UNIVERSITÀ DI PISA

Dipartimento di Matematica
Corso di Laurea in Matematica

Calcolo Scientifico

Il Metodo delle Iterazioni dei Sottospazi

Docenti:
Prof.ssa L. Aceto
Prof. D.A. Bini

Presentata da:
Ivan Bioli

Anno Accademico 2020/2021

Sommario

Nel seguente seminario presenteremo il Metodo delle Iterazioni dei Sottospazi per il calcolo delle p autocoppie dominanti di una matrice quadrata. Tale metodo viene applicato per il calcolo delle p autocoppie dominanti nel caso in cui p sia molto piccolo rispetto alla taglia della matrice. Nel seguito analizzeremo alcune varianti del Metodo che permettono di accelerarne la convergenza. Metteremo a confronto gli algoritmi proposti sia dal punto di vista teorico che sperimentale, al fine di valutarne l'efficienza.

1 Introduzione

1.1 Assunzioni e setting

Il *Metodo delle Iterazioni dei Sottospazi* può essere applicato a matrici qualsiasi, ma lo presenteremo nel caso specifico di matrici hermitiane, alle quali è possibile applicare una tecnica di accelerazione che migliora notevolmente la convergenza. Supponiamo dunque di avere $A \in \mathbb{C}^{n\times n}$ matrice simmetrica reale o hermitiana con autovalori $\lambda_1, \lambda_2, \ldots, \lambda_n$ ordinati in ordine decrescente di modulo e relativi autovettori $u_1, \ldots, u_n$, che possiamo assumere ortonormali. Sia p un intero $1 < p \ll n$, vogliamo calcolare i p autovalori dominanti nell'ipotesi che questi siano separati dal resto dello spettro:

$$|\lambda_1| \geq \cdots \geq |\lambda_p| > |\lambda_{p+1}| \geq \cdots \geq |\lambda_n| \tag{1}$$

Per calcolare i p autovalori più piccoli in modulo o i più vicini a una certa costante assegnata α, basta sostituire A con A^{-1} o $(A - \alpha I)^{-1}$ rispettivamente (nell'ipotesi che tali matrici siano effettivamente invertibili). Per questioni di efficienza e stabilità numerica inoltre, non si esegue il calcolo dell'inversa ma si risolvono sistemi lineari.

1.2 Definizioni e Notazioni

Per analizzare la convergenza del metodo abbiamo bisogno della nozione di angolo tra due sottospazi. Il caso che ci interessa è quello di sottospazi della stessa dimensione.

Definizione 1. *Siano $Q_1, Q_2 \in \mathbb{C}^{n\times q}$ matrici con colonne ortonormali, $Q_1^*Q_1 = Q_2^*Q_2 = I_q$, e sia $S_i = Im(Q_i)$. Definiamo l'angolo tra i sottospazi S_1, S_2 come $\angle(S_1, S_2) = \theta, 0 \leq \theta \leq \pi/2$ tale che $\sin\theta = \|Q_2Q_2^* - Q_1Q_1^*\|$*

Useremo inoltre la seguente comoda notazione:

Notazione. *Data $B \in \mathbb{K}^{m\times n}$ matrice a coefficienti nel campo $\mathbb{K}$ (nel nostro caso $\mathbb{K} = \mathbb{R}$ o $\mathbb{K} = \mathbb{C}$), indichiamo con $\mathcal{R}(B)$ lo Span delle colonne di B, cioè l'immagine di B*

2 Algoritmi

Presentiamo in questa sezione gli algoritmi che andremo a confrontare sia dal punto di vista teorico che sperimentale. Per le dimostrazioni non riportate si faccia riferimento a [1, 3, 4].

2.1 Metodo delle Iterazioni Ortogonali

Il primo algoritmo è quello delle **Iterazioni Ortogonali** nella versione standard, senza metodi che ne accelerino la convergenza e applicabile a qualsiasi matrice.

Algoritmo 1 Iterazioni Ortogonali

1: Sia $X \in \mathbb{C}^{n\times p}$ matrice con colonne ortonormali, $X^*X = I_p$
2: $X^{(0)} := X, k = 0$
3: **repeat**
4: $\quad k := k + 1$
5: $\quad Z^{(k)} := AX^{(k-1)}$
6: $\quad Z^{(k)} =: Q^{(k)}R^{(k)}$ /*Fattorizzazione QR di $Z^{(k)}$*/
7: $\quad X^{(k)} = Q^{(k)}$
8: $\quad \lambda^{(k)} := diag(R^{(k)})$
9: **until** $\|\lambda^{(k)} - \lambda^{(k-1)}\|/\|\lambda^{(k)}\| < tol$ **or** $k > itmax$

Si noti che, poiché la matrice $R^{(k)}$ è triangolare superiore, le prime j colonne di $X^{(k)}$ dipendono unicamente dalle prime j colonne di $X^{(k-1)}$. Dunque la $j-esima$ colonna influenza soltanto le colonne alla propria destra. In particolare se applicassimo l'Algoritmo 1 a partire da una matrice $\hat{X} \in \mathbb{C}^{n\times q}$ tale che $Xe_i = \hat{X}e_i$ per $i = 1, \ldots, j$ allora, per ogni k, avremmo che $X^{(k)}e_i = \hat{X}^{(k)}e_i$ per $i = 1, \ldots, j$. In particolare, questo ci dice che la prima colonna $X^{(k)}e_i$ sta eseguendo il *Metodo delle Potenze*.

Per quanto riguarda la convergenza di tale metodo valgono i seguenti risultati relativamente alla convergenza di autospazi e autovettori.

Teorema 1 (Convergenza dei sottospazi)**.** *Sia $U_p := [u_1, \ldots, u_p]$ la matrice formata dagli autovettori corrispondenti ai p autovalori dominanti $\lambda_1, \ldots, \lambda_p$ di A. Sia $X^{(0)} \in \mathbb{C}^{n\times p}$ tale che $U_p^*X^{(0)}$ sia non singolare. Allora, se $|\lambda_p| > |\lambda_{p+1}|$, il sottospazio $\mathcal{R}(X^{(k)})$ converge al sottospazio $\mathcal{R}(U_p)$ e vale:*

$$\tan\theta^{(k)} \leq \left|\frac{\lambda_{p+1}}{\lambda_p}\right|^k \tan\theta^{(0)}, \qquad \theta^{(k)} = \angle(\mathcal{R}(X^{(k)}), \mathcal{R}(U_p)) \tag{2}$$

Teorema 2. *Supponiamo che non solo $W_p := U_p^*X^{(0)}$ sia non singolare, ma che lo sia anche la matrice $W_q := U_q^*X^{(0)}$ $1 \leq q \leq p$. Supponiamo che $|\lambda_{q-1}| > |\lambda_q| \geq \cdots \geq |\lambda_p| > |\lambda_{p+1}|$. Allora:*

$$\sin\angle(\mathcal{R}([x_q^{(k)}, \ldots, x_p^{(k)}]), \mathcal{R}([u_q, \ldots, u_p])) \leq c \cdot max\left\{\left|\frac{\lambda_q}{\lambda_q - 1}\right|^k, \left|\frac{\lambda_{p+1}}{\lambda_p}\right|^k\right\} \tag{3}$$

Corollario 2.1. *Supponiamo che* $|\lambda_{j-1}| > |\lambda_j| > |\lambda_{j+1}|$ *e che* W_j *sia non singolare. Allora:*

$$\sin \angle(x_j^{(k)}, u_j) \leq c \cdot max \left\{ \left| \frac{\lambda_j}{\lambda_{j-1}} \right|^k, \left| \frac{\lambda_{j+1}}{\lambda_j} \right|^k \right\} \tag{4}$$

Relativamente alla convergenza degli autovalori, sotto ipotesi leggermente più restrittive sullo spettro di A, vale il seguente teorema:

Teorema 3 (Convergenza degli autovalori)**.** *Supponiamo che i p autovalori dominanti di A siano tutti distinti tra loro e dai rimanenti autovalori,*

$$|\lambda_1| > \cdots > |\lambda_p| > |\lambda_{p+1}|$$

Allora per gli elementi diagonali della matrice $R^{(k)}$ *vale la stima:*

$$|r_{ii}^{(k)} - \lambda_i| = O\left(\left| \frac{\lambda_{i+1}}{\lambda_i} \right|^k + \left| \frac{\lambda_i}{\lambda_{i-1}} \right|^k \right) \tag{5}$$

dove per $i = 1$ *il secondo termine non è presente.*

La velocità di convergenza, sia per gli autovalori che per gli autovettori, dipende dai rapporti $|\lambda_{i+1}/\lambda_i|$ ed è dunque potenzialmente molto lenta se sono presenti due autovalori successivi con moduli che differiscono di poco. La convergenza non è invece neanche assicurata se non vi è stretta separazione di tutti gli autovalori da λ_1 a λ_{p+1}.

2.2 Metodo delle Iterazioni Ortogonali con accelerazione

Sfruttando una generalizzazione del quoziente di Rayleigh si ottiene il cosiddetto metodo delle **Iterazioni Ortogonali con accelerazione di Rayleigh-Ritz**, che presentiamo in due varianti. Per semplicità di esposizione supporremo che la matrice A sia simmetrica reale o Hermitiana definita positiva.

Quello che ci porta a pensare che la velocità di convergenza possa essere notevolmente migliorata con un opportuno cambio di base è il seguente teorema.

Teorema 4. *Sia* $X^{(0)}$ *come nel Teorema 1 e siano* $u_i, 1 \leq i \leq p$ *gli autovettori corrispondenti ai p autovalori dominanti* $\lambda_1, \ldots, \lambda_p$ *di A. Allora:*

$$\min_{x \in \mathcal{R}(X^{(k)})} \sin \angle(u_i, x) \leq c \left(\frac{\lambda_{p+1}}{\lambda_i} \right)^k \tag{6}$$

Dimostrazione. Si può facilmente mostrare per induzione che:

$$A^k X^{(0)} = X^{(k)} R, \qquad R = R^{(k)} R^{(k-1)} \cdots R^{(1)}$$

Sia inoltre $U \in \mathbb{C}^{n \times n}$ matrice unitaria tale che $U^* A U = diag(\lambda_1, \ldots, \lambda_n) =: \Lambda$. Possiamo partizionare le matrici Λ e $U^* X^{(k)} =: \hat{X}^{(k)}$ come:

$$\Lambda = diag(\Lambda_1, \Lambda_2), \quad \hat{X}^{(k)} = \begin{bmatrix} \hat{X}_1^{(k)} \\ \hat{X}_2^{(k)} \end{bmatrix} \qquad \Lambda_1, \hat{X}_1^{(k)} \in \mathbb{C}^{p \times p}$$

e si ottiene

$$X^{(k)}R = A^kX^{(0)} = U\Lambda^kU^*X^{(0)} = U\begin{bmatrix} \Lambda_1^k\hat{X}_1^{(0)} \\ \Lambda_2^k\hat{X}_2^{(0)} \end{bmatrix} = U\begin{bmatrix} I_p \\ S^{(k)} \end{bmatrix}\Lambda_1^k\hat{X}_1^{(0)} \qquad (7)$$

dove $S^{(k)} = \Lambda_2^k\hat{X}_2^{(0)}\hat{X}_1^{(0)^{-1}}\Lambda_1^{-k} \in \mathbb{C}^{(n-p)\times p}$. Si ricava dunque facilmente che:

$$s_{ij}^{(k)} = s_{ij}\left(\frac{\lambda_{p+i}}{\lambda_j}\right)^k, \quad s_{ij} := s_{ij}^{(0)} \qquad 1 \le i \le n-p, \quad 1 \le j \le p$$

Abbiamo dunque ottenuto che $\mathcal{R}(X^{(k)}) = \mathcal{R}\left(U\begin{bmatrix} I_p \\ S^{(k)} \end{bmatrix}\right)$ e pertanto dalle formule ricavate in precedenza per $1 \le i \le p$:

$$\begin{aligned}
\min_{x\in\mathcal{R}(X^{(k)})} \sin\angle(u_i, x) &\le \sin\angle\left(u_i, U\begin{pmatrix} I_p \\ S^{(k)} \end{pmatrix} e_i\right) = \\
&= \left\| (I - u_iu_i^*)U \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \\ s_{1i}(\lambda_{p+1}/\lambda_i)^k \\ \vdots \\ s_{n-p,i}(\lambda_n/\lambda_i)^k \end{bmatrix} \right\| \Big/ \left\| \begin{pmatrix} I_p \\ S^{(k)} \end{pmatrix} e_i \right\| \le \\
&\le \left\| (I - u_iu_i^*)\left(u_i + \sum_{j=p+1}^{n} s_{j-p,i}\left(\frac{\lambda_j}{\lambda_i}\right)^k u_j\right) \right\| = \\
&= \sqrt{\sum_{j=1}^{n-p} s_{ji}^2 \frac{\lambda_{p+j}^{2k}}{\lambda_i^{2k}}} \le \left(\frac{\lambda_{p+1}}{\lambda_i}\right)^k \sqrt{\sum_{j=1}^{n-p} s_{ji}^2}
\end{aligned} \qquad (8)$$

□

Osserviamo però che anche se il sottospazio generato dalle colonne della matrice di partenza $X^{(0)}$ coincidesse con il sottospazio generato dai primi p autovettori, gli elementi diagonali di $R^{(k)}$ non coinciderebbero con gli autovalori $\lambda_1, \ldots, \lambda_p$ e la velocità di convergenza sarebbe governata dai rapporti di cui sopra. Però gli autovalori della matrice

$$H^{(k)} := Q^{(k)^*}AQ^{(k)} \in \mathbb{C}^{p\times p} \qquad \textit{(quoziente di Rayleigh generalizzato)}$$

restrizione di A al sottospazio $\mathcal{R}(Q^{(k)})$, coinciderebbero proprio con $\lambda_1, \ldots, \lambda_p$ e il costo del loro calcolo sarebbe trascurabile dato che $p \ll n$. Si noti che la matrice $H^{(k)} := Q^{(k)^*}AQ^{(k)}$ è una sorta di quoziente di Rayleigh generalizzato dove il vettore unitario viene sostituito dalla matrice $Q^{(k)}$ con colonne ortonormali. Queste osservazioni suggeriscono la seguente variante.

Algoritmo 2 Iterazioni ortogonali con accelerazione di Rayleigh-Ritz, versione 1

Sia $X \in \mathbb{C}^{n\times p}$ matrice con colonne ortonormali, $X^*X = I_p$
$X^{(0)} := X, k = 0$
repeat
  $k := k + 1$
  $Z^{(k)} := AX^{(k-1)}$
  $Z^{(k)} =: Q^{(k)}R^{(k)}$ /*Fattorizzazione QR di $Z^{(k)}$*/
  $H^{(k)} := Q^{(k)^*}AQ^{(k)}$
  $H^{(k)} =: F^{(k)}D^{(k)}F^{(k)^*}$ /*Decomposizione spettrale di $H^{(k)} \in \mathbb{C}^{p\times p}$*/
  $X^{(k)} = Q^{(k)}F^{(k)}$
  $\lambda^{(k)} := diag(D^{(k)})$
until $\|\lambda^{(k)} - \lambda^{(k-1)}\|/\|\lambda^{(k)}\| < tol$ **or** $k > itmax$

Si osservi che le colonne della matrice $X^{(k)}$ generano, nei due metodi presentati, lo stesso sottospazio vettoriale, cioè quello generato dalle colonne di $A^kX^{(0)}$. La base però è diversa e questo è ciò che permette di accelerare notevolmente la convergenza.

Nota. *Le colonne $x_i^{(k)}$ della matrice $X^{(k)}$ sono chiamate **vettori di Ritz** e gli autovalori $d_1^{(k)} \leq \cdots \leq d_p^{(k)}$ sulla diagonale di $D^{(k)}$ sono chiamati **valori di Ritz**.*

Il calcolo di tutti gli autovalori e autovettori della matrice $H^{(k)}$ viene svolto tramite l'algoritmo QR per matrici simmetriche e il costo è trascurabile dato che la matrice ha taglia $p \ll n$. Il costo del calcolo del quoziente di Rayleigh generalizzato $H^{(k)} := Q^{(k)^*}AQ^{(k)}$ è però piuttosto alto poiché richiede il calcolo di p moltiplicazioni di A per un vettore, con un costo che è dunque doppio rispetto a quello del metodo senza accelerazione (per un maggiore approfondimento sul costo computazionale si veda la sezione dedicata).

Possiamo ovviare a questo problema con un astuto cambio di base. Cerchiamo di scrivere $X^{(k)}$ nella forma:

$$X^{(k)} = Z^{(k)}G^{(k)}, \qquad G^{(k)} \in \mathbb{C}^{p\times p} \text{ non singolare} \tag{9}$$

richiedendo che $X^{(k)}$ abbia colonne ortonormali

$$G^{(k)^*}Z^{(k)^*}Z^{(k)}G^{(k)} = I_p \tag{10}$$

Osserviamo inoltre che se A è invertibile (come nel caso di A definita positiva) e u è autovettore di A relativo all'autovalore λ, allora u è anche autovettore di A^{-1} relativo all'autovalore $1/\lambda$. Pertanto se $X^{(k)}$ avesse per colonne i p autovettori dominanti, posto $\Delta = diag(\lambda_1, \ldots, \lambda_p)$ si avrebbe:

$$A^{-1}X^{(k)} = \Delta^{-1} \qquad \text{e quindi} \qquad X^{(k)^*}A^{-2}X^{(k)} = \Delta^{-2}$$

Imponiamo dunque anche che:

$$\left(Z^{(k)}G^{(k)}\right)^* A^{-2}(Z^{(k)}G^{(k)}) = \Delta^{(k)^{-2}} = \text{diagonale} \tag{11}$$

Sfruttando la definizione di $Z^{(k)}$ e sostituendo nella (11)

$$\left(AX^{(k-1)}G^{(k)}\right)^* A^{-2}(AX^{(k-1)}G^{(k)}) = G^{(k)^*}G^{(k)} = \Delta^{(k)^{-2}} \tag{12}$$

Dunque la matrice $Y^{(k)} := G^{(k)}\Delta^{(k)}$ è ortogonale e dalle condizioni in (10) ricaviamo:

$$Y^{(k)^*} Z^{(k)^*} Z^{(k)} Y^{(k)} = \Delta^{(k)^2} \tag{13}$$

Pertanto le colonne di $Y^{(k)}$ formano una base ortonormale di autovettori per la matrice $\hat{H}^{(k)} := Z^{(k)^*} Z^{(k)}$ e $\Delta^{(k)^2}$ è la matrice diagonale con elementi diagonali uguali agli autovalori di $\hat{H}^{(k)}$. Calcolate queste due matrici possiamo ottenere

$$X^{(k)} = Z^{(k)} G^{(k)} = Z^{(k)} Y^{(k)} \Delta^{(k)^{-1}}$$

Queste osservazioni i suggeriscono una seconda variante.

Algoritmo 3 Iterazioni ortogonali con accelerazione di Rayleigh-Ritz, versione 2

Sia $X \in \mathbb{C}^{n\times p}$ matrice con colonne ortonormali, $X^*X = I_p$
$X^{(0)} := X, k = 0$
repeat
  $k := k + 1$
  $Z^{(k)} := AX^{(k-1)}$
  $\hat{H}^{(k)} := Z^{(k)^*} Z^{(k)}$
  $\hat{H}^{(k)} =: Y^{(k)} \Delta^{(k)^2} Y^{(k)*}$ /*Decomposizione spettrale di $\hat{H}^{(k)} \in \mathbb{C}^{p\times p}$*/
  $X^{(k)} = Z^{(k)} Y^{(k)} \Delta^{(k)^{-1}}$ /*$= Z^{(k)} G^{(k)}$*/
  $\lambda^{(k)} := diag(\Delta^{(k)})$
until $\|\lambda^{(k)} - \lambda^{(k-1)}\| / \|\lambda^{(k)}\| < tol$ **or** $k > itmax$

Una terza variante è la cosiddetta subroutine `ritzritz`, programmata da Rutishauser [5]. Ci limitiamo a presentarla, per una giustificazione e per un'analisi teorica della velocità di convergenza si vedano [5, 4].

Algoritmo 4 Iterazioni ortogonali con accelerazione di Rayleigh-Ritz, versione 3 (`ritzritz`)

Sia $X \in \mathbb{C}^{n\times p}$ matrice con colonne ortonormali, $X^*X = I_p$
$X^{(0)} := X, k = 0$
repeat
  $k := k + 1$
  $Z^{(k)} := AX^{(k-1)}$
  $Z^{(k)} =: Q^{(k)} R^{(k)}$ /*Fattorizzazione QR di $Z^{(k)}$*/
  $H^{(k)} := R^{(k)} R^{(k)^*}$
  $H^{(k)} =: P^{(k)} \Delta^{(k)^2} P^{(k)^*}$ /*Decomposizione spettrale di $H^{(k)}$*/
  $X^{(k)} = Q^{(k)} P^{(k)}$
  $\lambda^{(k)} := diag(\Delta^{(k)})$
until $\|\lambda^{(k)} - \lambda^{(k-1)}\| / \|\lambda^{(k)}\| < tol$ **or** $k > itmax$

2.2.1 Analisi teorica della velocità di convergenza

Mostreremo adesso che i vettori di Ritz convergono agli autovettori con la velocità di convergenza che ci lascia sperare il Teorema 4. Per procedere alla dimostrazione abbiamo bisogno di un lemma preliminare.

Lemma 5. *Sia y un vettore unitario e $\theta \in \mathbb{C}$. Sia λ l'autovalore di A più vicino a θ e sia u il corrispondente autovettore, che supponiamo anch'esso unitario. Sia $\gamma = \min_{\lambda_i(A)\neq\lambda}|\lambda_i(A) - \theta|$ e sia $\psi = \angle(y, u)$. Allora:*

$$\sin\psi \leq \frac{\|r(y)\|}{\gamma} := \frac{\|Ay - \theta y\|}{\gamma}$$

con $r(y) := Ay - \theta y$ che svolge il ruolo di residuo

Dimostrazione. Scriviamo $y = u\cos\psi + v\sin\psi$ con $v\perp u, \|v\| = 1$. Allora:

$$\begin{aligned} Ay - \theta y &= (A - \theta I)u\cos\psi + (A - \theta I)u\sin\psi = \\ &= (\lambda - \theta)u\cos\psi + (A - \theta I)v\sin\psi \end{aligned}$$

Sfruttando il fatto che i due addendi sono ortogonali, cioè $u^*(A-\theta I)v = (\lambda-\theta)u^*v = 0$, si ottiene:

$$\begin{aligned} \|r(y)\|^2 &= (\lambda - \theta)^2\cos^2\psi + \|(A - \theta I)v\|^2\sin^2\psi \geq \\ &\geq \gamma^2\|v\|^2\sin^2\psi = \gamma^2\sin^2\psi \end{aligned}$$

dove si è sfruttato che v non ha componente lungo u autovettore relativo a λ. □

Teorema 6. *Supponiamo valgano le ipotesi del Teorema 1. Sia $x_i^{(k)} = X^{(k)}e_i$ l'i-esimo vettore di Ritz come calcolato dall'Algoritmo 3 e sia $y_i^{(k)} = U\begin{pmatrix} I_p \\ S^{(k)} \end{pmatrix} e_i$ (cfr. dimostrazione del Teorema 4). Allora vale la seguente disuguaglianza:*

$$\sin\angle(x_i^{(k)}, y_i^{(k)}) \leq c\left(\frac{\lambda_{p+1}}{\lambda i}\right)^k, \qquad 1 \leq i \leq p$$

Dimostrazione. Abbiamo già osservato nella dimostrazione del Teorema 4 che le colonne di $U\begin{pmatrix} I_p \\ S^{(k)} \end{pmatrix}$ formano una base di $\mathcal{R}(X^{(k)})$. Pertanto possiamo scrivere

$$x_i^{(k)} = U\begin{pmatrix} I_p \\ S^{(k)} \end{pmatrix} t_i, \qquad t_i \in \mathbb{C}^p.$$

Per determinare le colonne di $Y^{(k)}$ e gli elementi diagonali di $\Delta^{(k)^2}$ invece del problema agli autovalori

$$X^{(k-1)^*}A^2X^{(k-1)}y = \hat{H}^{(k)}y = \mu^2 y$$

nella 'base' ortonormale formata dalle colonne di $X^{(k)}$ consideriamo l'equivalente problema agli autovalori:

$$\begin{bmatrix} I_p & S^{(k)^*} \end{bmatrix} U^*A^2U \begin{bmatrix} I_p \\ S^{(k)} \end{bmatrix} t = \mu^2 \begin{bmatrix} I_p & S^{(k)^*} \end{bmatrix} \begin{bmatrix} I_p \\ S^{(k)} \end{bmatrix} t \tag{14}$$

Si può mostrare che t_i tale che $x_i^{(k)} = U\begin{pmatrix} I_p \\ S^{(k)} \end{pmatrix} t_i$ risolve il precedente problema agli autovalori, diciamo μ_i il corrispondente autovalore.

Sia (μ, t) una autocoppia di (14). Allora abbiamo:

$$\begin{aligned}
0 &= \begin{bmatrix} I_p & S^{(k)^*} \end{bmatrix} U^* A^2 U \begin{bmatrix} I_p \\ S^{(k)} \end{bmatrix} t - \mu^2 \begin{bmatrix} I_p & S^{(k)^*} \end{bmatrix} \begin{bmatrix} I_p \\ S^{(k)} \end{bmatrix} t = \\
&= \begin{bmatrix} I_p & S^{(k)} \end{bmatrix} \Lambda^2 \begin{bmatrix} I_p \\ S^{(k)^*} \end{bmatrix} t - \mu^2 \left(I_p + S^{(k)^*} S^{(k)}\right) = \\
&= \left(\Lambda_1^2 + S^{(k)^*} \Lambda_2^2 S^{(k)}\right) t - \mu^2 \left(I_p + S^{(k)^*} S^{(k)}\right) = \\
&= \left((\Lambda_1^2 - \mu^2 I) + S^{(k)^*} (\Lambda_2^2 - \mu^2 I) S^{(k)}\right) t = \\
&= \left((\Lambda_1^2 - \mu^2 I) + \Lambda_1^{-k} S^{(0)^*} \Lambda_2^k (\Lambda_2^2 - \mu^2 I) \Lambda_2^k S^{(0)} \Lambda_1^{-k}\right) t = \\
&= \left((\Lambda_1^2 - \mu^2 I) + \left(\frac{1}{\lambda_{p+1}} \Lambda_1\right)^{-k} H_k \left(\frac{1}{\lambda_{p+1}} \Lambda_1\right)^{-k}\right) t
\end{aligned} \tag{15}$$

dove

$$H_k = \lambda_{p+1}^{-2k} S^{(0)^*} \Lambda_2^k (\Lambda_2^2 - \mu^2 I) \Lambda_2^k S^{(0)}$$

Poiché il più grande autovalore di Λ_2 è λ_{p+1} la successione H_k è limitata, sia $\|H_k\| \leq c_1 \quad \forall k > 0$. Pertanto per $k \to \infty$:

$$\left(\left(\frac{1}{\lambda_{p+1}} \Lambda_1\right)^{-k} H_k \left(\frac{1}{\lambda_{p+1}} \Lambda_1\right)^{-k}\right) t \to 0$$

Possiamo interpretare l'equazione (15) come una perturbazione della matrice diagonale $\Lambda_2^2 - \mu^2 I$. Dunque per k sufficientemente grande (dipendente da i) c'è un μ_i^2 che è vicino a λ_i^2 e un t_i che è vicino a e_i. Supponiamo che k sia sufficientemente grande da avere

$$|\mu_i^2 - \lambda_i^2| \leq \rho := \frac{1}{2} \min_{\lambda_j \neq \lambda_i} |\lambda_i^2 - \lambda_j^2|$$

cosicché μ_i^2 è più vicino a λ_i^2 di ogni altro $\lambda_j^2, j \neq i$.

Consideriamo adesso la matrice con colonne ortonormali:

$$B = U \begin{pmatrix} I_p \\ S^{(k)^*} \end{pmatrix} \left(I_p + S^{(k)^*} S^{(k)}\right)^{-1/2}, \qquad B^* B = I_p$$

Se (μ_i^2, t_i) è una autocoppia di (15), allora $\left(\mu_i^2, \left(I_p + S^{(k)^*} S^{(k)}\right)^{1/2} t_i\right)$ è una autocoppia di:

$$B^* A^2 B t = \mu^2 t \tag{16}$$

Poiché per k sufficientemente grande (λ_i^2, e_i) è una buona approssimazione della autocoppia (μ_i^2, t_i) di (15), allora anche $\left(\lambda_i^2, \left(I_p + S^{(k)^*} S^{(k)}\right)^{1/2} e_i\right)$ è una buona approssimazione della autocoppia $\left(\mu_i^2, \left(I_p + S^{(k)^*} S^{(k)}\right)^{1/2} t_i\right)$ di (16). Adesso applichiamo il Lemma 5 alla matrice $B^* A^2 B$ con

$$\begin{aligned}
\theta &= \lambda_i^2 \\
y &= \left(I_p + S^{(k)^*} S^{(k)}\right)^{1/2} e_i \Big/ \left\| \left(I_p + S^{(k)^*} S^{(k)}\right)^{1/2} e_i \right\| \\
u &= \left(I_p + S^{(k)^*} S^{(k)}\right)^{1/2} t_i \Big/ \left\| \left(I_p + S^{(k)^*} S^{(k)}\right)^{1/2} t_i \right\|
\end{aligned}$$

Si osservi che per k sufficientemente grande, dato che μ_j si può fare vicino a piacere a λ_j, allora $\gamma := \min_{\mu_j \neq \mu_i} |\mu_j^2 - \lambda_i^2| \geq \rho$. Inoltre:

$$\left\| \left(I_p + S^{(k)^*} S^{(k)}\right)^{1/2} e_i \right\|^2 = e_i^* \left(I_p + S^{(k)^*} S^{(k)}\right) e_i \geq 1$$

e dunque abbiamo la stima:

$$\begin{aligned}
\|r(y)\| &= \left\| (B^* A^2 B - \lambda_i^2 I) \left(I_p + S^{(k)^*} S^{(k)}\right)^{1/2} e_i \right\| \Big/ \left\| \left(I_p + S^{(k)^*} S^{(k)}\right)^{1/2} e_i \right\| \leq \\
&\leq \left\| \left(I_p + S^{(k)^*} S^{(k)}\right)^{-1/2} \left(\begin{bmatrix} I_p & S^{(k)^*} \end{bmatrix} U^* A^2 U \begin{bmatrix} I_p \\ S^{(k)} \end{bmatrix} - \lambda_i^2 \left(I_p - S^{(k)^*} S^{(k)}\right) \right) \right\| \leq \\
&\leq \left\| \left(I_p + S^{(k)^*} S^{(k)}\right)^{-1/2} \right\| \left\| \left(\Lambda_1^2 - \lambda_i^2 I + \left(\lambda_{p+1}^{-1} \Lambda_1\right)^{-k} H_k \left(\lambda_{p+1}^{-1} \Lambda_1\right)^{-k} \right) e_i \right\| \leq \\
&\leq \left\| \left(\lambda_{p+1} \Lambda_1^{-1}\right)^k H_k \left(\lambda_{p+1} \Lambda_1^{-1}\right)^k e_i \right\| \leq \\
&\leq \left\| \lambda_{p+1} \Lambda_1^{-1} \right\|^k \|H_k\| \left\| \left(\lambda_{p+1} \Lambda_1^{-1}\right)^k e_i \right\| \leq c_1 \left(\frac{\lambda_{p+1}}{\lambda_p}\right)^k \left(\frac{\lambda_{p+1}}{\lambda_i}\right)^k \leq c_1 \left(\frac{\lambda_{p+1}}{\lambda_i}\right)^k
\end{aligned}$$

Dunque per il Lemma 5 vale, per k sufficientemente grande,

$$\begin{aligned}
\sin \angle(x_i^{(k)}, y_i^{(k)}) &\leq \sin \angle \left(B \left(I_p + S^{(k)^*} S^{(k)}\right)^{1/2} t_i, B \left(I_p + S^{(k)^*} S^{(k)}\right)^{1/2} e_i \right) = \\
&= \sin \angle \left(\left(I_p + S^{(k)^*} S^{(k)}\right)^{1/2} t_i, \left(I_p + S^{(k)^*} S^{(k)}\right)^{1/2} e_i \right) \leq \\
&\leq \frac{c_1}{\rho} \left(\frac{\lambda_{p+1}}{\lambda_i}\right)^k
\end{aligned} \tag{17}$$

da cui la tesi. □

Nella dimostrazione del Teorema 4 abbiamo mostrato

$$\sin \angle(u_i, y_i^{(k)}) \leq c_1 \left(\frac{\lambda_{p+1}}{\lambda i}\right)^k$$

e nel precedente teorema abbiamo mostrato

$$\sin \angle(x_i^{(k)}, y_i^{(k)}) \leq c_2 \left(\frac{\lambda_{p+1}}{\lambda i}\right)^k$$

Di conseguenza otteniamo che

$$\sin \angle(x_i^{(k)}, u_i) \leq c_3 \left(\frac{\lambda_{p+1}}{\lambda i}\right)^k \tag{18}$$

e cioè la convergenza dell'i-esimo vettore di Ritz all'i-esimo autovalore è lineare con fattore di convergenza $\gamma_i = \lambda_{p+1}/\lambda_i$. Si noti che il fattore di convergenza è notevolmente migliore rispetto a quello del metodo senza accelerazione, ma non solo: per avere la convergenza del metodo non è necessaria la stretta separazione di tutti $p+1$ autovettori dominanti, basta che siano separati il p-esimo e il successivo.

Per quanto riguarda la convergenza degli autovalori, si può mostrare che nel caso dell'Algoritmo 2 vale:

$$|\lambda_i^{(k)} - \lambda_i| = O\left(\left|\frac{\lambda_{p+1}}{\lambda_i}\right|^k\right) \tag{19}$$

La convergenza degli autovalori ha invece velocità doppia nel caso dell'Algoritmo 3 applicato ad A matrice simmetrica reale o hermitiana e definita positiva. Come approssimazione dell'autovalore $\lambda_j > 0$ possiamo considerare infatti la radice quadrata di:

$$\lambda_j^{(k+1)^2} = \frac{\|Ax_j^{(k)}\|^2}{\|x_j^{(k)}\|^2} = x_j^{(k)^*} A^2 x_j^{(k)}$$

Essendo $\|x_j^{(k)}\| = 1$ posto $\phi^{(k)} = \angle(x_j^{(k)}, u_j)$ possiamo scrivere $x_j^{(k)} = \cos\phi^{(k)}\, u_j + \sin\phi^{(k)}\, v$ con $v \perp u_j$, $\|v\| = 1$. Ma allora:

$$\begin{aligned}\lambda_j^{(k+1)^2} &= x_j^{(k)^*} A^2 x_j^{(k)} = \cos^2\phi^{(k)}\, u_j^* A^2 u_j + \sin^2\phi^{(k)}\, v^* A^2 v \leq \\ &\leq \cos^2\phi^{(k)}\, \lambda_j^2 + \sin^2\phi^{(k)}\, \|A\|_2^2 = \cos^2\phi^{(k)}\, \lambda_j^2 + \sin^2\phi^{(k)}\, \lambda_1^2 = \\ &= \lambda_j^2 + (\lambda_1^2 - \lambda_j^2)\sin^2\phi^{(k)}\end{aligned}$$

Dunque:

$$\begin{aligned}|\lambda_j^{(k+1)} - \lambda_j| &= \frac{\lambda_1^2 - \lambda_j^2}{|\lambda_j^{(k+1)} + \lambda_j|}\sin^2\phi^{(k)} = O(\sin^2\phi^{(k)}) = \\ &= O\left(\frac{\lambda_{p+1}}{\lambda_i}\right)^{2k}\end{aligned} \tag{20}$$

e cioè la velocità di convergenza è doppia rispetto a quella degli autovettori.

2.3 Analisi teorica del costo computazionale

Il costo computazionale dei precedenti algoritmi è fortemente influenzato dalla struttura della matrice A, che se sparsa permette di ridurre notevolmente il numero di operazioni aritmetiche necessarie. Nell'analisi del costo in termini di operazioni aritmetiche indicheremo dunque con OP il costo della moltiplicazione di un vettore per A. Il costo per passo di ciascuno degli algoritmi presentati è riassunto nelle seguenti tabelle, dove sono proposte anche alcune strategie per ridurre la complessità in spazio degli algoritmi.

Iterazione Algoritmo 1	
$Z^{(k)} := AX^{(k-1)}$	p chiamate di OP
$Z^{(k)} =: Q^{(k)}R^{(k)}$	$p(p+1)n$ operaz.
$X^{(k)} = Q^{(k)}$	0 operaz.

Iterazione Algoritmo 2	
$Z^{(k)} := AX^{(k-1)}$	p chiamate di OP, $Z^{(k)}$ sovrascrive $X^{(k)}$
$Z^{(k)} =: Q^{(k)}R^{(k)}$	$p(p+1)n$ operaz., $Q^{(k)}$ sovrascrive $Z^{(k)}$
$H^{(k)} := Q^{(k)^*}(AQ^{(k)})$	p chiamate di OP, $\frac{1}{2}p(p+1)n$ operaz.
$H^{(k)} =: F^{(k)}D^{(k)}F^{(k)^*}$	kp^3 operaz. (k dipende dal metodo)
$X^{(k)} = Q^{(k)}F^{(k)}$	kp^2n operaz. $X^{(k)}$ sovrascrive $Q^{(k)}$

Iterazione Algoritmo 3	
$Z^{(k)} := AX^{(k-1)}$	p chiamate di OP, $Z^{(k)}$ sovrascrive $X^{(k)}$
$\hat{H}^{(k)} := Z^{(k)^*} Z^{(k)}$	$\frac{1}{2}p(p+1)n$ operaz.
$\hat{H}^{(k)} =: Y^{(k)} \Delta^{(k)^2} Y^{(k)*}$	kp^3 operaz. (k dipende dal metodo)
$X^{(k)} = Z^{(k)} Y^{(k)} \Delta^{(k)^{-1}}$	$kp^2 n$ operaz. $X^{(k)}$ sovrascrive $Z^{(k)}$

Iterazione Algoritmo 4	
$Z^{(k)} := AX^{(k-1)}$	p chiamate di OP, $Z^{(k)}$ sovrascrive $X^{(k)}$
$Z^{(k)} =: Q^{(k)} R^{(k)}$	$p(p+1)n$ operaz., $Q^{(k)}$ sovrascrive $Z^{(k)}$
$H^{(k)} := R^{(k)} R^{(k)^*}$	$\frac{1}{3}p^3$ operaz.
$H^{(k)} =: P^{(k)} \Delta^{(k)^2} P^{(k)^*}$	kp^3 operaz. (k dipende dal metodo)
$X^{(k)} = Q^{(k)} P^{(k)}$	$kp^2 n$ operaz. $X^{(k)}$ sovrascrive $Q^{(k)}$

Si può osservare come il costo sia dominato, quantomeno per matrici non troppo sparse, dal numero di chiamate di OP. Pertanto ci aspettiamo che l'Algoritmo 2 risulti più lento nell'eseguire una singola iterazione rispetto agli altri algoritmi.

3 Sperimentazione numerica

Riportiamo le metodologie e i risultati della sperimentazione numerica effettuata per confrontare gli algoritmi proposti in precedenza. Per effettuare i test abbiamo usato MATLAB R2019b su PC con processore Intel® Core™ i7 Q720 1.60GHz, 8 GB di RAM e Windows 10 Home (64 bit).

3.1 Implementazione degli algoritmi in MATLAB®

Riportiamo nel seguito l'implementazione in MATLAB degli algoritmi usata per eseguire la sperimentazione numerica.

Listing 1: Algoritmo 1

```
function [X,lambda,it] = basic(A,p,X_0,itmax,tol)
%function [X,lambda,it] = basic(A,p,X_0,itmax,tol)
%Esegue il Metodo delle Iterazioni Ortogonali nella
   versione standard, determinando le p autocoppie
   dominanti della matrice A a partire dalla matrice con
   colonne ortonormali X_0.
%IN INPUT
%    - p: numero di autocoppie dominanti da calcolare
%    - X_0: matrice con colonne ortonormali di partenza
%    - itmax: numero massimo di iterazioni svolte
%    - tol: tolleranza per condizioni di terminazione
%IN OUTPUT
%    - X: approssimazioni degli autovettori
%    - lambda: approssimazioni degli autovalori
%    - it: numero di iterazioni effettivamente svolte

    X = X_0;
    n = size(X,1);
    lambda = zeros(p,1);
    for it = 1:itmax
        lambda_prec = lambda;              %precedente,
           necessario per la condizione di terminazione
        X = A*X;
        [X,R] = qr(X,0);
        lambda = diag(R);
        if (norm(lambda_prec-lambda)/norm(lambda) < tol)
            break;
        end
    end
end
```

Listing 2: Algoritmo 2

```
function [X,lambda,it] = acceleration_v1(A,p,X_0,itmax,
   tol)
%function [X,lambda,it] = acceleration_v1(A,p,X_0,itmax,
   tol)
%Esegue il Metodo delle Iterazioni Ortogonali con
   accelerazione di Rayleigh-Ritz nella prima versione,
   determinando le p autocoppie dominanti della matrice A
    a partire dalla matrice con colonne ortonormali X_0.
%IN INPUT
%    - p: numero di autocoppie dominanti da calcolare
%    - X_0: matrice con colonne ortonormali di partenza
%    - itmax: numero massimo di iterazioni svolte
%    - tol: tolleranza per condizioni di terminazione
%IN OUTPUT
%    - X: approssimazioni degli autovettori
%    - lambda: approssimazioni degli autovalori
%    - it: numero di iterazioni effettivamente svolte

    X = X_0;
    n = size(X,1);
    lambda = zeros(p,1);
    for it = 1:itmax
        lambda_prec = lambda;       %precedente,
           necessario per la condizione di terminazione
        X = A*X;
        [X,~] = qr(X,0);
        H = X'*A*X;
        [F,D] = eig(H);
        lambda = diag(D);
        X=X*F;
        if (norm(lambda_prec-lambda)/norm(lambda) < tol)
            break;
        end
    end
    [~, loc] = max (abs(X));
    linear_index = sub2ind(size(X),loc, 1:p);
    P = A*X;
    lambda = (P(linear_index)./X(linear_index))';
    [~,index] = sort(abs(lambda),'descend');
    lambda = lambda(index);
    X = X(:,index);
end
```

Listing 3: Algoritmo 3

```
function [X,lambda,it] = acceleration_v2(A,p,X_0,itmax,
   tol)
%function [X,lambda, it] = acceleration_v2(A,p,X_0,itmax,
   tol)
%Esegue il Metodo delle Iterazioni Ortogonali con
   accelerazione di Rayleigh-Ritz nella seconda versione,
    determinando le p autocoppie dominanti della matrice
   A a partire dalla matrice con colonne ortonormali X_0.
%IN INPUT
%    - p: numero di autocoppie dominanti da calcolare
%    - X_0: matrice con colonne ortonormali di partenza
%    - itmax: numero massimo di iterazioni svolte
%    - tol: tolleranza per condizioni di terminazione
%IN OUTPUT
%    - X: approssimazioni degli autovettori
%    - lambda: approssimazioni degli autovalori
%    - it: numero di iterazioni effettivamente svolte

    X = X_0;
    n = size(X,1);
    lambda = zeros(p,1);
    for it = 1:itmax
        lambda_prec = lambda;             %precedente,
           necessario per la condizione di terminazione
        X = A*X;
        H = X'*X;
        [Y,Delta] = eig(H);
        lambda = sqrt(diag(Delta));
        X=X*Y*diag(1./lambda);
        if (norm(lambda_prec-lambda)/norm(lambda) < tol)
            break;
        end
    end
    [~, loc] = max (abs(X));
    linear_index = sub2ind(size(X),loc, 1:p);
    P = A*X;
    lambda = (P(linear_index)./X(linear_index))';
    [~,index] = sort(abs(lambda),'descend');
    lambda = lambda(index);
    X = X(:,index);
end
```

Listing 4: Algoritmo 4

```
function [X,lambda,it] = ritzritz(A,p,X_0,itmax,tol)
%function [X,lambda, it] = ritzritz(A,p,X_0,itmax,tol)
%Esegue il Metodo delle Iterazioni Ortogonali con
   accelerazione di Rayleigh-Ritz usando l'algoritmo
   ritritz. Determina le p autocoppie dominanti della
   matrice A a partire dalla matrice con colonne
   ortonormali X_0.
%IN INPUT
%    - p: numero di autocoppie dominanti da calcolare
%    - X_0: matrice con colonne ortonormali di partenza
%    - itmax: numero massimo di iterazioni svolte
%    - tol: tolleranza per condizioni di terminazione
%IN OUTPUT
%    - X: approssimazioni degli autovettori
%    - lambda: approssimazioni degli autovalori
%    - it: numero di iterazioni effettivamente svolte

    X = X_0;
    n = size(X,1);
    lambda = zeros(p,1);
    for it = 1:itmax
        lambda_prec = lambda;            %precedente,
           necessario per la condizione di terminazione
        X = A*X;
        [X,R] = qr(X,0);
        H = R*R';
        [P,Delta] = eig(H);
        lambda = sqrt(diag(Delta));
        X = X*P;
        if (norm(lambda_prec-lambda)/norm(lambda) < tol)
            break;
        end
    end
    [~, loc] = max (abs(X));
    linear_index = sub2ind(size(X),loc, 1:p);
    P = A*X;
    lambda = (P(linear_index)./X(linear_index))';
    [~,index] = sort(abs(lambda),'descend');
    lambda = lambda(index);
    X = X(:,index);
end
```

3.2 Metodo e risultati del confronto sperimentale tra gli Algoritmi

Abbiamo testato i quattro algoritmi su `N` problemi di taglia n, andando a calcolare le p autocoppie dominanti di $A \in \mathbb{R}^{n\times n}$ matrice di taglia n simmetrica sparsa (circa 10 entrate non nulle per ogni riga) generata casualmente. Ogni algoritmo arresta le proprie iterazioni quando si verifica la condizione $\|\lambda^{(k)} - \lambda^{(k-1)}\|/\|\lambda^{(k)}\| < tol$ oppure viene superato il numero massimo di iterazioni `itmax`.

I quattro algoritmi sono stati confrontati, al variare di n e p come riportato nelle seguenti tabelle, in termini di:

- `it`: numero di iterazioni svolte per ciascuna istanza;
- `time`: tempo di risoluzione di un'istanza, in secondi;
- `it_time`: tempo di esecuzione di un'iterazione, in secondi;
- `values_err`: errore relativo massimo commesso sull'approssimazione degli autovettori (assumendo come esatti quelli calcolati da MATLAB)
- `vectors_err`: errore relativo in norma massimo commesso sull'approssimazione degli autovalori (assumendo come esatti quelli calcolati da MATLAB)

Per confrontare gli algoritmi al variare di p abbiamo considerato $N = 20$ problemi di taglia $n = 1000$. I risultati sono riportati nelle seguenti tabelle.

Tabella 1: $n = 1000$, $p = 5$, `itmax` $= 5000$, $tol = 10^{-10}$

Algoritmo	`it`	`time`	`it_time`	`values_err`	`vectors_err`
1	3846	0.996	2.580×10^{-4}	1.857×10^{-6}	0.030
2	3287	1.805	5.482×10^{-4}	5.545×10^{-4}	5.251×10^{-4}
3	2765	0.795	2.853×10^{-4}	8.536×10^{-4}	7.385×10^{-4}
4	2765	0.942	3.402×10^{-4}	8.536×10^{-4}	7.385×10^{-4}

Tabella 2: $n = 1000$, A def. positiva, $p = 5$, `itmax` $= 5000$, $tol = 10^{-10}$

Algoritmo	`it`	`time`	`it_time`	`values_err`	`vectors_err`
1	3468	0.915	2.652×10^{-4}	1.242×10^{-6}	0.030
2	1832	1.031	5.653×10^{-4}	1.393×10^{-7}	3.363×10^{-4}
3	1832	0.524	2.872×10^{-4}	1.391×10^{-7}	3.362×10^{-4}
4	1832	0.623	3.437×10^{-4}	1.391×10^{-7}	3.362×10^{-4}

Tabella 3: $n = 1000$, $p = 10$, `itmax` $= 7500$, $tol = 10^{-10}$

Algoritmo	`it`	`time`	`it_time`	`values_err`	`vectors_err`
1	4647	2.363	5.095×10^{-4}	1.124×10^{-5}	0.073
2	3273	3.136	9.595×10^{-4}	1.552×10^{-4}	8.051×10^{-4}
3	2737	1.319	4.824×10^{-4}	2.248×10^{-4}	8.769×10^{-4}
4	2737	1.697	6.199×10^{-4}	2.249×10^{-4}	8.769×10^{-4}

Tabella 4: $n = 1000$, A def. positiva, $p = 10$, `itmax` $= 7500$, $tol = 10^{-10}$

Algoritmo	`it`	`time`	`it_time`	`values_err`	`vectors_err`
1	4613	2.347	5.087×10^{-4}	3.610×10^{-5}	0.186
2	2377	2.325	9.755×10^{-4}	4.866×10^{-8}	1.833×10^{-4}
3	2376	2.161	4.887×10^{-4}	4.874×10^{-8}	1.837×10^{-4}
4	2376	1.501	6.314×10^{-4}	4.873×10^{-8}	1.847×10^{-4}

Tabella 5: $n = 1000$, $p = 15$, `itmax` $= 7500$, $tol = 10^{-10}$

Algoritmo	`it`	`time`	`it_time`	`values_err`	`vectors_err`
1	7087	5.881	8.294×10^{-4}	1.076×10^{-5}	0.146
2	4571	7.071	1.532×10^{-3}	1.554×10^{-4}	1.994×10^{-4}
3	3687	2.712	7.328×10^{-4}	1.412×10^{-4}	3.282×10^{-4}
4	3687	3.747	1.047×10^{-3}	1.411×10^{-4}	3.282×10^{-4}

Tabella 6: $n = 1000$, A def. positiva, $p = 15$, `itmax` $= 7500$, $tol = 10^{-10}$

Algoritmo	`it`	`time`	`it_time`	`values_err`	`vectors_err`
1	6970	5.891	8.465×10^{-4}	0.019	0.173
2	4420	6.961	1.644×10^{-3}	4.328×10^{-7}	0.006
3	4399	2.302	7.537×10^{-4}	4.405×10^{-7}	0.006
4	4385	4.494	1.079×10^{-3}	4.411×10^{-7}	0.006

Tabella 7: $n = 1000$, $p = 20$, `itmax` $= 7500$, $tol = 10^{-10}$

Algoritmo	`it`	`time`	`it_time`	`values_err`	`vectors_err`
1	7293	8.377	1.198×10^{-3}	8.529×10^{-6}	0.104
2	3575	7.547	2.137×10^{-3}	4.003×10^{-6}	3.123×10^{-5}
3	2735	2.409	8.809×10^{-4}	8.510×10^{-5}	1.473×10^{-4}
4	2735	3.8673	1.447×10^{-3}	8.514×10^{-5}	1.473×10^{-4}

Tabella 8: $n = 1000$, A def. positiva, $p = 20$, `itmax` $= 7500$, $tol = 10^{-10}$

Algoritmo	`it`	`time`	`it_time`	`values_err`	`vectors_err`
1	7492	8.793	1.267×10^{-3}	0.019	0.163
2	4857	10.534	2.299×10^{-3}	9.473×10^{-7}	0.009
3	4827	4.391	9.062×10^{-4}	9.536×10^{-7}	0.009
4	4835	6.951	1.435×10^{-3}	9.525×10^{-7}	0.009

Osserviamo come i risultati sperimentali siano coerenti con l'analisi teorica del costo computazionale: il tempo di esecuzione di un'iterazione dell'Algoritmo 2 è quasi doppio rispetto agli Algoritmi [1,3,4], che infatti eseguono la metà delle chiamate di OP.

Coerentemente con l'analisi teorica della velocità di convergenza inoltre gli Algoritmi [2,3,4] eseguono un numero di iterazioni notevolmente minore rispetto all'Algoritmo 1, raggiungendo inoltre una precisione sostanzialmente maggiore sia sugli autovalori che sugli autovettori.

Le differenze di cui sopra tra gli algoritmi si fanno ancor più marcate al crescere di p. Si può notare come nel passaggio da $p = 5$ a $p = 10$ il tempo di calcolo per un'istanza sia quasi raddoppiato e in alcuni casi triplicato. Questo è dovuto a un leggero aumento del numero di iterazioni svolte, ma sopratutto all'innalzarsi del costo computazionale in tempo di un'iterazione (raddoppia il numero di chiamate di OP). Il numero di iterazioni svolte aumenta indistintamente per tutti gli Algoritmi anche nel passaggio da $p = 10$ a $p = 15$, con la differenza che è invece meno marcata tra $p = 15$ e $p = 20$ poiché è minore l'incremento relativo del valore di p. L'incremento del tempo di esecuzione di un'iterazione segue invece un andamento più regolare.

In aggiunta è interessante notare un comportamento diverso degli algoritmi nel caso di matrici simmetriche qualsiasi o simmetriche definite positive. L'Algoritmo 1, non presenta sostanziali differenze di prestazione tra i due casi, né in termini di tempo né in termini di precisione. Per quanto riguarda invece gli altri tre algoritmi, pensati per essere applicati a matrici definite positive, si possono notare delle differenze notevoli. In primo luogo, mentre nel caso di matrici simmetriche qualsiasi l'Algoritmo 2 esegue un numero maggiore di iterazioni rispetto agli Algoritmi 3 e 4, nel caso di matrici definite positive gli Algoritmi [2,3,4] sono sostanzialmente equivalenti tra loro in termini di numero di iterazioni e precisione raggiunta (sia sugli autovalori che sugli autovettori). Questo mostra come in realtà gli Algoritmi [3,4] siano soltanto una "riscrittura" dell'Algoritmo 2 volta a svolgere il maniera implicita il Quoziente di Rayleigh generalizzato. Inoltre si può notare, a parità di numero di autocoppie calcolato, che la precisione raggiunta sugli autovalori è notevolmente migliore nel caso di matrici definite positive. Per quanto riguarda invece il numero di iterazioni e la precisione raggiunta sugli autovettori, al crescere di p le differenze tra i due casi si assottigliano e addirittura per $p = 15, 20$ sono migliori le prestazioni del caso di matrici simmetriche qualsiasi. Il tempo di esecuzione di un'iterazione è simile per entrambi i casi.

Ci possiamo anche chiedere cosa succede se facciamo variare non p ma n. A tal scopo abbiamo testato i quattro algoritmi su `N = 10` problemi di taglia $n = 5000, 10000$ (oltre al caso $n = 1000$ già esaminato), andando a calcolare le 5 autocoppie dominanti. I risultati sono riportati nelle seguenti tabelle.

All'aumentare di n il numero di iterazioni svolte non aumenta eccessivamente (in alcuni casi addirittura diminuisce), ma aumenta il tempo di esecuzione di un'iterazione e di conseguenza il tempo di risoluzione di un'istanza. L'aumento del tempo di esecuzione di un'iterazione sembra essere proporzionale all'aumento della taglia della matrice. Questo è coerente con l'analisi teorica della complessità computazionale in tempo: se la matrice A è sparsa, come nel caso da noi esaminato sperimentalmente, il costo di una chiamata di OP è un $O(n)$ e dunque il costo di un'iterazione di uno

Tabella 9: $n = 5000, \quad p = 5, \quad \texttt{itmax} = 5000, \quad tol = 10^{-10}$

Algoritmo	it	time	it_time	values_err	vectors_err
1	3336	6.388	1.945×10^{-3}	9.361×10^{-7}	0.033
2	2163	8.214	3.939×10^{-3}	1.266×10^{-4}	0.003
3	1883	3.578	1.971×10^{-3}	1.833×10^{-3}	0.004
4	1883	3.963	2.184×10^{-3}	1.833×10^{-3}	0.004

Tabella 10: $n = 5000, \quad A$ def. positiva, $\quad p = 5, \quad \texttt{itmax} = 5000, \quad tol = 10^{-10}$

Algoritmo	it	time	it_time	values_err	vectors_err
1	2657	4.986	1.947×10^{-3}	3.867×10^{-8}	0.039
2	1729	6.591	3.866×10^{-3}	4.757×10^{-7}	0.003
3	1730	3.248	1.915×10^{-3}	4.735×10^{-7}	0.003
4	1730	3.556	2.101×10^{-3}	4.735×10^{-7}	0.003

Tabella 11: $n = 10000, \quad p = 5, \quad \texttt{itmax} = 5000, \quad tol = 10^{-10}$

Algoritmo	it	time	it_time	values_err	vectors_err
1	4515	10.116	2.229×10^{-3}	1.851×10^{-6}	0.050
2	2167	10.088	4.715×10^{-3}	1.006×10^{-4}	5.528×10^{-4}
3	1527	3.427	2.206×10^{-3}	1.875×10^{-3}	0.003
4	1527	3.925	2.613×10^{-3}	1.859×10^{-3}	0.003

Tabella 12: $n = 10000, \quad A$ def. positiva, $\quad p = 5, \quad \texttt{itmax} = 5000, \quad tol = 10^{-10}$

Algoritmo	it	time	it_time	values_err	vectors_err
1	4417	9.870	2.235×10^{-3}	6.695×10^{-5}	0.374
2	2896	13.422	4.607×10^{-3}	3.575×10^{-8}	0.020
3	2897	6.402	2.221×10^{-3}	3.574×10^{-8}	0.020
4	2897	7.168	2.549×10^{-3}	3.574×10^{-8}	0.020

qualsiasi degli algoritmi è $O(n)$ (cambia però la costante moltiplicativa a seconda dell'algoritmo selezionato).

3.3 Confronto tra la velocità di convergenza teorica e quella effettiva degli algoritmi

Oltre a confrontare tra loro gli algoritmi risulta piuttosto naturale chiedersi se la convergenza di autovalori e autovettori sia effettivamente lineare e, in tal caso, se il fattore di convergenza sia quello ricavato teoricamente.

Abbiamo analizzato sotto tale aspetto ciascuno dei quattro algoritmi, calcolando le $p = 6$ autocoppie dominanti di una matrice simmetrica definita positiva di taglia $n = 10000$. I risultati sono riportati nei grafici delle seguenti pagine.

Dai plot in scala semilogaritmica dell'errore si può osservare come la convergenza sia effettivamente lineare sia per gli autovalori che per gli autovettori. Dopo una iniziale fase in cui l'andamento è maggiormente influenzato dalla scelta della matrice di partenza $X^{(0)}$, in cui cioè prevale la costante moltiplicativa, l'errore diminuisce linearmente, salvo poi stabilizzarsi se viene raggiunta la precisione di macchina.

Per quanto riguarda il fattore di convergenza, per tutti gli algoritmi c'è una buona corrispondenza con quanto previsto teoricamente, con le linee tratteggiate della previsione della convergenza che asintoticamente risultano quasi parallele a quelle della convergenza effettiva (fatta eccezione per le prime iterazioni, in cui l'andamento è influenzato dalle condizioni iniziali). La convergenza degli autovalori dell'Algoritmo 1 è quella maggiormente caotica nelle prime iterazioni, ma questo non ostacola un andamento asintotico in linea con quello previsto teoricamente.

In particolare il fattore di convergenza ricavato teoricamente per la j-esima autocoppia è

$$\gamma_j = max \left\{ \left| \frac{\lambda_j}{\lambda_{j-1}} \right|, \left| \frac{\lambda_{j+1}}{\lambda_j} \right| \right\}$$

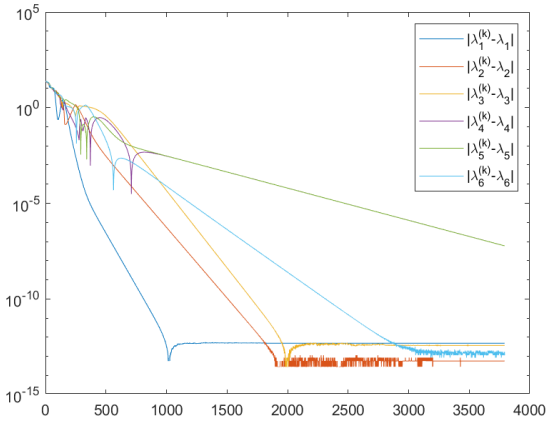
per l'Algoritmo 1 e

$$\gamma_{\lambda_j} = \left(\frac{\lambda_{p+1}}{\lambda_j} \right)^2, \qquad \gamma_{x_j} = \left(\frac{\lambda_{p+1}}{\lambda_j} \right)$$

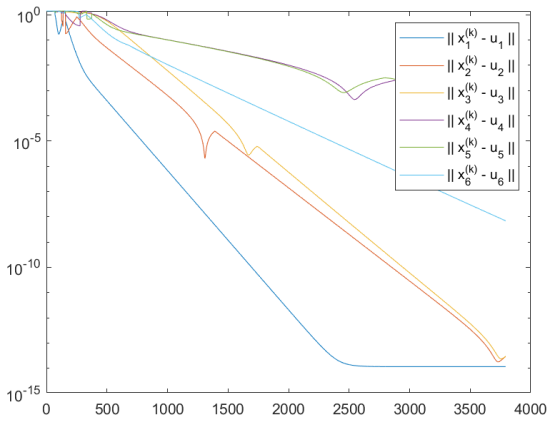
rispettivamente per gli autovalori e per gli autovettori per gli Algoritmi [2,3,4]. Dunque per gli Algoritmi [2,3,4] $\lambda_j^{(k)}$ converge a λ_j più velocemente di quanto faccia $\lambda_{j+1}^{(k)}$ a λ_{j+1}, cosa che si può osservare nelle Figure [3,5,7]. La Figura 1 mostra che ciò non necessariamente accade per l'Algoritmo 1, cosa che ci aspettavamo anche dall'analisi teorica.

3.3.1 Metodo delle Iterazioni ortogonali non accelerato

Figura 1: Algoritmo 1. Plot in scala semilogaritmica dell'errore.

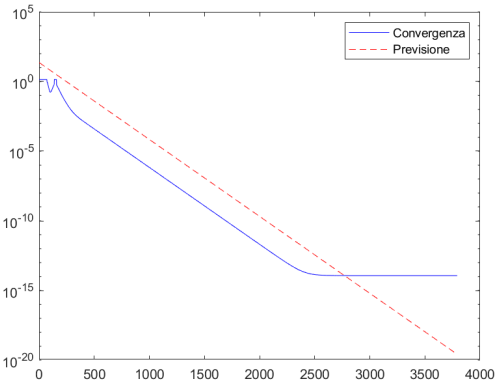


(a) Autovalori

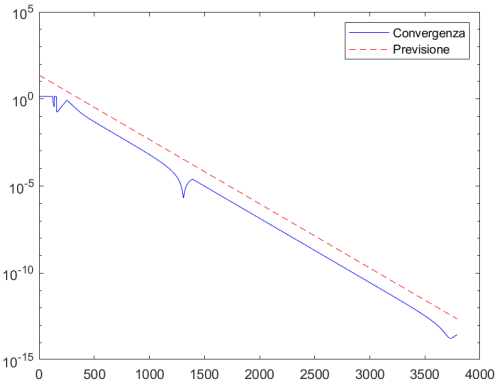


(b) Autovettori

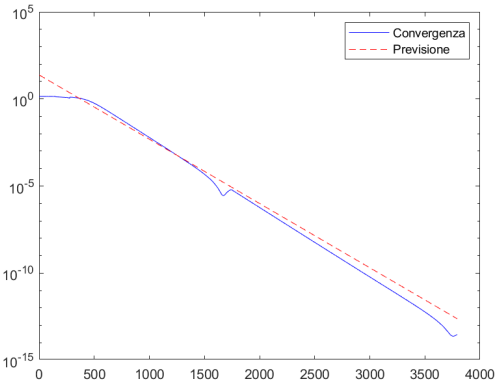
Figura 2: Algoritmo 1. Confronto, in scala semilogaritmica, tra la velocità di convergenza prevista teoricamente e quella sperimentale.



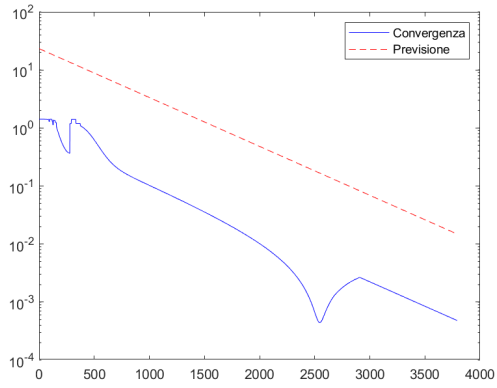
(a) $\|x_1^{(k)} - u_1\|$



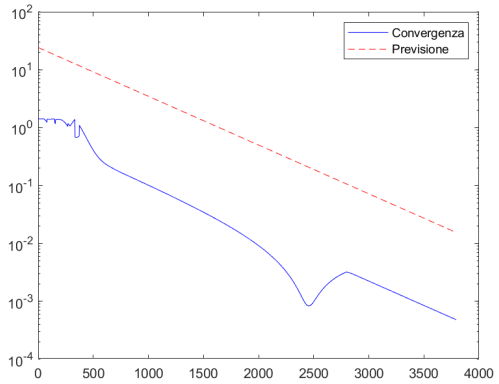
(b) $\|x_2^{(k)} - u_2\|$



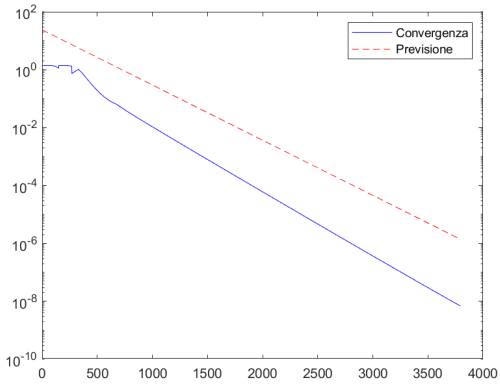
(c) $\|x_3^{(k)} - u_3\|$



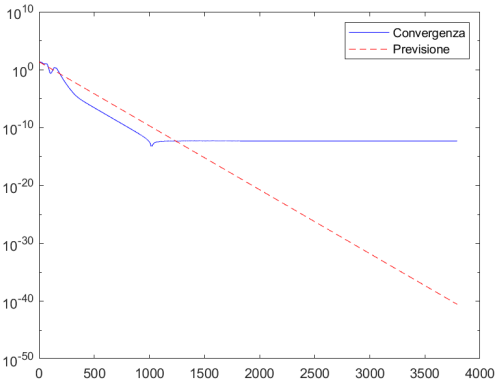
(d) $\|x_4^{(k)} - u_4\|$



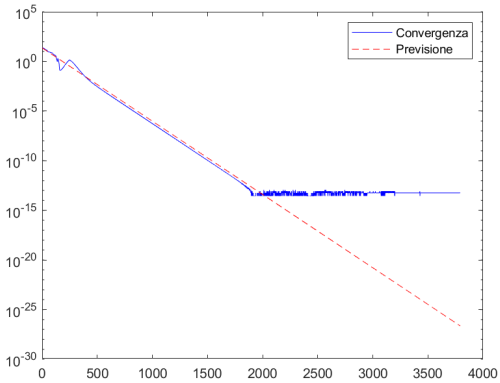
(e) $\|x_5^{(k)} - u_5\|$



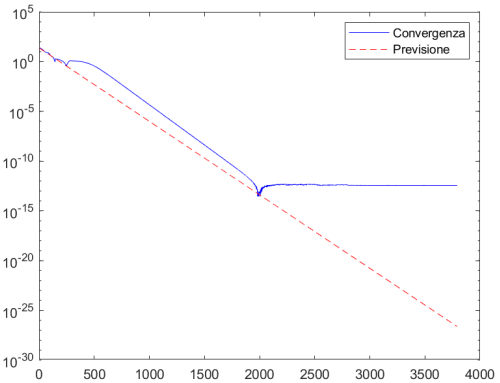
(f) $\|x_6^{(k)} - u_6\|$



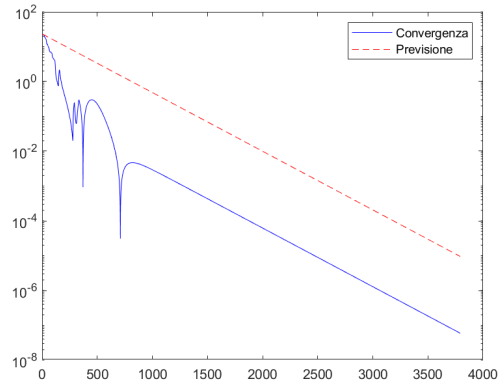
(g) $|\lambda_1^{(k)} - \lambda_1|$



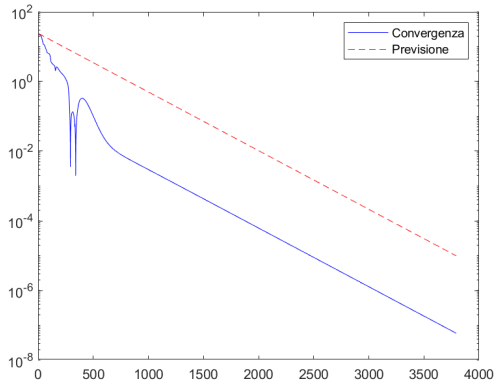
(h) $|\lambda_2^{(k)} - \lambda_2|$



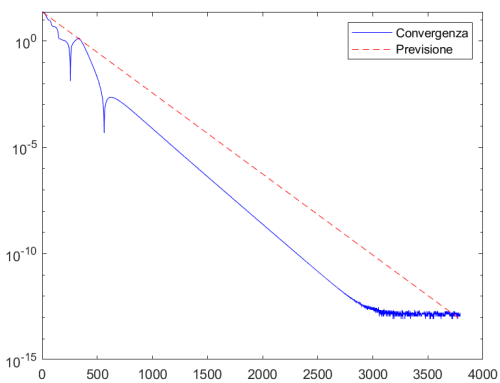
(i) $|\lambda_3^{(k)} - \lambda_3|$



(j) $|\lambda_4^{(k)} - \lambda_4|$



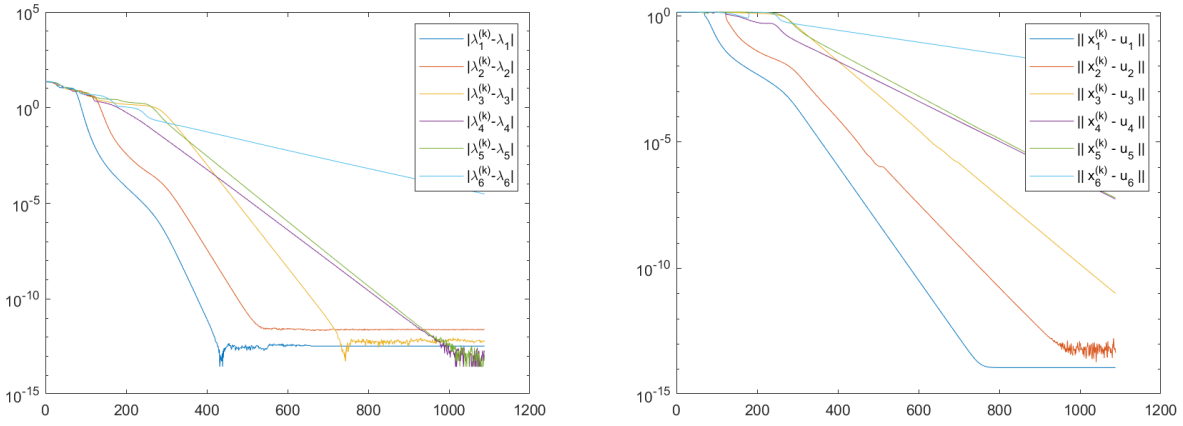
(k) $|\lambda_5^{(k)} - \lambda_5|$



(l) $|\lambda_6^{(k)} - \lambda_6|$

3.3.2 Metodo con accelerazione, prima versione

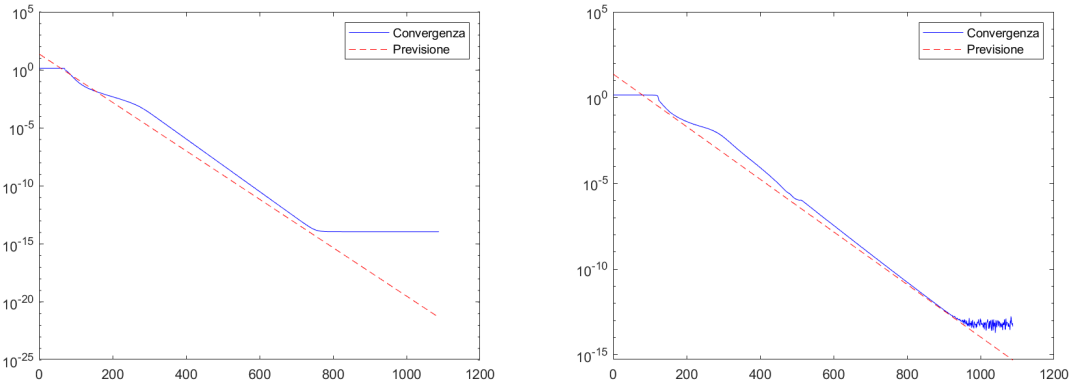
Figura 3: Algoritmo 2. Plot in scala semilogaritmica dell'errore.



(a) Autovalori

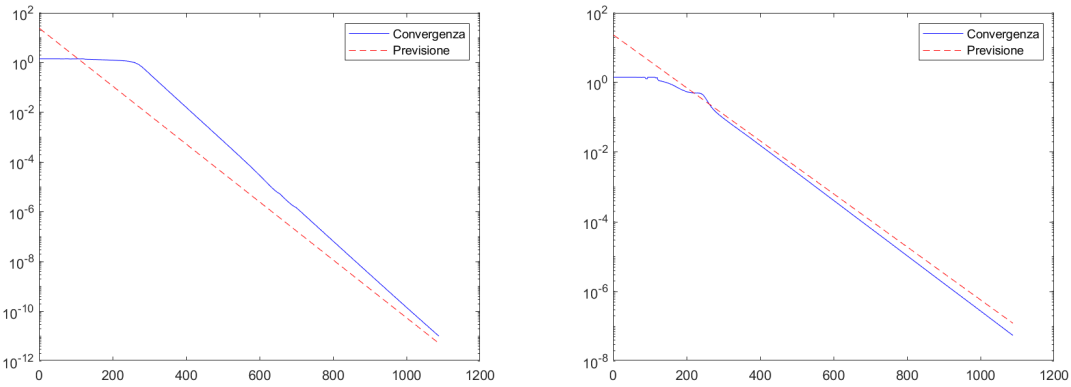
(b) Autovettori

Figura 4: Algoritmo 2. Confronto, in scala semilogaritmica, tra la velocità di convergenza prevista teoricamente e quella sperimentale.



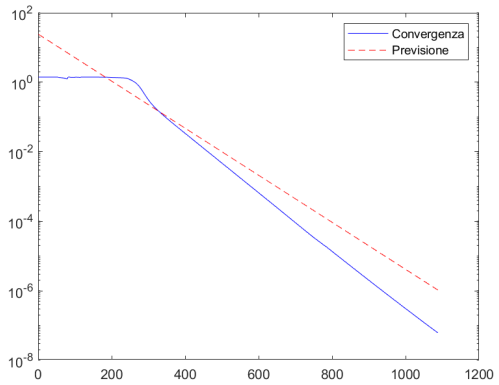
(a) $\|x_1^{(k)} - u_1\|$

(b) $\|x_2^{(k)} - u_2\|$

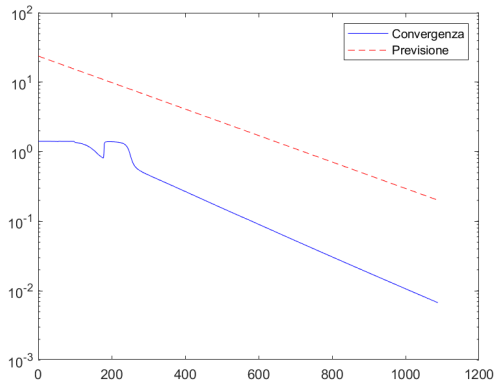


(c) $\|x_3^{(k)} - u_3\|$

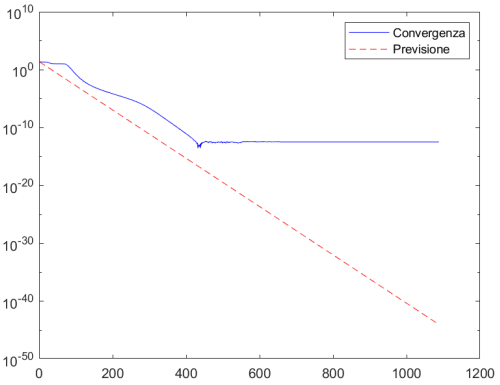
(d) $\|x_4^{(k)} - u_4\|$



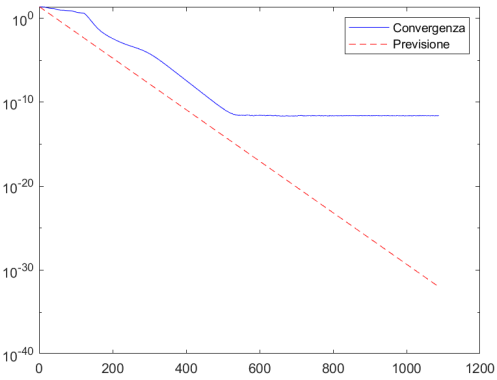
(e) $\|x_5^{(k)} - u_5\|$



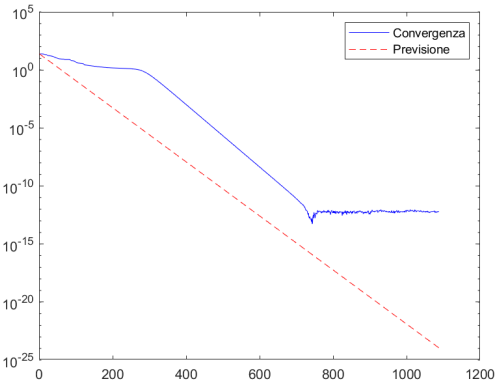
(f) $\|x_6^{(k)} - u_6\|$



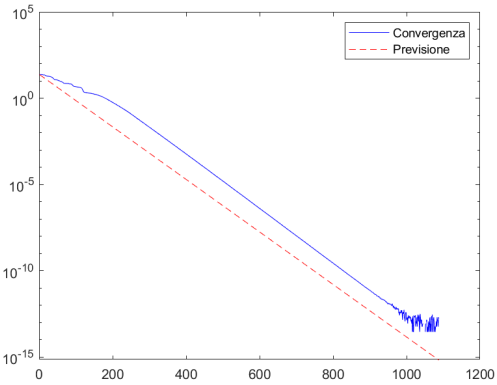
(g) $|\lambda_1^{(k)} - \lambda_1|$



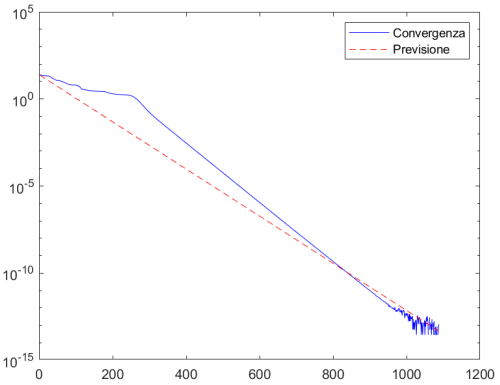
(h) $|\lambda_2^{(k)} - \lambda_2|$



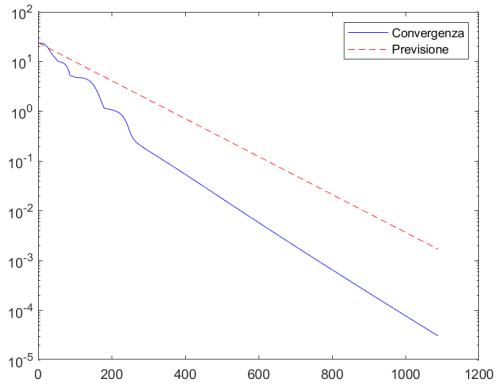
(i) $|\lambda_3^{(k)} - \lambda_3|$



(j) $|\lambda_4^{(k)} - \lambda_4|$



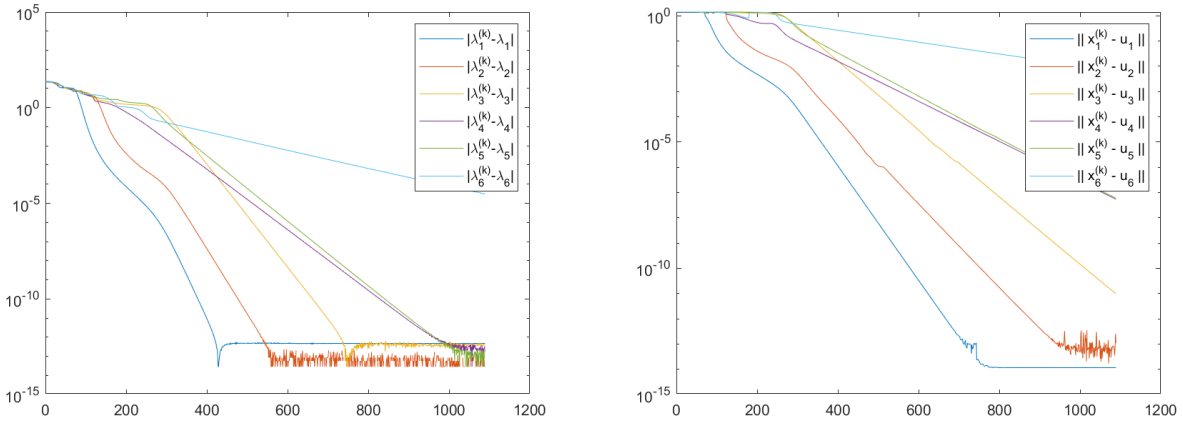
(k) $|\lambda_5^{(k)} - \lambda_5|$



(l) $|\lambda_6^{(k)} - \lambda_6|$

3.3.3 Metodo con accelerazione, seconda versione

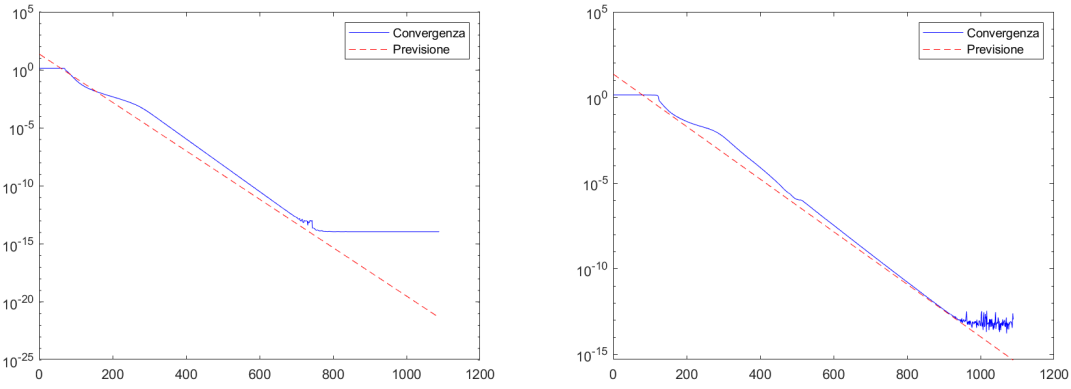
Figura 5: Algoritmo 3. Plot in scala semilogaritmica dell'errore.



(a) Autovalori

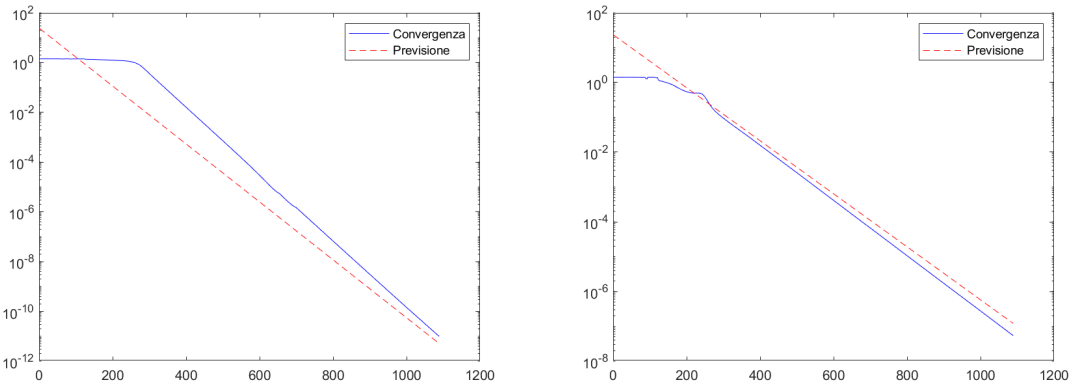
(b) Autovettori

Figura 6: Algoritmo 3. Confronto, in scala semilogaritmica, tra la velocità di convergenza prevista teoricamente e quella sperimentale.



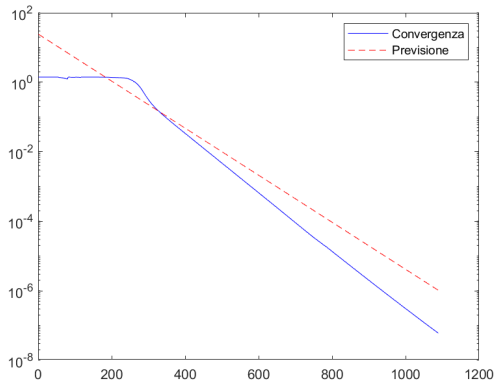
(a) $\|x_1^{(k)} - u_1\|$

(b) $\|x_2^{(k)} - u_2\|$

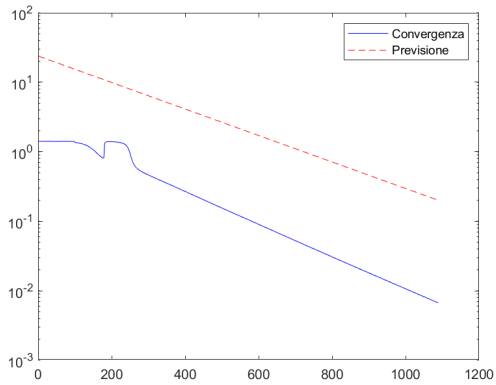


(c) $\|x_3^{(k)} - u_3\|$

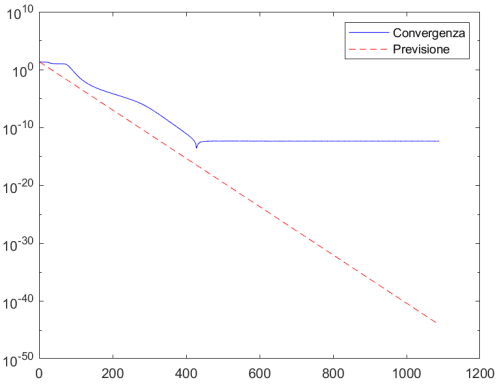
(d) $\|x_4^{(k)} - u_4\|$



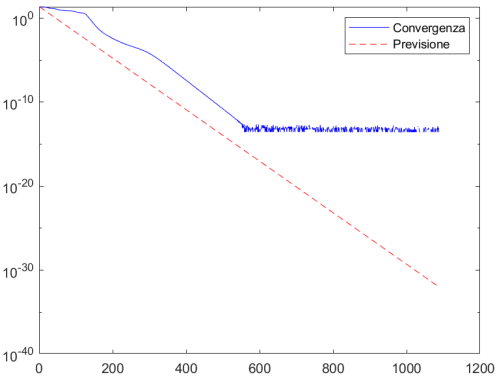
(e) $\|x_5^{(k)} - u_5\|$



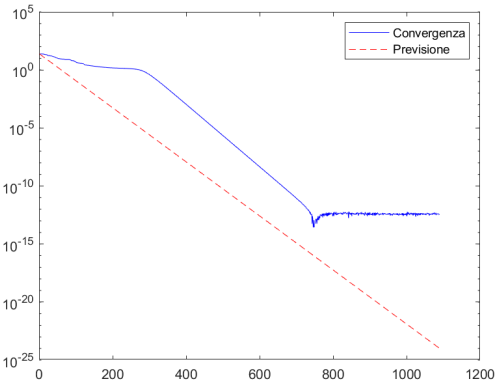
(f) $\|x_6^{(k)} - u_6\|$



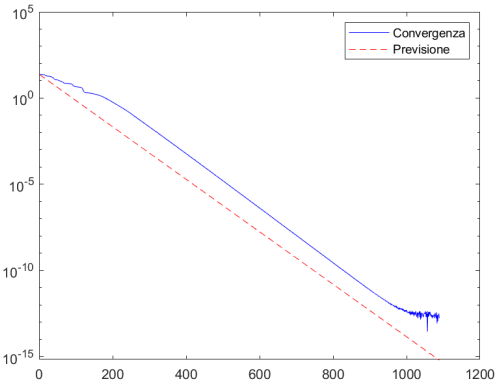
(g) $|\lambda_1^{(k)} - \lambda_1|$



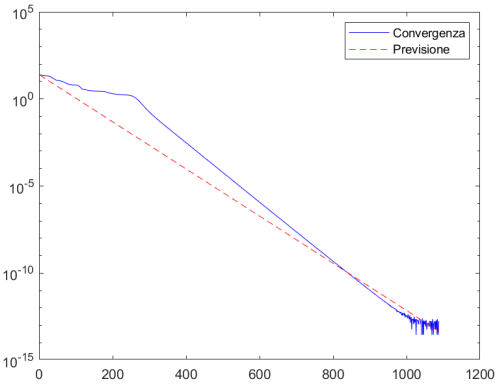
(h) $|\lambda_2^{(k)} - \lambda_2|$



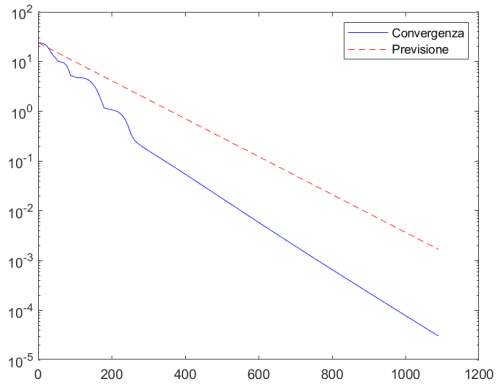
(i) $|\lambda_3^{(k)} - \lambda_3|$



(j) $|\lambda_4^{(k)} - \lambda_4|$



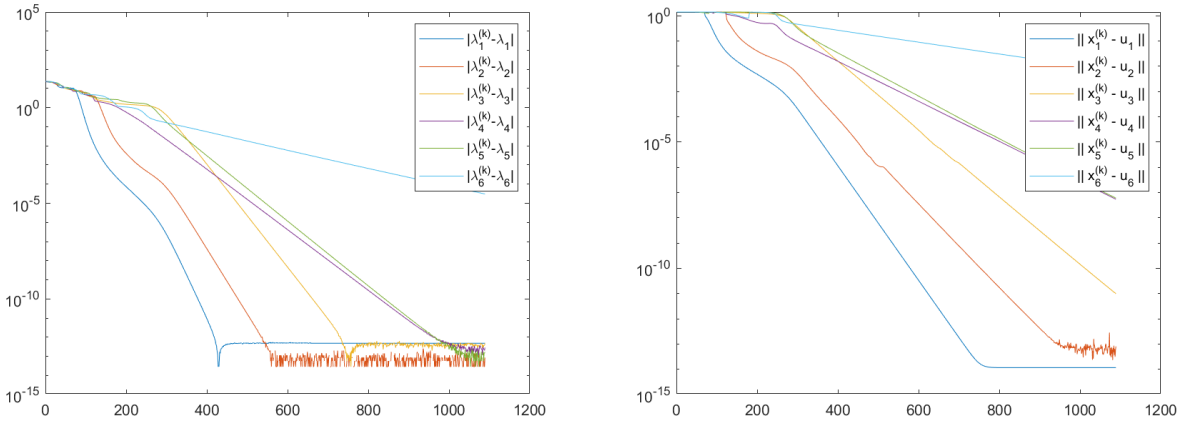
(k) $|\lambda_5^{(k)} - \lambda_5|$



(l) $|\lambda_6^{(k)} - \lambda_6|$

3.3.4 Metodo con accelerazione, terza versione (ritzritz)

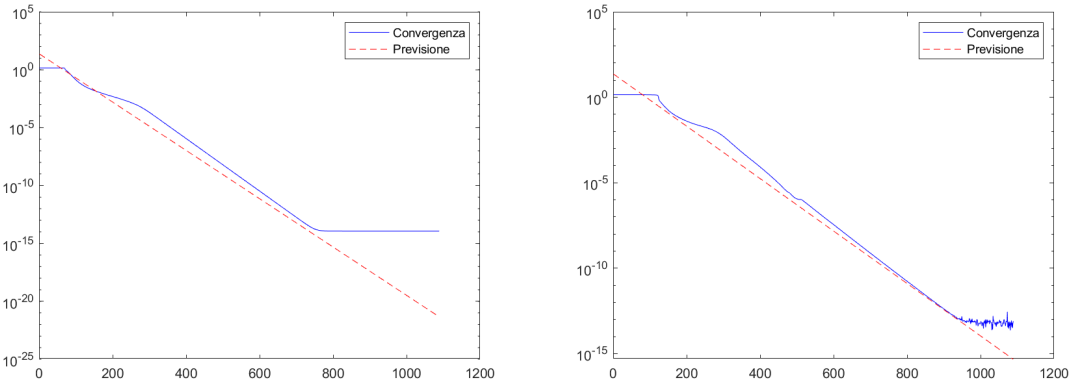
Figura 7: Algoritmo 4. Plot in scala semilogaritmica dell'errore.



(a) Autovalori

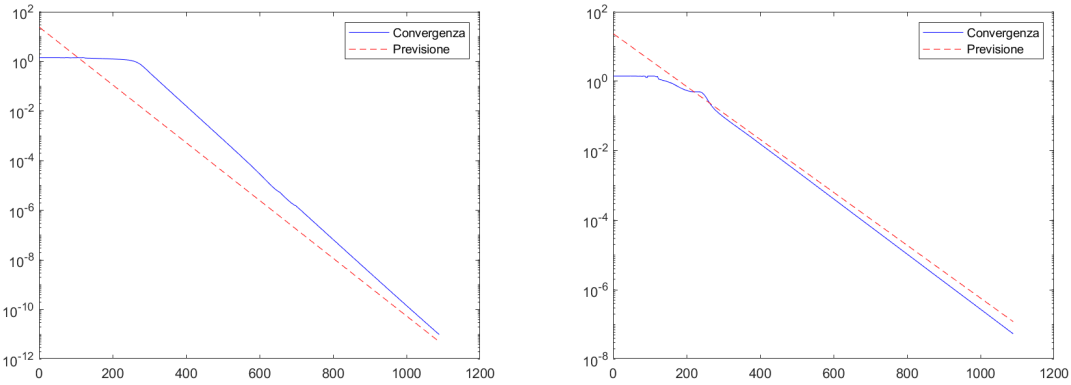
(b) Autovettori

Figura 8: Algoritmo 4. Confronto, in scala semilogaritmica, tra la velocità di convergenza prevista teoricamente e quella sperimentale.



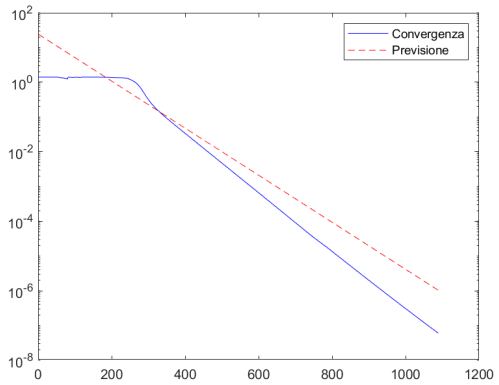
(a) $\|x_1^{(k)} - u_1\|$

(b) $\|x_2^{(k)} - u_2\|$

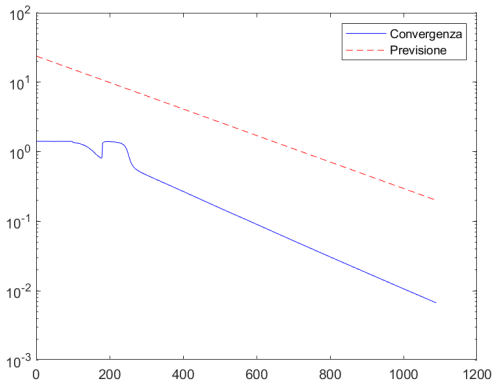


(c) $\|x_3^{(k)} - u_3\|$

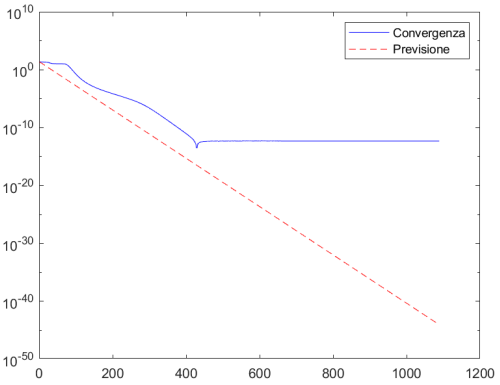
(d) $\|x_4^{(k)} - u_4\|$



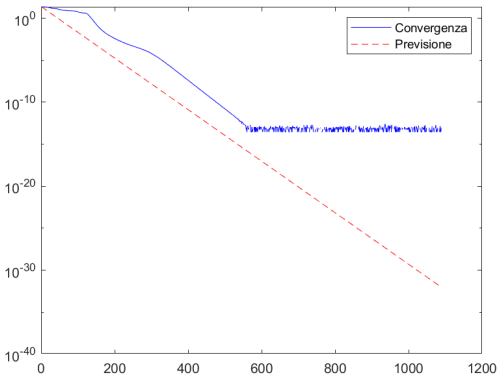
(e) $\|x_5^{(k)} - u_5\|$



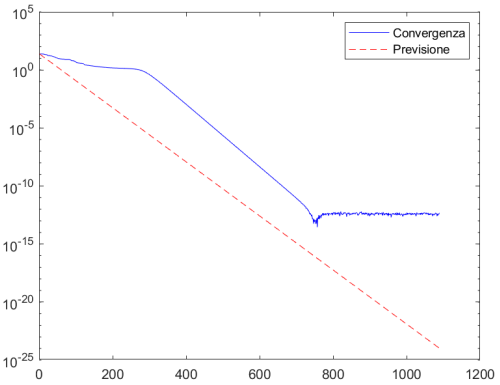
(f) $\|x_6^{(k)} - u_6\|$



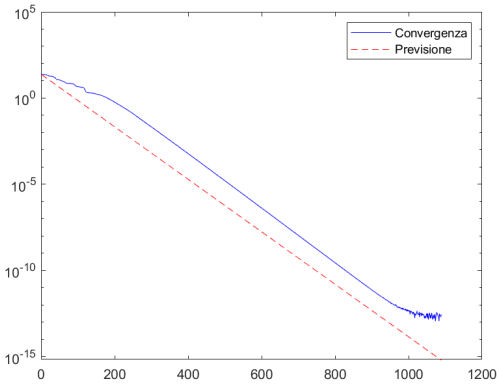
(g) $|\lambda_1^{(k)} - \lambda_1|$



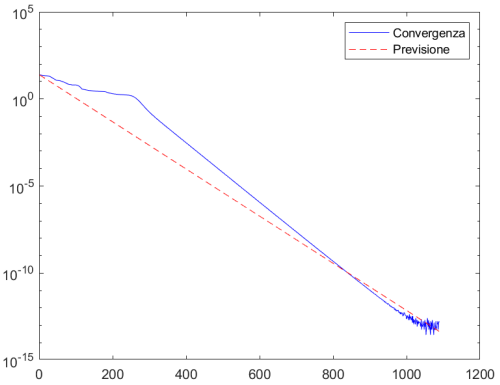
(h) $|\lambda_2^{(k)} - \lambda_2|$



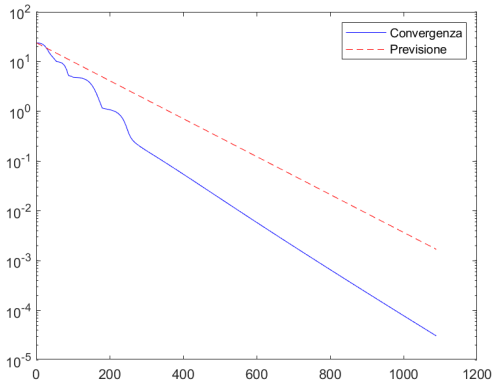
(i) $|\lambda_3^{(k)} - \lambda_3|$



(j) $|\lambda_4^{(k)} - \lambda_4|$



(k) $|\lambda_5^{(k)} - \lambda_5|$



(l) $|\lambda_6^{(k)} - \lambda_6|$

Riferimenti bibliografici

[1] Peter Arbenz. *Numerical Methods for Solving Large Scale Eigenvalue Problems*. 2016.

[2] Dario Bini. *Problemi di vibrazioni*. 2020.

[3] Dario Bini, Milvio Capovani e Ornella Menchi. *Metodi numerici per l'algebra lineare*. Zanichelli, 1988.

[4] Beresford N. Parlett. «14. Subspace Iteration». In: *The Symmetric Eigenvalue Problem*. Society for Industrial e Applied Mathematics, pp. 323–337.

[5] H. Rutishauser. «Simultaneous Iteration Method for Symmetric Matrices». In: *Handbook for Automatic Computation: Volume II: Linear Algebra*. Springer Berlin Heidelberg, 1971, pp. 284–302.