Ricerca del massimo autovalore

Aurelio Roccasanta

01 Agosto 2025

1 Introduzione

La diagonalizzazione di una matrice, e quindi la determinazione dei suoi autovalori, è di fondamentale importanza in numerosi ambiti applicativi, noti come problemi agli autovalori. Tuttavia, la diagonalizzazione di matrici di grandi dimensioni non è un'operazione immediata, nemmeno dal punto di vista computazionale. La difficoltà principale deriva dall'assenza di una soluzione analitica generale per l'equazione del polinomio caratteristico.
In questi casi, diventa utile ricorrere a metodi alternativi che permettano di stimare gli autovalori in modo più o meno preciso, riducendo tempi di calcolo e complessità computazionale. In particolare, il calcolo dell'autovalore massimo può essere affrontato in modo relativamente semplice attraverso i cosiddetti metodi delle potenze, di cui parlerò più nel dettaglio nella sezione successiva.

2 Cenni teorici

Due formule per il calcolo del massimo autovalore della matrice A reale simmetrica (e quindi quadrata) di dimensione k sono:

$$|\lambda_1| = \lim_{n\to\infty} \sqrt[n]{\mathrm{Tr}(A^n)} \tag{1}$$

che chiamerò *metodo della radice*, e:

$$\lambda_1 = \lim_{n\to\infty} \frac{\mathrm{Tr}(A^{n+1})}{\mathrm{Tr}(A^n)} \tag{2}$$

che invece chiamerò *metodo del rapporto*.

Innanzitutto si può osservare (come conseguenza del *teorema spettrale*) che, poiché la matrice A è reale e simmetrica, essa può essere diagonalizzata mediante una matrice ortogonale di autovettori. Ciò garantisce l'esistenza degli autovalori (non necessariamente distinti) e, di conseguenza, possiamo anche affermare l'esistenza di una matrice ortogonale P tale che:

$$A = PDP^{-1} \tag{3}$$

dove D è una matrice diagonale, con tutte le voci nulle eccetto, al più, quelle sulla diagonale principale, che corrispondono agli autovalori.

Dalla (3) si deduce che A e D sono matrici simili (l'esistenza di P che risolve l'equazione ce lo garantisce), per cui, per la proprietà di invarianza della traccia per matrici simili (dovuta alla ciclicità della stessa), deve valere:

$$\mathrm{Tr}(A) = \mathrm{Tr}(D) \tag{4}$$

2.1 Metodo della Radice

Sostituendo la (4) in (1) per il metodo della radice otteniamo:

$$|\lambda_1| = \lim_{n\to\infty} \sqrt[n]{\mathrm{Tr}(D^n)} \tag{5}$$

A questo punto peró risulta immediato calcolare la potenza della matrice (di dimensione k anch'essa essendo simile ad A) che equivale semplicemente alle potenze degli autovalori sulla diagonale, e la traccia alla loro somma, quindi otteniamo:

$$|\lambda_1| = \lim_{n\to\infty} \sqrt[n]{\sum_{i=1}^{k} |\lambda_i|^n} \tag{6}$$

Ora mi trovo finalmente nelle condizioni adatte a risolvere il limite, tenendo a mente che per ipotesi di λ_1 massimo vale $|\lambda_1| \geq |\lambda_i| \quad \forall i \quad \text{t.c.} \quad 1 \leq i \leq k$ e ispirandosi ai limiti dei polinomi, moltiplico il radicando per λ_1^n/λ_1^n:

$$|\lambda_1| = \lim_{n\to\infty} \sqrt[n]{\left(\frac{\lambda_1}{\lambda_1}\right)^n \sum_{i=1}^{k} |\lambda_i|^n} = \lim_{n\to\infty} \sqrt[n]{|\lambda_1|^n \sum_{i=1}^{k} \left(\frac{|\lambda_i|}{|\lambda_1|}\right)^n} \tag{7}$$

Ora per semplicitá suppongo che l'autovalore massimo sia unico e tratteró gli altri casi nell'analisi sulla precisione del metodo, quindi:

$$|\lambda_1| = \lim_{n\to\infty} |\lambda_1| \sqrt[n]{1 + \sum_{i=2}^{k} \left(\frac{|\lambda_i|}{|\lambda_1|}\right)^n} \tag{8}$$

dove in particolare la parentesi é < 1 per ogni autovalore e quindi per n che tende ad infinito la potenza di un numero positivo minore di 1 tenderá a 0 ovvero:

$$|\lambda_1| = \lim_{n\to\infty} \lambda_1 \sqrt[n]{1 + \sum_{i=2}^{k} \left(\frac{|\lambda_i|}{|\lambda_1|}\right)^n} = |\lambda_1| \sqrt[n]{1+0+0+\cdots+0} = |\lambda_1| \tag{9}$$

Ovviamente, nel calcolo del limite, n è stato considerato tendente all'infinito, cosa che naturalmente "non è fattibile" nell'applicazione del metodo tramite un programma. Il valore ottenuto porterà con sé un certo errore, che potremo

facilmente contenere entro un certo intervallo. Infatti, è sufficiente osservare che la convergenza a λ_1 è esponenzialmente più rapida se gli altri autovalori sono "più distanti da quello massimo" (fermo restando che siano minori in modulo). L'errore massimo sarà quindi necessariamente presente quando gli autovalori sono "più vicini possibile" a quello massimo, ovvero nei casi in cui $\lambda_1 = \lambda_2 = \cdots = \lambda_k$.

Andiamo quindi a studiare il comportamento del limite in (9) utilizzando quest'ultima considerazione, e assumendo ora che n sia un numero finito:

$$|\lambda_1(n)| = \lambda_1 \sqrt[n]{1 + \sum_{i=2}^{k} \left(\frac{|\lambda_i|}{|\lambda_1|}\right)^n} = \lambda_1 \sqrt[n]{\sum_{i=1}^{k} \left(\frac{|\lambda_1|}{|\lambda_1|}\right)^n} = \lambda_1 \sqrt[n]{\sum_{i=1}^{k} 1^n} = |\lambda_1| \sqrt[n]{k} \tag{10}$$

Da cui posso definire come $\epsilon_n = \sqrt[n]{k} - 1$ il fattore che introduce l'errore, ottenendo così:

$$|\lambda_1(n)| = |\lambda_1| (1 + \epsilon_n) \tag{11}$$

Osservando questo errore, notiamo una sua importante proprietà: sarà infatti evidente nell'esecuzione del programma che il valore reale viene approssimato "dall'alto". Vedremo quindi valori inizialmente più grandi, che tenderanno gradualmente ad abbassarsi fino ad arrivare alla miglior stima del valore reale.

2.2 Metodo del rapporto

Procedendo in modo analogo e sostituendo la (4) nella (2) otteniamo:

$$\lambda_1 = \lim_{n\to\infty} \frac{\mathrm{Tr}(D^{n+1})}{\mathrm{Tr}(D^n)} = \lim_{n\to\infty} \frac{\sum_{i=1}^{k} \lambda_i^{n+1}}{\sum_{i=1}^{k} \lambda_i^n} \tag{12}$$

A questo punto, ispirandoci ancora una volta ai limiti delle funzioni polinomiali e considerando λ_1 come massimo, dividiamo numeratore e denominatore per λ_1^{n+1}:

$$\lambda_1 = \lim_{n\to\infty} \frac{\sum_{i=1}^{k} \lambda_i^{n+1}}{\sum_{i=1}^{k} \lambda_i^n} = \lim_{n\to\infty} \lambda_1 \frac{\sum_{i=1}^{k} \left(\frac{\lambda_i}{\lambda_1}\right)^{n+1}}{\sum_{i=1}^{k} \left(\frac{\lambda_i}{\lambda_1}\right)^n} \tag{13}$$

Nel denominatore abbiamo raccolto $1/\lambda_1$, quindi abbiamo estratto λ_1 come fattore comune davanti all'espressione. Ora è sufficiente operare un cambio di indice sulla sommatoria per calcolare il limite più facilmente:

$$\lambda_1 = \lim_{n\to\infty} \sum_{i=1}^{k} \left(\frac{\lambda_i}{\lambda_1}\right)^{n+1} \Bigg/ \sum_{i=1}^{k} \left(\frac{\lambda_i}{\lambda_1}\right)^n = \lim_{n\to\infty} \lambda_1 \left(\frac{1 + \sum_{i=2}^{k} \left(\frac{\lambda_i}{\lambda_1}\right)^{n+1}}{1 + \sum_{i=2}^{k} \left(\frac{\lambda_i}{\lambda_1}\right)^n} \right) \tag{14}$$

Anche in questo caso, i rapporti tra autovalori nelle parentesi sono sempre minori di uno, e quindi, elevati alla potenza n, tendono a zero per $n \to \infty$, quindi:

$$\lambda_1 = \lim_{n\to\infty} \lambda_1 \left(\frac{1 + \sum_{i=2}^{k} \left(\frac{\lambda_i}{\lambda_1}\right)^{n+1}}{1 + \sum_{i=2}^{k} \left(\frac{\lambda_i}{\lambda_1}\right)^{n}} \right) = \lambda_1 \left(\frac{1+0}{1+0} \right) = \lambda_1 \tag{15}$$

Una considerazione interessante in questo caso è che il metodo del rapporto fornisce anche informazioni sul segno dell'autovalore, proprietà dovuta alla presenza di potenze "spaiate" tra numeratore e denominatore, che quindi mantengono alternativamente il segno. Anche in questo caso però, per quanto grande sia n, si ha a che fare con un valore finito, quindi sarà presente un certo errore. Possiamo definire il "fattore d'errore" ϵ_n come la "differenza relativa" tra il valore stimato $\lambda_1(n)$ e il valore effettivo λ_1:

$$\lambda_1(n) = \lambda_1 \cdot (1 - \epsilon_n) \tag{16}$$

Dove il fattore d'errore ϵ_n è dato da:

$$\epsilon_n = \frac{\sum_{i=2}^{k} \left(\frac{\lambda_i}{\lambda_1}\right)^{n} - \sum_{i=2}^{k} \left(\frac{\lambda_i}{\lambda_1}\right)^{n+1}}{1 + \sum_{i=2}^{k} \left(\frac{\lambda_i}{\lambda_1}\right)^{n}} \tag{17}$$

Il fattore d'errore ϵ_n rappresenta la differenza relativa tra il valore calcolato $\lambda_1(n)$ e il valore reale λ_1. Questo errore diminuisce rapidamente all'aumentare di n, specialmente quando gli autovalori λ_i per $i \geq 2$ sono significativamente più piccoli, in modulo, rispetto a λ_1. Anche osservando la forma dell'errore possiamo trarre informazioni utili: poiché i termini all'interno delle parentesi sono ≤ 1, il secondo addendo al numeratore è sempre minore del primo. Con n pari, tutti i valori delle somme sono positivi, per cui $\epsilon_n > 0$, e come definito prima, si ha $\lambda_1(n) < \lambda_1$, dimostrando quindi che il valore vero viene approssimato "asintoticamente da sotto".

3 Programma risolutivo

Puó essere trovato nel mio github qui

3.1 Main

Il *main* del programma è riportato nella Figura 1, il cui scopo è innanzitutto raccogliere, tramite una console interattiva, i dati di input — ovvero la matrice A, i cenni teorici e la sua dimensione — per poi chiamare le tre funzioni che calcolano il massimo autovalore e infine mostrarne i risultati (la terza esegue il calcolo con il metodo classico, così che i valori, accompagnati dall'errore di entrambi i metodi numerici, possano essere confrontati con quello reale). I valori ottenuti vengono scritti nel file `dati.dat`, che si troverà nella stessa directory in cui è stato avviato il processo.

Possiamo notare nel codice anche un'allocazione dinamica della memoria, in modo tale che la dimensione della matrice sia variabile (e potenzialmente anche il nome del file, ma io l'ho lasciato *hardcoded*). Un punto importante nel programma è la memoria allocata per le variabili intermedie utilizzate durante il calcolo degli autovalori.

Infatti, specialmente nel metodo del rapporto, è molto facile incorrere in un *overflow* di memoria per numeri troppo alti (più gli autovalori sono grandi e più alto è il limite — cioè l'esponente massimo — più sarà problematico per la naturale continuità del programma). Ho quindi optato per un `KIND=8`, che, tradotto, significa 64 bit per la rappresentazione del numero in sistema binario con virgola mobile, e ho poi trattato singolarmente i vari casi in cui viene raggiunto il limite.

```
PROGRAM autovalore

    IMPLICIT NONE

    REAL(KIND=8), ALLOCATABLE :: matrice(:,:)
    REAL(KIND=8), ALLOCATABLE :: valori_sqrt(:), valori_rapporto(:)
    CHARACTER(LEN=:), ALLOCATABLE :: nome_file
    INTEGER :: i, j, n, limite, unita_file

    ALLOCATE(CHARACTER(LEN=20) :: nome_file)
    nome_file = "dati.dat"
    unita_file = 10

    limite = 1000

    PRINT *, "Inserire la dimensione della matrice:"
    READ *, n

    ALLOCATE(matrice(n, n))!allocazione dinamica della memoria
    ALLOCATE(valori_sqrt(limite), valori_rapporto(limite))

    CALL InizializzazioneMatrice(matrice, n)

    PRINT *, "La matrice inserita è:"
    DO i = 1, n
        PRINT *, (matrice(i, j), j = 1, n)
    END DO

    OPEN(unit=unita_file, file=nome_file, status='replace')

    ! Intestazione
    WRITE(unita_file, '(A)') "# iterazione   metodo_sqrt             metodo_rapporto"

    !CALL CalcolaAutovalori(matrice, n)!massimo autovalore con metodo classico
    CALL trovaMaxAutovaloreSQRT(matrice, n, limite, valori_sqrt)
    CALL trovaMaxAutovaloreRapporto(matrice, n, limite, valori_rapporto)

    DO i = 1, limite !scrivo i risultati nel file tutti nello stesso momento
        WRITE(unita_file, '(I5, F25.15, F25.15)') i, valori_sqrt(i), valori_rapporto(i)
    END DO

    PRINT *, "Risultati scritti con successo nel file."

    DEALLOCATE(matrice)!libero la memoria dinamicamente assegnata
    DEALLOCATE(valori_sqrt, valori_rapporto)
    DEALLOCATE(nome_file)
```

Figure 1: main

3.2 Inizializzazione della Matrice

Questa semplice funzione itera su righe e colonne per richiedere, leggere e assegnare le entrate reali della matrice.

```
48  CONTAINS
49
50  SUBROUTINE InizializzazioneMatrice(mat, n)
51      REAL(KIND=8), INTENT(INOUT) :: mat(:,:)
52      INTEGER, INTENT(IN) :: n
53      DOUBLE PRECISION :: entrateMatrice
54      INTEGER :: i, j
55
56      DO i = 1, n
57          DO j = 1, n
58              PRINT *, "Inserisci l'elemento (", i, ",", j, "):"
59              READ *, entrateMatrice
60              mat(i, j) = REAL(entrateMatrice, KIND=8)
61          END DO
62      END DO
63  END SUBROUTINE InizializzazioneMatrice
```

Figure 2: Inizializzazione della matrice

3.3 Applicazione del metodo della radice

Questa funzione é abbastanza intuitiva: il funzionamento equivale al calcolo giá eseguito sopra per n crescenti fino ad arrivare a *limit*. A meno che nei passaggi una delle variabili intermedie non ecceda in memoria. In quel caso ho sfruttato la naturale decrescenza dei valori al salire di n per verificare che nei risultati non venga scritto *infinity* o NaN.

```
65  SUBROUTINE trovaMaxAutovaloreSQRT(mat, n, limit, risultati)
66      REAL(KIND=8), INTENT(IN) :: mat(:,:)
67      INTEGER, INTENT(IN) :: n, limit
68      REAL(KIND=8), INTENT(OUT) :: risultati(limit)
69      REAL(KIND=8), ALLOCATABLE :: Ak(:,:), temp(:,:)
70      INTEGER :: i, k
71      REAL(KIND=8) :: trace, valore_corrente, ultimo_valido
72
73      ALLOCATE(Ak(n,n), temp(n,n))
74      Ak = mat
75
76      ultimo_valido = 0.0_8
77
78      DO k = 1, limit
79
80          trace = 0.0_8
81          DO i = 1, n
82              trace = trace + Ak(i, i)
83          END DO
84
85          valore_corrente = trace ** (1.0_8 / REAL(k, KIND=8))
86
87          IF (.NOT. (valore_corrente == valore_corrente) .OR. ABS(valore_corrente) >= HUGE(valore_corrente)) THEN
88              risultati(k) = ultimo_valido
89          ELSE
90              risultati(k) = valore_corrente
91              ultimo_valido = valore_corrente
92          END IF
93
94          temp = MATMUL(Ak, mat)
95          Ak = temp
96      END DO
97
98      DEALLOCATE(Ak, temp)
99
100     PRINT '(A, F25.15)', "Valore finale (metodo radice):", ultimo_valido
101 END SUBROUTINE trovaMaxAutovaloreSQRT
```

Figure 3: Ricerca del massimo autovalore con il metodo della radice

3.4 Applicazione del metodo del rapporto

Anche in questo caso la funzione rispecchia i calcoli riportati in precedenza. Tuttavia, a differenza del metodo della radice, vanno effettuati più controlli sulle variabili intermedie: ciò che vogliamo evitare sono divisioni per zero e `infinity`.

Come per il metodo della radice, la funzione termina salvando i risultati in un vettore, in modo che possano essere scritti nel file parallelamente agli altri valori (in Fortran la scrittura su file equivale alla creazione di una nuova riga, e non è pertanto possibile scrivere colonne diverse in momenti differenti).

```
65  SUBROUTINE trovaMaxAutovaloreSQRT(mat, n, limit, risultati)
66      REAL(KIND=8), INTENT(IN) :: mat(:,:)
67      INTEGER, INTENT(IN) :: n, limit
68      REAL(KIND=8), INTENT(OUT) :: risultati(limit)
69      REAL(KIND=8), ALLOCATABLE :: Ak(:,:), temp(:,:)
70      INTEGER :: i, k
71      REAL(KIND=8) :: trace, valore_corrente, ultimo_valido
72
73      ALLOCATE(Ak(n,n), temp(n,n))
74      Ak = mat
75
76      ultimo_valido = 0.0_8
77
78      DO k = 1, limit
79
80          trace = 0.0_8
81          DO i = 1, n
82              trace = trace + Ak(i, i)
83          END DO
84
85          valore_corrente = trace ** (1.0_8 / REAL(k, KIND=8))
86
87          IF (.NOT. (valore_corrente == valore_corrente) .OR. ABS(valore_corrente) >= HUGE(valore_corrente)) THEN
88              risultati(k) = ultimo_valido
89          ELSE
90              risultati(k) = valore_corrente
91              ultimo_valido = valore_corrente
92          END IF
93
94          temp = MATMUL(Ak, mat)
95          Ak = temp
96      END DO
97
98      DEALLOCATE(Ak, temp)
99
100     PRINT '(A, F25.15)', "Valore finale (metodo radice):", ultimo_valido
101 END SUBROUTINE trovaMaxAutovaloreSQRT
```

Figure 4: Ricerca del massimo autovalore con il metodo del rapporto

3.5 Applicazione del metodo classico (DSYEV)

L'applicazione di questo metodo viene in realtà eseguita con l'ausilio del modulo `DSYEV`: la funzione prepara e commenta gli input di cui ha bisogno, e ne valuta gli output, tenendo conto del fatto che un'ipotesi chiave nella parte teorica è che la matrice sia simmetrica — condizione che non viene invece verificata nei due metodi numerici.

Per questo motivo il metodo classico viene chiamato per primo: se la matrice non è diagonalizzabile, o se per qualsiasi ragione non è possibile calcolare gli autovalori, il modulo utilizzato restituirà un codice di errore e il flusso del programma verrà interrotto immediatamente.

Il risultato ottenuto viene poi mostrato a schermo. Anche per le istruzioni di `PRINT` è stata mantenuta la stessa formattazione utilizzata per la scrittura nel file, in modo da evitare discrepanze — anche se lievi — dovute alla rappresentazione in virgola mobile.

```
127     ! Funzione per calcolare gli autovalori nel modo classico
128 SUBROUTINE CalcolaAutovalori(mat,n)
129         REAL(KIND=8), INTENT(INOUT) :: mat(:,:)
130         INTEGER, INTENT(IN) :: n
131         REAL(KIND=8), ALLOCATABLE :: autovalori(:)
132         REAL :: maxAutovalore
133         CHARACTER(LEN=1) :: jobz, uplo
134         INTEGER :: info
135 
136         ! Allocazione della variabile per gli autovalori
137         ALLOCATE(autovalori(n))
138 
139         ! Configurazione per LAPACK
140         jobz = 'N'  ! 'N' = calcola solo gli autovalori, 'V' = calcola anche gli autovettori
141         uplo = 'U'  ! 'U' = usa la parte superiore della matrice, 'L' = usa inferiore
142 
143         ! Calcolo degli autovalori con DSYEV
144         CALL DSYEV(jobz, uplo, n, mat, n, autovalori, info)
145 
146         ! Controllo il risultato
147         IF (info /= 0) THEN
148             PRINT *, "Errore nella diagonalizzazione: codice", info
149             STOP "Errore durante il calcolo degli autovalori"
150         ELSE
151             maxAutovalore = MAXVAL(autovalori)
152             PRINT '(A, F25.15)',"il valore dell'autovalore massimo trovato col metodo classico é:",maxAutovalore
153         END IF
154     END SUBROUTINE CalcolaAutovalori
155 
156 END PROGRAM autovalore
```

Figure 5: Ricerca del massimo autovalore con il metodo del rapporto

4 Risultati

In questa sezione sono riportati degli esempi di utilizzo del programma. Consideriamo come primo caso la seguente matrice reale e simmetrica:

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

questi sono i dati scritti nel file (per mostrarli ho utilizzato il programma `GNUPLOT`):

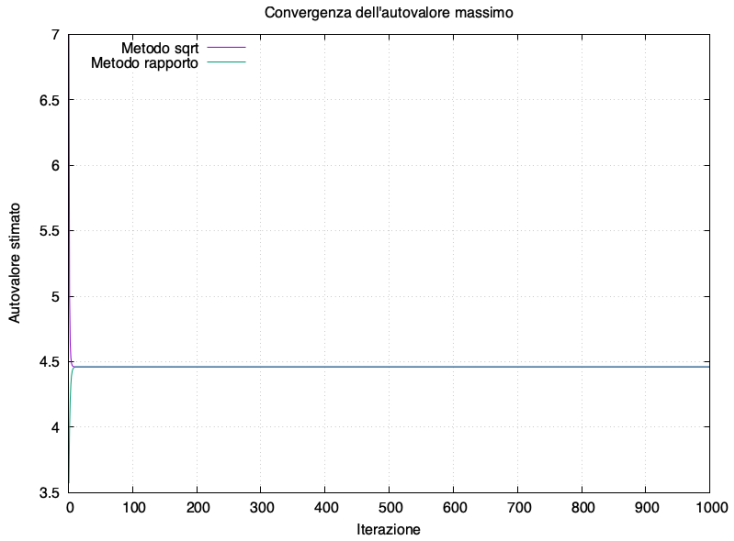

Figure 6: Metodi numerici sulla matrice A ($\lambda_1 = 4.46$)

Ora prendo come esempio la matrice reale e simmetrica B:

$$B = \begin{pmatrix} 67 & 121 & 88 \\ 121 & 93 & 44 \\ 88 & 44 & 76 \end{pmatrix}$$

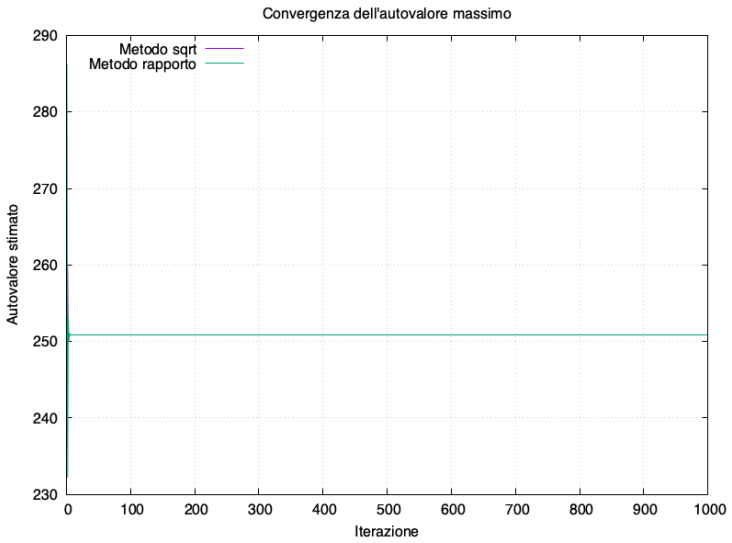

Figure 7: Metodi numerici sulla matrice B ($\lambda_1 = 250$)

5 Considerazioni finali

Guardando i grafici e tenendo a mente la trattazione teorica é evidente che nel caso del metodo della radice il valore reale viene asintoticamente raggiunto dall'alto, mentre per il metodo del rapporto il limite viene approcciato da sotto. Questa differenza suggerisce che una buona approssimazione dell'autovalore potrebbe essere la media tra i due; questa é solo una considerazione generica sui residui. Naturalmente i due metodi sono differenti e un'analisi piú approfondita potrebbe fare chiarezza su quale metodo usare in base alla matrice che si sta studiando. Il metodo classico rimane comunque il piú preciso, ma per matrici sempre piú grandi la complessitá del problema, e quindi il tempo di risoluzione, cresce in modo lineare per i metodi numerici, mentre per il metodo di diagonalizzazione classico cresce esponenzialmente ($O(n^3)$ come minimo e poi a salire). Va infine menzionato anche un importante limite di questi metodi, ovvero la grandezza dei numeri trattati per i quali si arriva facilmente ad un *overflow*, ma non dovrebbe essere difficile modificare il codice per aumentarne la memoria o anche renderlo piú efficiente.