
21: $h_{j+1,j} = \|\mathbf{v}\|$ *// costruzione di H*
22: **if** $h_{j+1,j} < \tau$ **then** uscita per breakdown **end**
23: $\mathbf{q}_{j+1} = \mathbf{v}/h_{j+1,j}$
24: **end for**
25: **Output:** $H \in \mathbb{R}^{(k+1)\times k}$ matrice di Hessenberg
26: $Q \in \mathbb{R}^{n\times k}$ con colonne ortonormali che costituiscono
27: una base per il sottospazio di Krylov $\mathcal{K}_k$

Nella **linea 21** si procede alla costruzione della matrice H il cui generico elemento sotto diagonale è $h_{j+1,j} = \|\mathbf{v}_j\|$; osserviamo che se per un dato indice $j < n$ la norma $h_{j+1,j}$ dovesse essere inferiore ad una tolleranza τ prefissata si presenta un *breakdown* (**linea 22**) che interrompe l'algoritmo

che come output avrà le matrici Q e H "tagliate" ossia di dimensioni $n \times j$ e $j \times j$. Osserviamo che il verificarsi di un breakdown al passo j indica che il vettore $A\mathbf{q}_j$ è linearmente dipendente dai precedenti e quindi lo spazio di Krylov con indice j contiene la soluzione del sistema lineare $A\mathbf{x} = \mathbf{b}$ come vedremo meglio in seguito nell'applicazione del metodo di Arnoldi proprio alla risoluzione dei sistemi lineari.
Nella **linea 24** vengono calcolate le colonne ortonormali della matrice Q.
L'algoritmo descritto permette di ottenere una fattorizzazione di A del tipo

$$AQ_k = Q_{k+1}\tilde{H}_k; \tag{3.8}$$

da questa relazione si ottiene

$$Q_k^T A Q_k = H_k := \begin{bmatrix} h_{11} & h_{12} & \cdots & \cdots & h_{1l} \\ h_{21} & h_{22} & \cdots & \cdots & h_{2l} \\ & h_{32} & \ddots & & h_{3l} \\ & & \ddots & \ddots & \vdots \\ & & & h_{l,l-1} & h_{ll} \end{bmatrix}$$

che per $k = n$ assume la forma

$$Q_n^T A Q_n = H_n,$$

dove $Q_n^T Q_n = Q_n Q_n^T = I \in \mathbb{R}^{n \times n}$. Questo mostra appunto che l'iterazione di Arnoldi porte la matrice A in forma di Hessenberg tramite n trasformazioni ortogonali. Poiché A e H_n sono unitariamente simili esse hanno gli stessi autovalori e quindi l'algoritmo di Arnoldi può essere sfruttato come passo iniziale per l'applicazione di un metodo per il calcolo degli autovalori, come ad esempio quello QR, che tragga vantaggio dalla struttura di Hessenberg. Vedremo successivamente come applicare il metodo di Arnoldi per approssimare gli autovalori estremali di una matrice A di grandi dimensioni.

3.3.3 Il metodo GMRES

L'applicazione dell'iterazione di Arnoldi alla risoluzione di un sistema lineare $A\mathbf{x} = \mathbf{b}$ con A non singolare di dimensione n porta al **metodo GMRES** (*Generalized Minimal RESiduals*). Questo metodo consiste nel costruire iterativamente una base ortonormale, tramite il metodo di Arnoldi, per lo spazio di Krylov $\mathcal{K}_l$, con $l = 1, \ldots, n$, approssimando ad ogni passo la soluzione del sistema $\mathbf{x}^*$ col vettore $\mathbf{x}^{(l)} \in \mathcal{K}_l$ che sia tale da minimizzare la norma-2 del residuo $\mathbf{r}^{(l)} = \mathbf{b} - A\mathbf{x}^{(l)}$.

In pratica se poniamo

$$\mathbf{x}^{(l)} = \arg\min_{\mathbf{x}\in\mathcal{K}_l} \|\mathbf{b} - A\mathbf{x}\|_2,$$

la soluzione viene proiettata nel sottospazio di Krylov $\mathcal{K}_l$ e viene determinata risolvendo un problema ai minimi quadrati. Osserviamo che per le proprietà dell'iterazione di Arnoldi, il metodo in molti casi fa si che si possa ottenere un'approssimazione sufficientemente accurata della soluzione del sistema in un numero di passi molto inferiore a n, pur essendo un metodo che termina in al più n passi.
Vediamo come si procede in maniera operativa ricordando che la matrice in forma di Hessenberg

$$H_l = Q_l^T A Q_l$$

è la rappresentazione nella base ortonormale $\mathbf{q}_1, \ldots, \mathbf{q}_l$ della proiezione ortogonale di A sul sottospazio $\mathcal{K}_l$. Infatti preso un generico vettore $\mathbf{y} \in \mathbb{R}^l$, il prodotto $Q_l\mathbf{y}$ esprime un generico vettore di $\mathcal{K}_l$ come combinazione lineare degli elementi della base ortonormale considerata. Si ha quindi

$$\|\mathbf{b} - A\mathbf{x}\| = \|AQ_l\mathbf{y} - \mathbf{b}\|.$$

Tenendo conto della (3.8) si ha

$$\|AQ_l\mathbf{y} - \mathbf{b}\| = \|Q_{l+1}\tilde{H}_l\mathbf{y} - \mathbf{b}\|,$$

da cui essendo $Q_{l+1}^T Q_{l+1} = I_{l+1}$ e osservando che $Q_{l+1}^T\mathbf{b} = \|\mathbf{b}\|\mathbf{e}_1$ si ottiene

$$\|\mathbf{b} - A\mathbf{x}\| = \left\|\tilde{H}_l\mathbf{y} - \|\mathbf{b}\|\mathbf{e}_1\right\|.$$

Quindi si ha che per risolvere il problema ai minimi quadrati

$$\mathbf{x}^{(l)} = \arg\min_{\mathbf{x}\in\mathcal{K}_l} \|\mathbf{b} - A\mathbf{x}\|_2$$

è possibile calcolare la soluzione $\mathbf{y}^{(l)}$ del problema di minimizzazione

$$\min_{\mathbf{y}\in\mathbb{R}^l}\left\|\tilde{H}_l\mathbf{y} - \|\mathbf{b}\|\mathbf{e}_1\right\|,$$

e calcolare successivamente $\mathbf{x}^{(l)} = Q_l\mathbf{y}^{(l)}$.
Abbiamo implementato il metodo GMRES come schematizzato negli Algoritmi 13 e 14.

Algoritmo 13 Metodo GMRES: inizializzazione

Input: $A \in \mathbb{R}^{n\times n}$,$\mathbf{b} \in \mathbb{R}^n$,
numero massimo di iterazioni *maxn*, tolleranza
per il breakdownτ, tolleranza *tol*, opzioni
maxn= $\min(n, 200)$,$\mathbf{x} = \mathbf{0} \in \mathbb{R}^n$
$\mathbf{res} = \mathbf{0} \in \mathbb{R}^{maxn}$,
$Q = O \in \mathbb{R}^{n\times maxn}$, $H = O \in \mathbb{R}^{(maxn+1)\times maxn}$
$\mathbf{q}_1 = \mathbf{b}/\|\mathbf{b}\|$
$\mathbf{v} = A\mathbf{q}_1$
$h_{11} = \mathbf{q}_1^T\mathbf{v}$,
$\mathbf{v} = \mathbf{v} - h_{11}\mathbf{q}_1$
$h_{21} = \|\mathbf{v}\|$, $\mathbf{q}_2 = \mathbf{v}/h_{21}$
$\mathbf{y} = H_{2,1}[\|\mathbf{b}; 0\|]$,

3.3.4 Approssimazione degli autovalori

L'iterazione di Arnoldi può essere utilizzata, come si è già osservato, per la ricerca degli autovalori della matrice A a cui è applicato dato che esso genera una riduzione di A in forma di Hessenberg tramite delle trasformazioni ortogonali. Una volta calcolata la matrice $H_l = Q_l^T AQ_l \in \mathbb{R}^{l\times l}$ è possibile calcolare, con un minor costo computazionale dovuto alla particolare struttura che essa presenta, i suoi autovalori detti *valori di Ritz* e autovettori corrispondenti. Per questo motivo tale calcolo può essere effettuato utilizzando la funzione `eig` predefinita in MATLAB.
Si può dimostrare, come in [14], che gli autovalori estremali di H_l convergono agli autovalori estremali di A e anzi, al crescere del numero delle iterazioni, un numero crescente di autovalori delle due matrici tende a coincidere. Questo permette, data una matrice di grandi dimensioni, ottenere delle approssimazioni sufficientemente accurate dei suoi autovalori estremali e questo può essere fatto con un costo computazionale relativamente basso essendo la matrice strutturata e di dimensione inferiore alla dimensione della matrice A di partenza. Utilizzando l'iterazione di Arnoldi, abbiamo implementato un algoritmo che permette l'approssimazione di un certo numero di autovalori k di una matrice non simmetrica di dimensione n. Il metodo di Arnoldi razionale è una variante dell'iterazione di Arnoldi che può essere applicata sia al calcolo di un certo numero di autovalori di una matrice non Hermitiana di grandi dimensioni, sia all'approssimazione di funzioni matriciali. Queste

Algoritmo 14 Metodo GMRES: ciclo principale

13: $k = 1$
14: **repeat**
15: $k = k + 1$, $\mathbf{y0} = \mathbf{y}$
16: $\mathbf{v} = A\mathbf{q}_k$
17: **if** mgs *// ortogonalizzazione tramite MGS*
18: **for** $i = 1, \ldots, k$
19: $h_{ik} = \mathbf{q}_i^T \mathbf{v}$
20: $\mathbf{v} = \mathbf{v} - h_{ik}\mathbf{q}_i$
21: **end for**
22: **else** *// ortogonalizzazione tramite CGS*
23: $h_{1:k,k} = \mathbf{q}_{(:,1:k)}^T \mathbf{v}$
24: $\mathbf{v} = \mathbf{v} - \mathbf{q}_{(:,1:k)} h_{1:k,k}$
25: **end if**
26: **if** reorth *// riortogonalizzazione delle colonne di Q*
27: **for** $i = 1, \ldots, k$
28: $\mathbf{v} = \mathbf{v} - (\mathbf{q}_i^T \mathbf{v})\mathbf{q}_i$
29: **end for**
30: **end if**
31: $h_{k+1,k} = \|\mathbf{v}\|$ *// costruzione di H*
32: **if** $h_{k+1,k} < \tau$ **then** uscita per breakdown **end**
33: $\mathbf{q}_{k+1} = \mathbf{v}/h_{k+1,k}$
34: $\mathbf{y} = H_{k+1,k}/\|\mathbf{b}\|\mathbf{e}_1$
35: $\text{res}_k = \|H_{k+1,k}\mathbf{y} - \|\mathbf{b}\|\mathbf{e}_1$
36: **until** $(\|\mathbf{y} - [y0; 0]\|) < tol \cdot \|\mathbf{y}\|$ or $k < maxn$
37: $\mathbf{x} = Q_{n,k}\mathbf{y}$
38: **Output:** $H \in \mathbb{R}^{(k+1)\times k}$ matrice di Hessenberg
39: $Q \in \mathbb{R}^{n\times k}$ con colonne ortonormali che costituiscono
40: una base per il sottospazio di Krylov $\mathcal{K}_k$

applicazioni unitamente al calcolo degli pseudospettri sono discusse in [8]. Lo schema strutturale del metodo di Arnoldi applicato all'approssimazione degli autovalori di una matrice, è illustrato nell'Algoritmo 15.

Algoritmo 15 Metodo di Arnoldi per l'approssimazione di autovalori

Input: $A \in \mathbb{R}^{n\times n}$, $\mathbf{b} \in \mathbb{R}^n$,
numero massimo di iterazioni N, tolleranza di breakdown τ,
tolleranza *toll*, opzioni, k numero autovalori che si vogliono
approssimare
$Q = O \in \mathbb{R}^{n\times N}$, $H = O \in \mathbb{R}^{(N+1)\times N}$
$\mathbf{q}_1 = \mathbf{b}/\|\mathbf{b}\|$ // costruzione della prima colonna di Q
$j = 0$
lam=inf // inizializziamo il vettore che conterrà i k autovalori di H_k
repeat
  $j = j + 1$
  costruzione delle matrici H e Q tramite l'iterazione di Arnoldi 12
  calcolo degli autovalori di H_k posti nel vettore lam
  in ordine decrescente
until convergenza
calcolo dell'errore di approssimazione
Output: $H \in \mathbb{R}^{(k+1)\times k}$ matrice di Hessenberg
$Q \in \mathbb{R}^{n\times k}$ con colonne ortonormali che costituiscono
una base per il sottospazio di Krylov $\mathcal{K}_k$
approssimazione dei primi k autovalori di A

3.4 Il metodo di Lanczos

L'iterazione di Lanczos non è altro che il metodo di Arnoldi applicato ad una matrice Hermitiana. Supponiamo, per semplicità di notazione, che la matrice A a cui applicare il metodo sia reale e simmetrica.

Vediamo cosa capita se applichiamo il processo di Arnoldi in questo particolare caso. Osserviamo innanzitutto che in questo caso la matrice $H_l = Q_l^T A Q_l$ (detta matrice di Ritz) sarà simmetrica oltre che essere in forma di Hessenberg e quindi necessariamente dovrà essere tridiagonale. Essa sarà tale quindi che i suoi autovalori, i *valori di Ritz*, sono reali. Il metodo di Lanczos quindi consente di ottenere una fattorizzazione $AQ_l = Q_{l+1}\tilde{H}_l$ dove $Q_l = [\mathbf{q}_1 \quad \mathbf{q}_2 \cdots \mathbf{q}_l] \in \mathbb{R}^{n\times l}$ è costituita da colonne ortonormali che costituiscono una base per il sottospazio di Krylov di dimensione l.

Ponendo $h_{ii} = \alpha_i$ e $h_{i+1,i} = h_{i,i+1} = \beta_i$, la matrice H_l diviene

$$H_l = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \beta_{l-1} \\ & & & \beta_{l-1} & \alpha_l \end{bmatrix}$$

Il fatto che la matrice H_l abbia questa particolare struttura, porta ad un abbattimento della complessità computazionale.
L'iterazione di Lanczos porterà, quindi, alla fattorizzazione $AQ_l = Q_{l+1}\tilde{H}_l$ con $\tilde{H}_l \in \mathbb{R}^{(l+1)\times l}$ costituita dalla matrice tridiagonale H_l e da una riga costituita da elementi nulli tranne l'ultimo pari a β_l ovvero della forma

$$\tilde{H}_l = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \beta_2 & \alpha_3 & \ddots & \\ & & \ddots & \ddots & \beta_{l-1} \\ & & & \beta_{l-1} & \alpha_l \\ & & & & \beta_l \end{bmatrix}. \tag{3.9}$$

Prima di considerare una schematizzazione del metodo implemetato in MATLAB, vediamo la forma che assume l'iterazione di Lanczos. Al primo passo si pone

$$\mathbf{q}_1 = \frac{\mathbf{b}}{\|\mathbf{b}\|}$$

che risulta essere la prima colonna della matrice ortogonale Q_l. Si impone poi che

$$A\mathbf{q}_1 = \alpha_1\mathbf{q}_1 + \beta_1\mathbf{q}_2$$

dove α_1 e β_1 sono gli elementi della prima colonna di H_l che determiniamo tramite i seguenti passaggi:

$$\tilde{\mathbf{q}}_2 = \beta_1\mathbf{q}_2 = A\mathbf{q}_1 - \alpha_1\mathbf{q}_1,$$

imponendo la seguente condizione di ortogonalità

$$0 = \mathbf{q}_1^T\tilde{\mathbf{q}}_2 = \mathbf{q}_1^T A\mathbf{q}_1 - \alpha_1\mathbf{q}_1^T\mathbf{q}_1$$

otteniamo

$$\alpha_1 = \mathbf{q}_1^T A\mathbf{q}_1$$

e poniamo $\beta_1 = \|\tilde{\mathbf{q}}_2\|$. La seconda colonne di Q sarà data da $\mathbf{q}_2 = \frac{\tilde{\mathbf{q}}_2}{\beta_1}$.

Al generico passo j l'iterazione di Lanczos avrà la forma

$$A\mathbf{q}_j = \beta_{j-1}\mathbf{q}_{j-1} + \alpha_j\mathbf{q}_j + \beta_j\mathbf{q}_{j+1}, \quad j = 1, \ldots, l,$$

$$\tilde{\mathbf{q}_{j+1}} = A\mathbf{q}_j - \alpha_j\mathbf{q}_j - \beta_{j-1}\mathbf{q}_{j-1}, \quad j = 1, \ldots, l,$$

$$\mathbf{q}_{j+1} = \frac{\tilde{\mathbf{q}_{j+1}}}{\beta_j}, \quad j = 1, \ldots, l,$$

con

$$\alpha_j = \mathbf{q}_j^T A\mathbf{q}_j, \qquad \beta_j = \|\tilde{\mathbf{q}_{j+1}}\|.$$

La mappa strutturale del metodo implementato su MATLAB è riportata nell'Algoritmo 16.
Come nel metodo di Arnoldi, quando ad un certo passo $j < k$ si verifica un *breakdown* (**linea 26**) in output il metodo implementato avrà le matrici Q e H ridotte e cioè di dimensione $n \times j$ e $j \times j$ essendo appunto j l'indice in cui si presenta un breakdown. Il presentarsi di un breakdown nel caso dell'applicazione del metodo alla risoluzione di un sistema lineare $A\mathbf{x} = \mathbf{b}$ è un buon segno in quanto, come già osservato, il sottospazio di Krylov $\mathcal{K}_j$ conterrà la soluzione esatta del sistema lineare simmetrico.

3.4.1 Il metodo MINRES

L'applicazione del metodo di Lanczos alla risoluzione di un sistema lineare simmetrico $A\mathbf{x} = \mathbf{b}$ conduce al metodo MINRES (Minimal Residuals) che risulta essere il metodo GMRES applicato a matrici simmetriche; quindi si procede come nel metodo GMRES classico calcolando ad ogni iterazione l'approssimazione della soluzione $\mathbf{x}^*$ del sistema rappresentata dal vettore $\mathbf{x}^{(l)}$ che minimizzi la norma euclidea del residuo al passo l. Tale soluzione approssimata, come si è visto, si calcola come

$$\mathbf{x}^{(l)} = Q_l\mathbf{y}^{(l)}$$

dove $\mathbf{y}^{(l)}$ risulta essere la soluzione del problema di minimo

$$\min_{\mathbf{y}\in\mathbb{R}^l} \left\| \tilde{H}_l\mathbf{y} - \|\mathbf{b}\|\mathbf{e}_1 \right\|,$$

in cui la matrice $\tilde{H}_l$ è data appunto dalla (3.9).
L'iterazione di Lanczos implementata per la risoluzione di un sistema lineare $A\mathbf{x} = \mathbf{b}$ con matrice A dei coefficienti simmetrica, viene schematizzata nell' Algoritmo 17–18.

Algoritmo 16 Metodo di Lanczos

Input: $A \in \mathbb{R}^{n\times n}$ simmetrica,$\mathbf{b} \in \mathbb{R}^n$,
numero di iterazioni k, tolleranza τ, opzioni
$Q = O \in \mathbb{R}^{n\times(k+1)}$, $H = O \in \mathbb{R}^{(k+1)\times k}$
$\mathbf{q}_1 = \mathbf{b}/\|\mathbf{b}\|$, $\mathbf{v} = A\mathbf{q}_1$
$\alpha = \mathbf{q}_1^T\mathbf{v}, \quad h_{11} = \alpha$
$\mathbf{v} - \mathbf{v} - \alpha\mathbf{q}_1$
$\beta = \|\mathbf{v}\|, \quad h_{21} = \beta$, $\mathbf{q}_2 = \mathbf{v}/\beta$
for $j = 2, \ldots, k$
  $h_{j-1,j} = \beta$
  $\mathbf{v} = A\mathbf{q}_j$
  if mgs *// ortogonalizzazione tramite MGS*
    $\mathbf{v} = \mathbf{v} - \beta\mathbf{q}_{j-1}, \quad \alpha = \mathbf{q}_j^T\mathbf{v}, \quad \mathbf{v} = \mathbf{v} - \alpha\mathbf{q}_j$
  else *// ortogonalizzazione tramite CGS*
    $\alpha = \mathbf{q}_j^T\mathbf{v}$, $\mathbf{v} = \mathbf{v} - \alpha\mathbf{q}_j - \beta\mathbf{q}_{j-1}$
  end if
  if reorth *// riortogonalizzazione delle colonne di Q*
    for $i = 1, \ldots, j$
      $\mathbf{v} = \mathbf{v} - (\mathbf{v}^T\mathbf{q}_i)\mathbf{q}_i$
    end for
  end if
  $\beta = \|\mathbf{v}\|$ *// costruzione di H*
  if $\beta < \tau$ **then** uscita per breakdown **end**
  $\mathbf{q}_{j+1} = \mathbf{v}/\beta$
  $h_{jj} = \alpha$ *// costruzione della diagonale principale di A*
  $h_{j+1,j} = \beta$
end for
Output: $H \in \mathbb{R}^{(k+1)\times k}$ matrice tridiagonale
$Q \in \mathbb{R}^{n\times k}$ con colonne ortonormali che costituiscono
una base per il sottospazio di Krylov $\mathcal{K}_k$

3.4.2 Applicazione al calcolo degli autovalori

Anche utilizzando l'iterazione di Lanczos è possibile approssimare gli autovalori di una matrice simmetrica A di ordine n. Una volta calcolata la matrice

Algoritmo 17 Metodo di Lanczos per sistemi lineari: inizializzazione

Input: $A \in \mathbb{R}^{n\times n}$ simmetrica, $\mathbf{x}^* = \mathbf{1} \in R^n$ // soluzione esatta,
$\mathbf{b} = A\mathbf{x}^* \in \mathbb{R}^n$ // termine noto del sistema,
tolleranza per il breakdown τ,
tolleranza *tol*, opzioni(*mgs*, *reorth*, *sol*)
maxn $N = \min(n, 200)$, $\mathbf{x} = \mathbf{0} \in \mathbb{R}^n$
if sol
  $\mathbf{e} = \mathbf{0} \in \mathbb{R}^N$ // vettore degli errori
end if
$\mathbf{res} = \mathbf{0} \in \mathbb{R}^N$,
$Q = O \in \mathbb{R}^{n\times N}$, $H = O \in \mathbb{R}^{(N+1)\times N}$
$\mathbf{q}_1 = \mathbf{b}/\|\mathbf{b}\|$
$\mathbf{v} = A\mathbf{q}_1$
$\alpha = \mathbf{q}_1^T\mathbf{v}$, $h_{11} = \alpha$
$\mathbf{v} - \mathbf{v} - \alpha\mathbf{q}_1$
$\beta = \|\mathbf{v}\|$, $h_{21} = \beta$, $\mathbf{q}_2 = \mathbf{v}/\beta$
$\mathbf{y} = H_{2,1}/\|\mathbf{b}\|\mathbf{e}_1$

$H_n = Q_n^T A Q_n$ è possibile calcolare con un minor costo computazionale grazie alla struttura tridiagonale che essa presenta, i suoi autovalori detti *valori di Ritz* e autovettori corrispondenti. È possibile dimostrare, come in [14], che gli autovalori estremali di H_n convergono agli autovalori estremali di A e anzi, al crescere del numero delle iterazioni, un numero crescente di autovalori delle due matrici tende a coincidere. Questo permette, data una matrice di grandi dimensioni, di ottenere delle approssimazioni sufficientemente accurate dei suoi autovalori estremali e questo può essere fatto con un costo computazionale relativamente basso essendo la matrice tridiagonale.
Abbiamo implementato l'iterazione di Lanczos per l'approssimazione di un certo numero k di autovalori di una matrice simmetrica A costruita tramite una trasformazione di similitudine in modo che i suoi autovalori fossero noti, per verificare l'efficienza del metodo. In particolare la matrice A simmetrica di ordine n, del problema test, avente autovalori $\lambda_i = i$ per $i = 1, \ldots, n$ è stata costruita come

$$A = QDQ^T$$

dove $D = \mathrm{diag}(1, 2, \ldots, n)$ e $Q \in \mathbb{R}^{n\times n}$ è una matrice ortogonale.

Algoritmo 18 Metodo di Lanczos per sistemi lineari: ciclo principale

k=1
repeat
  $k = k + 1$, $\mathbf{y0} = \mathbf{y}$
  $h_{k-1,k} = \beta, \quad \mathbf{v} = A\mathbf{q}_k$
  if mgs *// ortogonalizzazione tramite MGS*
    $\mathbf{v} = \mathbf{v} - \beta\mathbf{q}_{k-1}, \quad \alpha = \mathbf{q}_k^T\mathbf{v}, \quad \mathbf{v} = \mathbf{v} - \alpha\mathbf{q}_k$
  else *// ortogonalizzazione tramite CGS*
    $\alpha = \mathbf{q}_k^T\mathbf{v}, \quad \mathbf{v} = \mathbf{v} - \alpha\mathbf{q}_k - \beta\mathbf{q}_{k-1}$
  end if
  if reorth *// riortogonalizzazione delle colonne di Q*
    for $j = 1, \ldots, k$
      $\mathbf{v} = \mathbf{v} - (\mathbf{v}^T\mathbf{q}_j)\mathbf{q}_j$
    end for
  end if
  $\beta = \|\mathbf{v}\|$ *// costruzione degli elementi sottodiagonale di H*
  $h_{kk} = \alpha$ *// posizionamento degli elementi diagonali di H*
  if $\beta < \tau$ **then** uscita per breakdown **end**
  $\mathbf{q}_{k+1} = \mathbf{v}/\beta$, $h_{k+1,k} = \beta$ *// costruzione di Q e H*
  $\mathbf{y} = H_{k+1,k}/\|\mathbf{b}\|\mathbf{e}_1$
  $res_k = \left\|H_{k+1,k}\mathbf{y} - \|\mathbf{b}\|\mathbf{e}_1\right\|$
  if sol
    $e_k = \|Q_{n,k}\mathbf{y} - \mathbf{x}^*\|$
  end if
until $(\|\mathbf{y} - [y0; 0]\|) < tol \cdot \|\mathbf{y}\|$ or $k < N$
$\mathbf{x} = Q_{n,k}\mathbf{y}, \quad \mathbf{res} = \mathbf{res}_k$
$Q = Q_{n,k+1}$, $H = H_{k+1,k}$
Output: $H \in \mathbb{R}^{(k+1)\times k}$ matrice tridiagonale
  $Q \in \mathbb{R}^{n\times k}$ con colonne ortonormali che costituiscono
  una base per il sottospazio di Krylov $\mathcal{K}_k$
  soluzione $\mathbf{x}$

Il metodo di Lanczos implementato, fornirà le matrici tridiagonale e ortogonale della fattorizzazione di A e il vettore contenente i k autovalori di H_k che si verificherà, nel problema test, essere delle approssimazioni dei primi k autovalori di A. La mappa strutturale del metodo è illustrata nell' Algoritmo 19–20.

Algoritmo 19 Metodo di Lanczos per autovalori: inizializzazione

Input: $A \in \mathbb{R}^{n\times n}$ simmetrica,$\mathbf{b} \in \mathbb{R}^n$,
numero di autovalori di H che devono convergere k,
maxit, tolleranza τ, tolleranza di breakdown , *toll*, opzioni
$N = \min(n, maxit)$, $\mathbf{e} = \mathbf{0} \in \mathbb{R}^N$ // vettore degli errori
$Q = O \in \mathbb{R}^{n\times N}$, $H = O \in \mathbb{R}^{(N+1)\times N}$
$\mathbf{q}_1 = \mathbf{b}/\|\mathbf{b}\|$
$\mathbf{v} = A\mathbf{q}_1$
$\alpha = \mathbf{q}_1^T\mathbf{v}$, $h_{11} = \alpha$
$\mathbf{v} - \mathbf{v} - \alpha\mathbf{q}_1$
$\beta = \|\mathbf{v}\|, \quad h_{21} = \beta, \quad \mathbf{q}_2 = \mathbf{v}/\beta$

3.5 Conseguenze del breakdown

Tornando alla definizione di sottospazio di Krylov osserviamo che se i vettori $A^k\mathbf{b}$ con $k = 0, 1, \ldots, n-1$ risultassero essere tutti indipendenti, si avrebbe che

$$\mathcal{K}_n \equiv \mathbb{R}^n$$

e quindi l'algoritmo considerato terminerebbe dopo n passi.
Se non si avesse questa indipendenza lineare potrebbe accadere che, per un certo $l < n$, il vettore $A^l\mathbf{b}$ risulti essere dipendente dai precedenti e quindi esisterebbero degli scalari $\alpha_i, i = 0, 1, \ldots, l$ non tutti nulli, tali che

$$\sum_{i=0}^{l} \alpha_i A^i\mathbf{b} = 0.$$

Se $\alpha_0 \neq 0$, è possibile esplicitare il vettore $\mathbf{b}$ nella precedente e ottenere quindi

$$A\mathbf{x}^* = \mathbf{b} = -\frac{1}{\alpha_0}\sum_{i=0}^{l} \alpha_i A^i\mathbf{b} = A\Big(-\frac{1}{\alpha_0}\sum_{i=0}^{l} \alpha_i A^{i-1}\mathbf{b}\Big),$$

Algoritmo 20 Metodo di Lanczos per autovalori: ciclo principale

$j = 1$
lam=inf
repeat
  lam1=lam
  $h_{j-1,j} = \beta$
  $\mathbf{v} = A\mathbf{q}_j$
  if mgs *// ortogonalizzazione tramite MGS*
    $\mathbf{v} = \mathbf{v} - \beta\mathbf{q}_{j-1}$
    $\alpha = \mathbf{q}_j^T\mathbf{v}$
    $\mathbf{v} = \mathbf{v} - \alpha\mathbf{q}_j$
  else *// ortogonalizzazione tramite CGS*
    $\alpha = \mathbf{q}_j^T\mathbf{v}$
    $\mathbf{v} = \mathbf{v} - \alpha\mathbf{q}_j - \beta\mathbf{q}_{j-1}$
  end if
  if reorth *// riortogonalizzazione delle colonne di Q*
    for $i = 1, \ldots, j$
      $\mathbf{v} = \mathbf{v} - (\mathbf{v}^T\mathbf{q}_i)\mathbf{q}_i$
    end for
  end if
  $\beta = \|\mathbf{v}\|$ *// costruzione degli elementi sottodiagonali di H*
  if $\beta < \tau$ **then** uscita per breakdown **end**
  $\mathbf{q}_{j+1} = \mathbf{v}/\beta$
  $h_{jj} = \alpha$ *// costruzione della diagonale principale di A*
  $h_{j+1,j} = \beta$
  lam=`eig`$(H_{k,k})$ *// calcolo dei k autovalori di H*
until $\mathbf{e} <$ *toll* or $j < N + 1$
Output: $H \in \mathbb{R}^{(k+1)\times k}$ matrice tridiagonale
  $Q \in \mathbb{R}^{n\times k}$ con colonne ortonormali che costituiscono
  una base per il sottospazio di Krylov $\mathcal{K}_k$

dove $\mathbf{x}^*$ denota la soluzione esatta del sistema lineare che quindi sarà data

da

$$\mathbf{x}^* = -\frac{1}{\alpha_0}\sum_{i=0}^{l}\alpha_i A^{i-1}\mathbf{b}.$$

il che mostra che il metodo terminerebbe in l passi. Il fatto che tali metodi terminano in al più n passi, potrebbe far pensare che essi rientrino tra i metodi diretti. Vengono in realtà considerati iterativi, o meglio pseudo-iterativi, perché nel caso in cui vengano precondizionati possono condurre ad un'approssimazione sufficientemente accurata della soluzione in un numero di iterazioni molto inferiore alla dimensione n e questo è ovviamente utile nel caso di sistemi lineari di dimensioni molto grandi.
Quindi in base a quanto detto il verificarsi di un *breakdown* ad un certo passo $j < n$ sia nel metodo GMRES che nel metodo MINRES, indica che il vettore $A\mathbf{q}_j$ è linearmente dipendente dai precedenti e pertanto il sottospazio di Krylov $\mathcal{K}_j$ conterrà la soluzione esatta del sistema lineare $A\mathbf{x} = \mathbf{b}$ [10].
Considerando invece l'applicazione dei metodi di Arnoldi e di Lanczos al calcolo degli autovalori di una matrice, il verificarsi di un breakdown ad un certo passo j denota l'individuazione di un autospazio invariante V_j di dimensione j e significa inoltre che il vettore iniziale $\mathbf{b}$ è combinazione lineare di un numero finito di autovettori. Quindi nel caso si verifichi un breakdown non è possibile determinare altri autovalori oltre quelli già determinati sino al passo j che risultano essere esatti e non delle approssimazioni dei primi j autovalori di A.
Potrebbe essere però necessario continuare il calcolo per determinare gli altri autovalori; in tal caso occorre effettuare un *restart* del metodo partendo da un vettore che sia ortogonale al sottospazio generato dai vettori di V_j già determinati. Tale vettore una volta normalizzato, verrà aggiunto a quelli di V_j prima di procedere col calcolo delle approssimazionni dei restanti autovalori. Un'applicazione del metodo di Arnoldi con restart al calcolo degli pseudospettri si può trovare in [1].

3.6 Il metodo di Golub-Kahan

La bidiagonalizzazione di Golub-Kahan consiste nella riduzione di una matrice non simmetrica $A \in \mathbb{R}^{m\times n}$, $m \geq n$, in una matrice bidiagonale di dimensione inferiore. La fattorizzazione che si ottiene grazie a questo metodo è alla base del metodo LSQR per la risoluzione di un problema ai minimi quadrati, che non è altro che l'applicazione dell'iterazione di Golub-Kahan alla risoluzione di un sistema lineare. Un'altra sua importante applicazione

è il calcolo delle approssimazioni dei valori singolari della matrice A come vedremo in seguito.
Data quindi una matrice $A \in \mathbb{R}^{m\times n}$ non simmetrica e un vettore iniziale $\mathbf{b} \in \mathbb{R}^m$, l'applicazione di k passi del metodo consente di ottenere le decomposizioni

$$AV_k = U_{k+1}B_{k+1,k} \qquad A^TU_k = V_kB_k^T$$

dove le matrici $U_{k+1} = [\mathbf{u}_1 \quad \cdots \quad \mathbf{u}_{k+1}] \in \mathbb{R}^{m\times(k+1)}$ e $V_k = [\mathbf{v}_1 \quad \cdots \quad \mathbf{v}_k] \in \mathbb{R}^{n\times k}$ hanno colonne ortonormali e $U_k = [\mathbf{u}_1 \quad \cdots \quad \mathbf{u}_k] \in \mathbb{R}^{n\times k}$ dove $\mathbf{u}_1 = \frac{\mathbf{b}}{\|\mathbf{b}\|}$. Le colonne di V_k costituiscono una base per il sottospazio di Krylov $\mathcal{K}_k$ mentre la matrice

$$B_{k+1,k} = \begin{bmatrix} \rho_1 & & & & \\ \sigma_1 & \rho_2 & & & \\ & \sigma_3 & \rho_3 & & \\ & & \ddots & \ddots & \\ & & & \sigma_k & \rho_k \\ & & & & \sigma_{k+1} \end{bmatrix} \in \mathbb{R}^{(k+1)\times k}$$

è una matrice bidiagonale inferiore le cui entrate ρ_i e σ_i sono tutte positive. Vediamo come ottenere questa riduzione di A in forma bidiagonale, dato il vettore $\mathbf{b} \in \mathbb{R}^m$. Al passo iniziale si normalizza il vettore dato $\mathbf{b}$, ponendo

$$\mathbf{u}_1 = \frac{\mathbf{b}}{\|\mathbf{b}\|}.$$

Si calcola poi $\tilde{\mathbf{v}}_\mathbf{1} = A^T\mathbf{u}_1 = \rho_1\mathbf{v}_1$, da cui ponendo $\rho_1 = \|\tilde{\mathbf{v}}_\mathbf{1}\|$, si ottiene

$$\mathbf{v}_1 = \frac{\tilde{\mathbf{v}}_\mathbf{1}}{\rho_1}.$$

Dalla relazione

$$A\mathbf{v}_1 = \rho_1\mathbf{u}_1 + \sigma_1\mathbf{u}_2,$$

per determinare $\mathbf{u}_2$ poniamo

$$\tilde{\mathbf{u}}_2 = A\mathbf{v}_1 - \rho_1\mathbf{u}_1, \quad \sigma_1 = \|\tilde{\mathbf{u}}_2\|$$

da cui si ottiene quindi

$$\mathbf{u}_2 = \frac{\tilde{\mathbf{u}}_2}{\sigma_1}.$$

Al generico passo i l'iterazione di Golub-Kahan assume la forma

$$\tilde{\mathbf{v}}_i = A^T\mathbf{u}_i - \sigma_{i-1}\mathbf{v}_{i-1}, \quad \mathbf{v}_i = \frac{\tilde{\mathbf{v}}_i}{\rho_i},$$

$$\tilde{\mathbf{u}}_{i+1} = A\mathbf{v}_i - \rho_i \mathbf{u}_i, \quad \mathbf{u}_{i+1} = \frac{\tilde{\mathbf{u}}_{i+1}}{\sigma_i},$$

dove si ha

$$\rho_i = \|\tilde{\mathbf{v}}_i\|, \qquad \sigma_i = \|\tilde{\mathbf{u}}_{i+1}\|.$$

L'implementazione del metodo di bidiagonalizzazione di Golub-Kahan può essere schematizzata come nell'Algoritmo 21–22.

Algoritmo 21 Metodo di Golub-Kahan: inizializzazione

1: **Input:** $A \in \mathbb{R}^{m\times n}$,$\mathbf{b} \in \mathbb{R}^m$,
2: numero di iterazioni k, tolleranza τ, opzioni
3: $V = O \in \mathbf{R}^{n\times k}$, $U = O \in \mathbb{R}^{m\times(k+1)}$, $B = O \in \mathbb{R}^{(k+1)\times k}$
4: $\mathbf{u}_1 = \mathbf{b}/\|\mathbf{b}\|$
5: $\mathbf{v} = A^T\mathbf{u}_1$
6: $\rho = \|\mathbf{v}\|$
7: $b_{11} = \rho$
8: $\mathbf{v}_1 = \mathbf{v}/\rho$
9: $\mathbf{u} = A\mathbf{v}_1 - \rho\mathbf{u}_1$
10: $\sigma = \|\mathbf{u}\|, \quad b_{21} = \sigma$
11: $\mathbf{u}_2 = \mathbf{u}/\sigma$

3.6.1 LSQR

L'iterazione di Golub-Kahan, come osservato, applicata alla risoluzione di un sistema lineare, conduce al metodo LSQR per la risoluzione di un problema ai minimi quadrati. Tramite il procedimento di bidiagonalizzazione si determina una base ortonormale per il sottospazio di Krylov $\mathcal{K}_l$, costituita dalle colonne della matrice $V_l \in \mathbb{R}^{n\times l}$ e quindi un qualsiasi vettore di $\mathcal{K}_l$ può essere espresso come combinazione lineare dei vettori di tale base nella forma $V_l\mathbf{y}$, essendo $\mathbf{y}$ un generico vettore di $\mathbb{R}^l$. Il metodo LSQR, consiste quindi nell'approssimare la soluzione di un sistema lineare $A\mathbf{x} = \mathbf{b}$ col vettore $\mathbf{x}^{(l)}$ che minimizzi la norma-2 del residuo al passo l. Quindi si ha un problema ai minimi quadrati rappresentato da

$$\mathbf{x}^{(l)} = \arg\min_{\mathbf{x}\in\mathcal{K}_l} \|\mathbf{b} - A\mathbf{x}\|. \tag{3.10}$$

Quindi il metodo LSQR proietta un problema ai minimi quadrati $\min_{\mathbf{x}\in\mathbb{R}^l} \|\mathbf{b}-A\mathbf{x}\|$ nel sottospazio di Krylov con indice l risolvendo quindi il problema (3.10)

Algoritmo 22 Metodo di Golub-Kahan: ciclo principale

for $j = 2, \ldots, k$
  $\mathbf{v} = A^T \mathbf{u}_j - \sigma \mathbf{v}_{j-1}$
  if reorth
    if mgs // riortogonalizzazione delle colonne di V tramite MGS
      for $i = 1, \ldots, j-1$, $\mathbf{v} = \mathbf{v} - (\mathbf{v}^T \mathbf{v}_i)\mathbf{v}_i$, **end**
    else // riortogonalizzazione delle colonne di V tramite CGS
      for $i = 1, \ldots, j-1$, $\mathbf{v} = \mathbf{v} - \mathbf{v}_{j-1}(\mathbf{v}_{j-1}^T \mathbf{v})$, **end**
    end if
  end if
  $\rho = \|\mathbf{v}\|$
  if $\rho < \tau$ **then** uscita per breakdown **end** // *controllo breakdown*
  $\mathbf{v}_j = \mathbf{v}/\rho, \quad \mathbf{u} = A\mathbf{v}_j - \rho \mathbf{u}_j$
  if reorth
    if mgs // riortogonalizzazione delle colonne di U tramite MGS
      for $i = 1, \ldots, j$, $\mathbf{u} = \mathbf{u} - (\mathbf{u}T\mathbf{u}_i)\mathbf{u}_i$, **end**
    else // riortogonalizzazione delle colonne di V tramite CGS
      for $i = 1, \ldots, j$, $\mathbf{u} = \mathbf{u} - \mathbf{u}_i(\mathbf{u}_i^T \mathbf{u})$, **end**
    end if
  end if
  $\sigma = \|\mathbf{u}\|$
  if $\sigma < \tau$ **then** uscita per breakdown **end** // *controllo breakdown*
  $\mathbf{u}_{j+1} = \mathbf{u}/\sigma, \quad b_{jj} = \rho, \quad b_{j+1,j} = \sigma$
end for
Output: $B \in \mathbb{R}^{(k+1)\times k}$ matrice bidiagonale
  $U \in \mathbb{R}^{m\times(k+1)}$
  $V \in \mathbb{R}^{n\times k}$ con colonne ortonormali che costituiscono
  una base per il sottospazio di Krylov $\mathcal{K}_k$

calcolando la soluzione $\mathbf{y}^{(l)}$ del problema di minimizzazione

$$\min_{\mathbf{y}\in\mathbb{R}^l} \left\| B_{l+1,l}\mathbf{y} - \|\mathbf{b}\|\mathbf{e}_1 \right\|,$$

e calcolando successivamente il vettore $\mathbf{x}^{(l)}$ che approssima la soluzione esatta del sistema come $\mathbf{x}^{(l)} = V_l \mathbf{y}^{(l)}$.
Lo schema del metodo LSQR implementato per la risoluzione di un sistema lineare $A\mathbf{x} = \mathbf{b}$ è illustrato nell' Algoritmo 23–24. Nel problema test considerato si è presa in esame una matrice A rettangolare $m \times n$, con $m \geq n$ come matrice dei coefficienti di un sistema lineare noto per valutare l'efficienza del metodo.

Algoritmo 23 Metodo LSQR: inizializzazione

Input: $A \in \mathbb{R}^{m\times n}$, $\mathbf{x}^* = \mathbf{1} \in R^n$ // soluzione esatta,
$\mathbf{b} = A\mathbf{x}^* \in \mathbb{R}^m$ // termine noto del sistema,
tolleranza τ, opzioni
numero massimo di iterazioni $N = \min(m, n)$
$\mathbf{x} = O \in \mathbf{R}^n$
if sol
  $\mathbf{e} = \mathbf{0} \in \mathbb{R}^N$ // vettore degli errori
end if
$\mathbf{res} = \mathbf{0} \in \mathbb{R}^N$ // vettore dei residui
$V = O \in \mathbf{R}^{n\times N}$, $U = O \in \mathbb{R}^{m\times(N+1)}$, $B = O \in \mathbb{R}^{(N+1)\times N}$
$\mathbf{u}_1 = \mathbf{b}/\|\mathbf{b}\|$
$\mathbf{v} = A^T\mathbf{u}_1$, $\rho = \|\mathbf{v}\|$, $b_{11} = \rho$
$\mathbf{v}_1 = \mathbf{v}/\rho$
$\mathbf{u} = A\mathbf{v}_1 - \rho\mathbf{u}_1$, $\sigma = \|\mathbf{u}\|$, $b_{21} = \sigma$
$\mathbf{u}_2 = \mathbf{u}/\sigma$
$\mathbf{y} = B_{2,1}/\|\mathbf{b}\|\mathbf{e}_1$

3.6.2 Applicazione al calcolo dei valori singolari

Come si è visto nel teorema (1.17) la SVD di una matrice $A \in \mathbb{R}^{m\times n}$ nella sua definizione formale $A = U\Sigma V^T$, è legata alla decomposizione spettrale della matrice A^TA data da $V\Sigma^T\Sigma V^T$.
Da un punto di vista operativo, la SVD di A si potrebbe calcolare tramite i seguenti passi:

- costruire la matrice A^TA (detta *matrice di covarianza* di A);

Algoritmo 24 Metodo LSQR: ciclo principale

19: $k = 1$
20: **repeat**
21: $k = k + 1, \quad \mathbf{y0} = \mathbf{y}, \quad \mathbf{v} = A^T\mathbf{u}_k - \sigma\mathbf{v}_{k-1}$
22: **if** reorth
23: **if** mgs // riortogonalizzazione delle colonne di V tramite MGS
24: **for** $i = 1, \ldots, k-1$, $\mathbf{v} = \mathbf{v} - (\mathbf{v}^T\mathbf{v}_k)\mathbf{v}_k$, **end**
25: **else** // riortogonalizzazione delle colonne di V tramite CGS
26: $\mathbf{v} = \mathbf{v} - V_{n,k-1}(V_{n,k-1}^T\mathbf{v})$
27: **end if**
28: **end if**
29: $\rho = \|\mathbf{v}\|$
30: **if** $\rho < \tau$ **then** uscita per breakdown **end** // *controllo breakdown*
31: $\mathbf{v}_k = \mathbf{v}/\rho$, $\mathbf{u} = A\mathbf{v}_k - \rho\mathbf{u}_k$
32: **if** reorth
33: **if** mgs // riortogonalizzazione delle colonne di U tramite MGS
34: **for** $i = 1, \ldots, k$, $\mathbf{u} = \mathbf{u} - (\mathbf{u}^T\mathbf{u}_i)\mathbf{u}_i$, **end**
35: **else** // riortogonalizzazione delle colonne di V tramite CGS
36: $\mathbf{u} = \mathbf{u} - U_{m,k}(U_{m,k}^T\mathbf{u})$
37: **end if**
38: **end if**
39: $\sigma = \|\mathbf{u}\|$
40: **if** $\sigma < \tau$ **then** uscita per breakdown **end** // *controllo breakdown*
41: $\mathbf{u}_{k+1} = \mathbf{u}/\sigma, \quad b_{kk} = \rho, \quad b_{k+1,k} = \sigma$
42: $\mathbf{y} = B_{k+1,k}/\|\mathbf{b}\|\mathbf{e}_1, \quad res_k = \left\|B_{k+1,k}\mathbf{y} - \|\mathbf{b}\|\mathbf{e}_1\right\|$
43: **if** sol
44: $e_k = \|V_{n,k}\mathbf{y} - \mathbf{x}^*\|$
45: **end if**
46: **until** $(\|\mathbf{y} - [y0; 0]\|) < tol \cdot \|\mathbf{y}\|$ or $k < N$
47: **Output:** $B \in \mathbb{R}^{(k+1)\times k}$ matrice bidiagonale
48: $U \in \mathbb{R}^{m\times(k+1)}$, $V \in \mathbb{R}^{n\times k}$, soluzione $\mathbf{x}$

- calcolare la decomposizione spettrale $A^TA = VDV^T$;
- formare la matrice diagonale $m \times n$ Σ i cui elementi siano le radici quadrate degli elementi della matrice diagonale D, che risultano essere appunto gli autovalori di A^TA;
- risolvere il sistema $U\Sigma = AV$ determinando la matrice ortogonale U (utilizzando ad esempio il metodo QR).

Questo procedimento è tuttavia instabile in quanto riduce un problema ai valori singolari, ad un problema agli autovalori che potrebbe essere più sensibile agli errori di perturbazione come si può vedere in [14].
Un metodo più stabile per determinate la SVD di una matrice A $m \times n$ è quello di ridurla inizialmente ad una forma bidiagonale utilizzando ad esempio il metodo di Golub-Kahan e quindi considerare la fattorizzazione

$$U_1^T A V_1 = B$$

dove $U_1 \in \mathbb{R}^{m\times(k+1)}$ e $V_1 \in \mathbb{R}^{n\times k}$ sono matrici con colonne ortogonali e $B \in \mathbb{R}^{(k+1)\times k}$ è bidiagonale, con $k < n$. Essendo la matrice B bidiagonale la sua SVD $B = U_2\Sigma V_2^T$ è più facilmente calcolabile e calcolando i prodotti $U = U_1U_2$, $V = V_1V_2$ è possibile determinare la SVD della matrice A che sarò data da

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

Quindi utilizzando la bidiagonalizzazione di Golub-Kahan è possibile approssimare i valori singolari di una matrice A rettangolare. L'algoritmo implementato in MATLAB, schematizzato nell'Algoritmo 25, utilizza l'iterazione di Golub-Kahan descritta nell'Algoritmo 21–22 per approssimare i valori singolari di una matrice rettangolare di ordine $m \times n$ con $m \geq n$. Nel problema test considerato si costruisce una matrice A di quest'ordine, tramite trasformazioni di similitudine, in modo che essa abbia valori singolari noti così da verificare l'efficienza del metodo. Per il calcolo dei valori singolari della matrice bidiagonale B generata dall'iterazione di Golub-Kahan, si è utilizzata la funzione `svd` predefinita in MATLAB che fornisce il vettore dei valori singolari della matrice a cui è applicata. Si è poi effettuato un controllo sull'errore di approssimazione per misurare l'efficienza del metodo applicato che si è dimostrata buona.

Algoritmo 25 GK per l'approssimazione dei valori singolari

Input: $A \in \mathbb{R}^{m\times n}$ con valori singolari noti, $\mathbf{b} \in \mathbb{R}^m$
numero massimo di iterazioni N, tolleranza τ, opzioni
k numero valori singolari per il confronto
$\mathbf{e} = \mathbf{0} \in \mathbb{R}^N$ // vettore degli errori
$V = O \in \mathbf{R}^{n\times N}$, $U = O \in \mathbb{R}^{m\times(N+1)}$, $B = O \in \mathbb{R}^{(N+1)\times N}$
sv=inf // *inizializziamo il vettore che conterrà i valori singolari di B*
costruzione delle prime due colonne di U, prime colonne di V e di B
$j = 1$
repeat
  $j = j + 1$
  costruzione delle matrici U,V,B come nell'Algoritmo 21
  calcolo dei valori singolari di B posti nel vettore **sv**
  in ordine decrescente
  si considerano almeno k valori singolari per il confronto
  calcolo dell'errore di approssimazione
until errore$<$ *tol* or $j < N + 1$
Output: $B \in \mathbb{R}^{(k+1)\times k}$ matrice bidiagonale
$U \in \mathbb{R}^{m\times(k+1)}$
$V \in \mathbb{R}^{n\times k}$ con colonne ortonormali che costituiscono
una base per il sottospazio di Krylov $\mathcal{K}_k$
vettore sv delle approssimazioni dei valori singolari di A

Capitolo 4

Test numerici

In questo capitolo vengono trattati alcuni esperimenti numerici effettuati per testare l'efficienza dei metodi di Arnoldi e Lanczos nell'approssimazione degli autovalori di matrici non Hermitiane e Hermitiane, rispettivamente, e del metodo di Golub-Kahan nell'approssimazione dei valori singolari di matrici quadrate e rettangolari.

4.1 Metodo di Arnoldi per l'approssimazione degli autovalori

L'applicazione del metodo di Arnoldi all'approssimazione degli autovalori di una matrice non Hermitiana è schematizzata nell'Algoritmo 15.
Per valutare l'efficienza del metodo sono stati effettuati dei test considerando delle matrici con autovalori noti. In particolare si è presa in esame una matrice simmetrica di dimensione n ottenuta tramite trasformazioni di similitudine del tipo

$$A = HDH^T$$

dove la matrice D viene costruita partendo da una matrice diagonale con elementi reali $1, \ldots, n$, così da ottenere una matrice A con autovalori reali noti. Si è effettuato un test considerando anche una matrice M_2 non simmetrica di ordine n con autovalori complessi ottenuta come la precedente ma considerando la matrice diagonale D costruita inserendo dei blocchi diagonali 2×2 con autovalori complessi noti.
Abbiamo effettuato dei test confrontando l'Algoritmo 15 con la funzione `eig` predefinita in MATLAB per il calcolo degli di autovalori, fissando l'attenzione sugli $n_\lambda = 6$ autovalori di massimo modulo, considerando matrici di dimensioni differenti da $n = 200$ a $n = 2000$ con passo di 200. In particolare

si osserva che nel metodo di Arnoldi da noi implementato per l'approssimazione di n_λ autovalori di una matrice delle due suddette tipologie, viene dosata l'accuratezza dell'approssimazione con la tolleranza del criterio di stop utilizzato.
Nella Tabella 4.1 viene mostrato il confronto tra l'Algoritmo 15 e la funzione `eig` di Matlab, considerando la matrice test A descritta in precedenza di dimensione n crescente, per l'approssimazione di 6 autovalori, considerando che l'iterazione di Arnoldi si ferma quando tali autovalori vengono calcolati con una precisione fissata pari a 10^{-2}. Si osserva in particolare che è stato possibile ottenere una precisione inferiore alla tolleranza fissata, oltre che una maggiore velocità del metodo rispetto alla funzione `eig` di MATLAB. Nella prima colonna della tabella si considera la dimensione crescente della matrice test considerata, nelle colonne 2-3 sono presenti i tempi di calcolo espressi in secondi delle approssimazioni dei 6 autovalori più grandi della matrice test effettuate con la funzione predefinita di MATLAB e con l'iterazione di Arnoldi. Nella colonna 4 è contenuto il numero delle iterazioni effettuate dal metodo, mentre nelle ultime due colonne sono presenti gli errori di approssimazione di `eig` e del metodo di Arnoldi da noi implementato. Si osserva in particolare che, fissata come criterio di stop una tolleranza di 10^{-2}, è stato possibile ottenere un errore di approssimazione inferiore ad essa cosa che denota l'efficienza del metodo.

Tabella 4.1: *Matrici non simmetriche con autovalori complessi, con tolleranza del metodo di Arnoldi pari a* 10^{-2}.

Dimensione	$time_{\text{EIG}}$	$time_{\text{Arn}}$	iterazioni	Err_{EIG}	Err_{Arn}
200	0.0923	0.1294	56	4.64e-15	8.74e-05
400	0.3565	0.2989	81	3.58e-15	1.86e-04
600	0.8729	0.5268	90	6.86e-15	1.72e-03
800	1.6369	0.9810	110	9.12e-15	6.01e-05
1000	2.5802	1.7760	134	1.03e-14	4.11e-05
1200	3.7340	1.8037	131	6.27e-15	4.83e-05
1400	5.3971	3.8964	168	5.54e-15	3.41e-05
1600	7.3521	1.9671	128	7.40e-15	6.90e-04
1800	9.6863	5.2364	180	1.16e-14	4.56e-05
2000	12.9823	3.2790	151	1.18e-14	5.05e-05

Nella Tabella 4.2 viene illustrato il confronto tra l'iterazione di Arnoldi e la funzione `eig`, considerando però una tolleranza fissata a 10^{-4}.

Come si può osservare nell'ultima colonna anche in questo caso, è stato possibile ottenere un errore di approssimazione inferiore a tale tolleranza in media dell'ordine di 10^{-7}.
Nella Figura 4.1 viene illustrato l'andamento del tempo di calcolo nel caso delle approssimazioni dei 6 autovalori di modulo maggiore di una matrice test non simmetrica con autovalori complessi al crescere della dimensione considerando una tolleranza dell'ordine di 10^{-2}, mentre nella Figura 4.2 viene mostrato tale confronto del tempo di esecuzione al crescere della dimensione nel caso di una tolleranza del criterio di stop dell'ordine di 10^{-4}. In entrambi i casi si può osservare che il tempo di esecuzione del metodo di Arnoldi per il calcolo delle approssimazioni degli autovalori, è inferiore ad `eig` come anche mostrato nelle Tabelle 4.1-4.2, e che tale vantaggio cresce al crescere della dimensione.

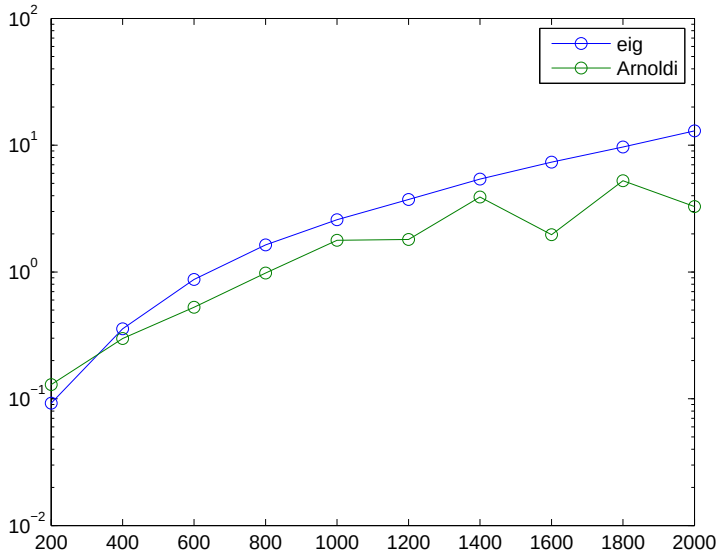

Figura 4.1: *Confronto tra Arnoldi e* `eig` *sul tempo di calcolo, tolleranza* 10^{-2}.

Per meglio illustrare i vantaggi in termini di tempo di elaborazione, nella Figura 4.3 viene mostrato l'andamento del tempo di esecuzione del metodo di Arnoldi e `eig` al crescere della dimensione per l'approssimazione dei 6 autovalori più grandi in modulo di una matrice proveniente dalla discretizzazione di una PDE di Poisson ottenuta tramite la funzione `gallery` di MATLAB. Poichè tale matrice risulta simmetrica, è stato necessario renderla non simmetrica utilizzando delle funzioni predefinite al fine di rendere giustificata

Tabella 4.2: *Matrici non simmetriche con autovalori complessi, con tolleranza del metodo di Arnoldi pari a* 10^{-4}.

Dimensione	$time_{\text{EIG}}$	$time_{\text{Arn}}$	iterazioni	Err_{EIG}	Err_{Arn}
200	0.0752	0.2006	73	4.45e-15	8.10e-07
400	0.2763	0.5385	95	7.46e-15	4.78e-07
600	0.8730	1.3626	127	4.39e-15	4.86e-07
800	1.5075	1.7768	135	5.70e-15	5.50e-07
1000	1.6623	4.3761	175	1.01e-14	3.53e-07
1200	3.8319	4.6039	179	1.46e-14	3.71e-07
1400	5.5389	4.6002	189	1.08e-14	3.18e-07
1600	6.8323	7.1819	201	2.25e-14	2.24e-07
1800	10.3864	9.5274	223	6.33e-15	2.16e-07
2000	12.3993	8.7636	211	1.31e-14	2.44e-07

l'applicazione del metodo di Arnoldi. Come viene illustrato nel grafico, il tempo di esecuzione dell'iterazione di Arnoldi è, nel caso particolare di questa matrice, estremamente inferiore a quello impiegato da `eig`. Il calcolo con quest'ultima funzione risulta impossibile per grandi dimensioni, pertanto è stato utilizzato solo per $n \leq 3000$.

4.2 Metodo di Lanczos per l'approssimazione degli autovalori

Nell'Algoritmo 19–20, come si è visto, è schematizzato il metodo di Lanczos per il calcolo degli autovalori di una matrice Hermitiana. Per valutare l'efficienza del metodo, sono stati effettuati dei test considerando matrici simmetiche con autovalori noti.
In particolare sono stati fatti dei test considerando una matrice simmetrica con autovalori reali $\lambda_j = j$ con $j = 1, \ldots, n$. Tale matrice test è stata costruita sempre utilizzando delle trasformazioni di similitudine, ossia come

$$A = HDH^T$$

dove $D = \text{diag}(1, \ldots, n)$ è una matrice diagonale le cui entrate sono appunto gli autovalori della matrice test considerata e H è una matrice elementare di Householder che è ortogonale oltre che simmetrica.
Anche nel caso dell'iterazione di Lanczos sono stati effettuati dei test confrontando l'Algoritmo 19-20 con la funzione `eig` predefinita in MATLAB per

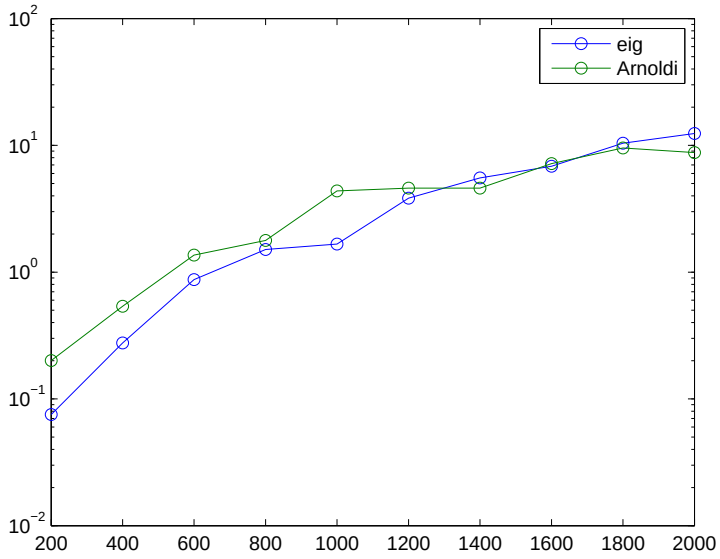

Figura 4.2: *Confronto tra Arnoldi e* `eig` *sul tempo di calcolo, tolleranza* 10^{-4}.

il calcolo di un certo numero di autovalori che abbiamo fissato sempre pari a 6, considerando matrici simmetriche della suddetta tipologia di dimensioni crescenti. In particolare si osserva che nel metodo di Lanczos da noi implementato per l'approssimazione dei $n_\lambda = 6$ autovalori estremali della matrice test, l'accuratezza dell'approssimazione viene dosata con la tolleranza del criterio di stop utilizzato. Nella Tabella 4.3 viene mostrato il confronto tra l'Algoritmo 19-20 e la funzione `eig` di Matlab, considerando matrici test simmetriche di dimensioni da 200 a 2000 (con passo di 200), per l'approssimazione dei 6 autovalori più grandi in modulo. In particolare tale tabella effettua questo confronto considerando una tolleranza del criterio di stop dell'ordine di 10^{-2} e si osserva che è stato ottenuto un errore di approssimazione per il metodo di Lanczos inferiore ad essa e dell'ordine di 10^{-5}.

Mentre nella Tabella 4.4 viene illustrato il confronto tra l'iterazione di Lanczos e `eig`, nel caso di una tolleranza fissata dell'ordine di 10^{-4} considerando che anche in tal caso è stato possibile ottenere un errore di approssimazione inferiore ad essa.

Come si evince dal confronto delle colonne 2-3 delle Tabelle 4.3-4.4 contenenti i tempi di calcolo in secondi, il metodo di Lanczos risulta più veloce della funzione `eig` in modo ancora più marcato di come osservato per l'iterazione di Arnoldi.

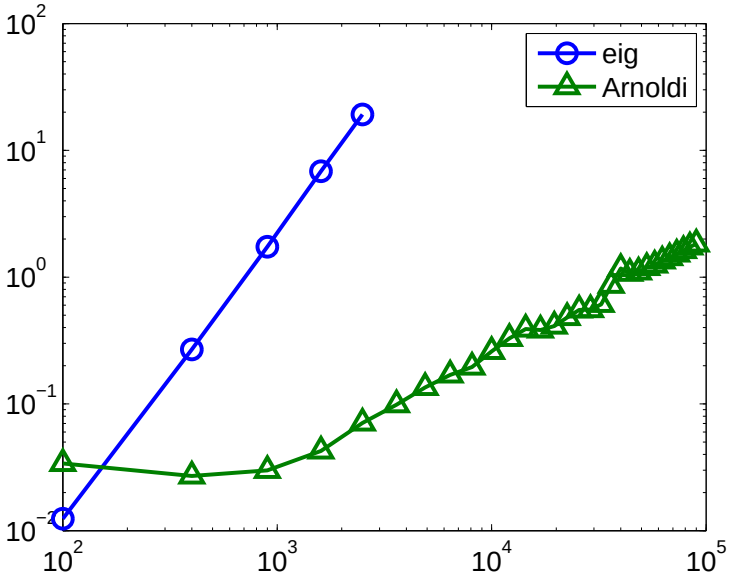

Figura 4.3: *Confronto tra Arnoldi e* `eig` *sul tempo di calcolo, tolleranza* 10^{-2}.

Nelle Figure 4.4 e 4.5, viene illustrato il grafico del tempo di calcolo al crescere della dimensione nel caso di Lanczos e `eig` come mostra la legenda considerando una tolleranza di 10^{-2} e di 10^{-4} rispettivamente.
Nella Figura 4.6 viene mostrato l'andamento del tempo di esecuzione del metodo di Lanczos e `eig` al crescere della dimensione per l'approssimazione dei 6 autovalori più grandi in modulo di una matrice simmetrica proveniente dalla discretizzazione di una PDE di Poisson ottenuta tramite la funzione `gallery` di MATLAB. Come viene illustrato nel grafico, il tempo di esecuzione del metodo di Lanczos per l'approssimazione degli autovalori è, anche nel caso particolare di questa matrice, molto inferiore a quello impiegato da `eig`.

4.3 Metodo di Golub-Kahan per l'approssimazione dei valori singolari

Nel Capitolo 3 si è visto che l'algoritmo implementato in MATLAB, schematizzato nell'Algoritmo 25, utilizza l'iterazione di Golub-Kahan descritta nell'Algoritmo 21–22 per approssimare i valori singolari di una matrice rettangolare di ordine $m \times n$ con $m \geq n$. Nei problemi test considerati viene

Tabella 4.3: *Matrici simmetriche con autovalori reali* $1, \cdots, n$*, con tolleranza del metodo di Lanczos pari a* 10^{-2}.

Dimensione	$time_{\text{EIG}}$	$time_{\text{Lan}}$	iterazioni	Err_{EIG}	Err_{Lan}
200	0.0680	0.0830	74	4.00e-15	9.59e-05
400	0.2872	0.1752	100	5.02e-15	1.13e-04
600	0.8823	0.3624	135	4.76e-15	3.47e-05
800	1.5703	0.3448	117	7.68e-15	1.25e-03
1000	2.5258	0.5113	133	3.76e-15	8.98e-04
1200	3.8045	0.9319	173	6.26e-15	5.28e-05
1400	5.3886	0.7144	133	8.46e-15	2.09e-03
1600	7.5382	1.2837	174	1.05e-14	3.00e-05
1800	10.1404	1.8267	197	1.34e-14	5.60e-05
2000	11.5494	1.7778	219	1.47e-14	3.29e-05

costruita una matrice M_r di quest'ordine, tramite trasformazioni di similitudine, in modo che essa abbia valori singolari noti così da verificare l'efficienza del metodo. In particolare tale matrice viene calcolata come

$$M_r = HSV^T$$

dove H è una matrice ortogonale di ordine m costruita tramite una trasformazione elementare di Householder, V è una matrice ortogonale con entrate casuali di ordine n mentre S è una matrice di ordine $m \times n$ costruita partendo da una matrice diagonale D le cui entrate $1, \ldots, n$ saranno appunto i valori singolari di M_r è aggiungendo ad essa una sottomatrice $(m-n) \times n$ di zeri. Si è poi applicato il metodo anche ad una matrice test quadrata $n \times n$ M_q con valori singolari noti costruita come

$$M_r = HDV^T,$$

dove H è una matrice elementare di Householder di dimensione n, V è una matrice ortogonale con entrate casuali di ordne n mentre $D = \text{diag}(1, \ldots, n)$ è una matrice diagonale le cui entrate saranno i valori singolari della matrice test considerata. Nella Tabella 4.5 viene effettuato un confronto tra il metodo di Golub-Kahan schematizzato dall'Algoritmo 25 e la funzione `svd` predefinita in MATLAB per il calcolo dei valori singolari di una matrice rettangolare di dimensioni crescenti con $m = 2n$. In particolare si è effettuato tale confronto per l'approssimazione di $n = 6$ valori singolari considerando una tolleranza del criterio di stop dell'ordine di 10^{-2}.

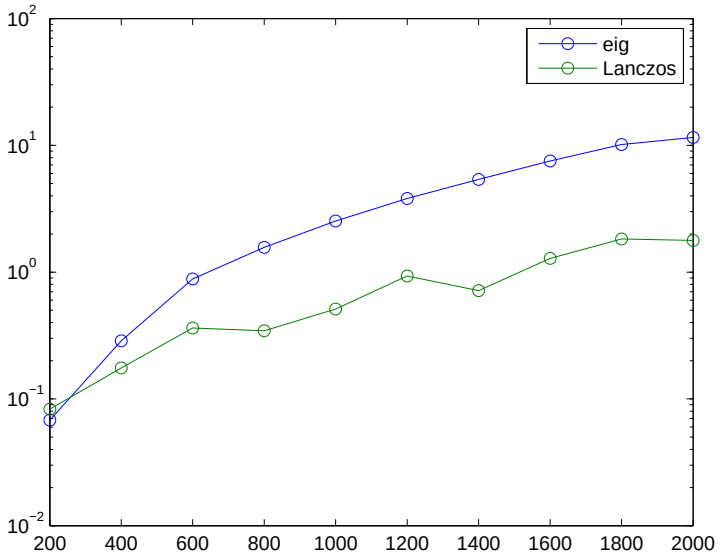

Figura 4.4: *Confronto tra Lanczos e* `eig` *sul tempo di calcolo, tolleranza* 10^{-2}

Si è osservato che al crescere della dimensione, il tempo impiegato da Golub-Kahan tende ad essere la metà del tempo di calcolo impiegato per l'approssimazione dei valori singolari dalla funzione `svd`.
Nella Tabella 4.6 viene illustrato il confronto tra il metodo di Golub-Kahan e la funzione `svd` per l'approssimazione dei 6 valori singolari più grandi di una matrice quadrata di dimensione n crescente da 1000 a 5000 (con passo 500) considerando una tolleranza del criterio di stop di 10^{-2}. Si è osservato che al crescere della dimensione il tempo di esecuzione (colonne 2-3) del metodo di Golub-Kahan è risultato inferiore a quello impiegato dalla funzione `svd` di MATLAB e si è ottenuto un errore di approssimazione inferiore a tale tolleranza e dell'ordine di 10^{-5}.

Tabella 4.4: *Matrici simmetriche con autovalori reali* $1, \cdots, n$*, con tolleranza del metodo di Lanczos pari a* 10^{-4}.

Dimensione	$time_{\text{EIG}}$	$time_{\text{Lan}}$	iterazioni	Err_{EIG}	Err_{Lan}
200	0.0594	0.1173	77	3.46e-15	7.34e-07
400	0.2960	0.2168	113	7.00e-15	5.87e-07
600	0.8606	0.3340	130	1.35e-14	3.60e-07
800	1.5186	0.6486	167	8.54e-15	4.23e-07
1000	2.3867	0.8607	178	1.40e-14	2.67e-07
1200	2.8831	0.5596	204	6.46e-15	1.68e-07
1400	2.0085	0.6469	225	5.86e-15	1.51e-07
1600	6.0314	2.0539	233	1.68e-14	2.12e-07
1800	9.9862	2.4745	242	7.09e-15	2.34e-07
2000	13.1322	2.6309	235	1.12e-14	8.33e-07

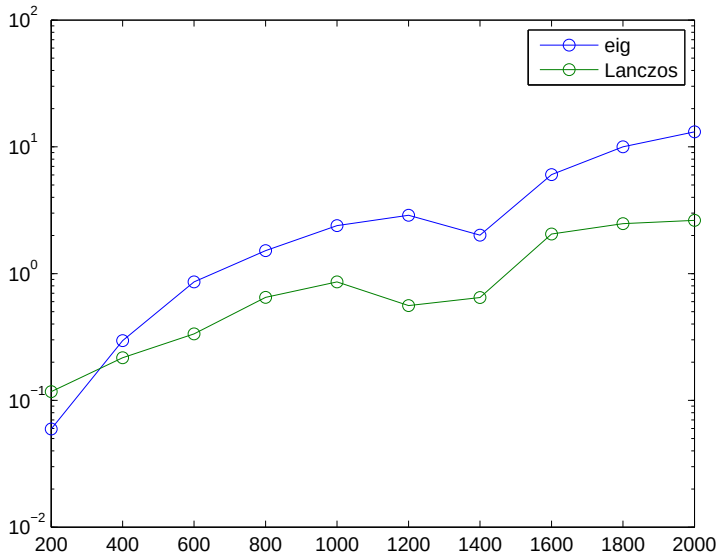

Figura 4.5: *Confronto tra Lanczos e* `eig` *sul tempo di calcolo, tolleranza* 10^{-4}

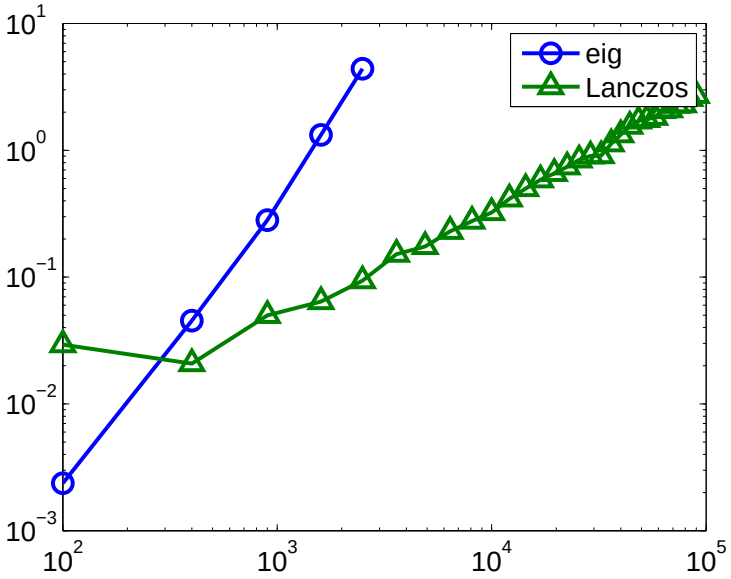

Figura 4.6: *Confronto tra Lanczos e* `eig` *sul tempo di calcolo, tolleranza* 10^{-2}.

Tabella 4.5: *Matrici rettangolari di dimensione* $m \times n$ $(m = 2n)$ *con valori singolari noti* $1, \ldots, n$, *tolleranza del metodo di GK pari a* 10^{-2}.

Dimensione	$time_{\text{SVD}}$	$time_{\text{GK}}$	iterazioni	Err_{SVD}	Err_{GK}
2000×1000	0.6644	5.9175	99	7.98e-15	2.33e-05
3000×1500	4.3028	16.0196	137	9.27e-15	1.87e-05
4000×2000	5.0419	8.0787	151	1.99e-14	1.61e-05
5000×2500	12.0011	11.0421	165	9.30e-15	1.82e-05
6000×3000	17.1090	13.4700	168	1.91e-14	1.35e-05
7000×3500	41.5371	29.8381	220	1.31e-14	2.08e-05
8000×4000	41.6930	20.8040	172	1.16e-14	4.60e-04
9000×4500	73.5532	57.3135	228	2.18e-14	2.23e-04
10000×5000	131.3536	66.8703	242	2.07e-14	1.92e-05

Tabella 4.6: *Matrici quadrate di dimensione n con valori singolari noti* $1, \ldots, n$*, tolleranza del metodo di GK pari a* 10^{-2}.

Dimensione	$time_{\mathrm{SVD}}$	$time_{\mathrm{GK}}$	iterazioni	Err_{SVD}	Err_{GK}
1000	0.9785	11.1380	99	4.10e-15	1.00e-03
1500	3.3245	15.4080	131	1.44e-14	4.23e-05
2000	5.4978	17.3525	140	1.14e-14	1.56e-05
2500	11.1279	24.2908	173	2.10e-14	2.37e-05
3000	21.4690	22.5339	153	2.12e-14	3.62e-04
3500	35.0332	35.3650	219	1.14e-14	9.40e-06
4000	44.5406	26.8428	212	1.18e-14	4.15e-05
4500	66.7437	41.4424	225	1.48e-14	2.92e-05
5000	80.8711	41.0184	240	3.31e-14	1.80e-05

Capitolo 5

Conclusioni

In questa tesi è stato affrontato il tema dei problemi agli autovalori e in particolare sono stati considerati i metodi di proiezione in sottospazi di Krylov per la loro risoluzione.
Il lavoro di questa tesi ha avuto come scopo l'implementazione dei metodi di Arnoldi e Lanczos per l'approssimazione degli autovalori estremali di matrici di grandi dimensioni e del metodo di bidiagonalizzazione di Golub-Kahan per l'approssimazione dei valori singolari estremali di matrici quadrate e rettangolari. Per testare l'efficienza di tali metodi implementati in MATLAB , nel capitolo 4 sono stati illustrati alcuni esperimenti numerici.
Dal confronto degli algoritmi suddetti con le funzioni `eig` e `svd` predefinite in MATLAB, si è ottenuto un tempo di risoluzione significativamente inferiore anche per dimensioni dell'ordine di 2000 e 10000 rispettivamente considerando che inoltre gli errori di approssimazione sono risultati inferiori alla tolleranza del criterio di stop denotando così l'efficienza dei metodi considerati.
Possibili sviluppi per un lavoro futuro riguardano un'analisi più approfondita dei metodi considerati nel caso di uno shift dello spettro e per il calcolo degli pseudospettri.

Bibliografia

[1] R Astudillo and Z Castillo. Computing pseudospectra using block implicitly restarted Arnoldi iteration. *Mathematical and Computer Modelling*, 57(9):2149–2157, 2013.

[2] Sergey Brin and Lawrence Page. Reprint of: The anatomy of a large-scale hypertextual web search engine. *Computer networks*, 56(18):3825–3833, 2012.

[3] L. Eldén. *Matrix Methods in Data Mining and Pattern Recognition*, volume 4 of *Fundamentals of Algorithms*. SIAM, Philadelphia, 2007.

[4] M. Embree and L. N. Trefethen. PSEUDOSPECTRA Gateway. `http://www.comlab.ox.ac.uk/pseudospectra`, 2002.

[5] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems*. SIAM, Philadelphia, 1998.

[6] S. H. Lui. Computation of pseudospectra by continuation. *SIAM Journal of Scientific Computing*, 18:565–573, 1997.

[7] D. Mezher and B. Philippe. PAT - a reliable path-following algorithm. *Numerical Algorithms*, 29:131–152, 2002.

[8] M. S. M.S. Pranic, L. Reichel, G. Rodriguez, Z. Wang, and X. Yu. A rational Arnoldi process with applications. Submitted, 2015.

[9] S. C. Reddy, P. J. Schmid, and D. S. Henningson. Pseudospectra of the Orr-Sommerfeld operator. *SIAM Journal on Applied Mathematics*, 53:15–47, 1993.

[10] G. Rodriguez. *Algoritmi Numerici*. Pitagora Editrice Bologna, Bologna, 2008.

[11] Y. Saad. *Numerical Methods for Large Eingenvalue Problems*. SIAM, Minneapolis, revised edition edition, 2011.

[12] K.-C. Toh and L. N. Trefethen. Calculation of pseudospectra by the Arnoldi iteration. *SIAM Journal of Scientific Computing*, 17:1–15, 1996.

[13] L. N. Trefethen. *Pseudospectra of matrices.* Oxford University, Computing Laboratory Numerical Analysis Group, 1991.

[14] L. N. Trefethen and D. Bau. *Numerical Linear Algebra.* SIAM, Philadelphia, 1997.

[15] H. van der Vost. *Iterative Krylov Method for Large Linear Systems.* The Cambridge University Press, New York, 2003.

[16] M. P. H. Wolff. Discrete approximation of unbounded operators and approximation of their spectra. *Journal of Approximation Theory*, 113:229–244, 2002.