Un algoritmo divide-et-impera per il problema agli autovalori non simmetrico

Francesca Pistolato

9 Maggio 2017

Piano della presentazione

Il problema agli autovalori

Definizione

Data una matrice $H \in \mathbb{R}^{n \times n}$ definiamo problema agli autovalori la risoluzione dell'equazione $det(\lambda I - H) = 0$.
Definiamo eigenpair la coppia (x, λ) tale che x è un vettore non nullo e $Hx = \lambda x$.

Più precisamente descriveremo un metodo per risolvere il problema agli autovalori algebrico, cioè determineremo anche un sistema di autovettori relativi.

Divide et impera

Risolvere tale equazione è equivalente a calcolare uno zero della funzione

$$F_s(x, \lambda) = \begin{pmatrix} Hx - \lambda x \\ e_s^T x - 1 \end{pmatrix},$$

dove $L(x) = e_s^T x - 1$ rappresenta una condizione di normalizzazione sul vettore x.

Sfrutteremo la velocità di convergenza del metodo di Newton nel caso di zeri semplici applicato a valori iniziali ottenuti dagli eigenpair di sotto-matrici di H.

Setting

Assumiamo che la matrice $H = (h_{i,j}) \in \mathbb{R}^{n \times n}$ sia in forma di Hessenberg superiore con elementi sotto-diagonali α_i non nulli, cioè della forma

$$H = \begin{pmatrix} d_1 & & & * \\ \alpha_1 & d_2 & & \\ & \ddots & \ddots & \\ 0 & & \alpha_{n-1} & d_n \end{pmatrix}.$$

Osservazione

- Le ipotesi fatte su H non sono restrittive
- Gli autovalori di H hanno molteplicità geometrica al più 1
- Qualsiasi sotto-matrice di testa o di coda di H ha la stessa struttura

Se $\alpha = h_{k+1,k}$ e $E = -\alpha e_{k+1} e_k^T$, allora

$$H = \left(\begin{array}{c|c} H_1 & H_{12} \\ \hline 0 \ \alpha & H_2 \end{array}\right) = \left(\begin{array}{c|c} H_1 & H_{12} \\ \hline 0 & H_2 \end{array}\right) - E := H_0 - E.$$

Proposizione

Se $H + E = H_0$ e (x, λ) è un eigenpair di H, allora in un intorno di 0 abbastanza piccolo, $(x(\varepsilon), \lambda(\varepsilon))$ è una funziona analitica di ε e un eigenpair di $H + \varepsilon E$. In particolare

$$|\lambda'(0)| = \frac{|\alpha| \cdot |y_{k+1}| \cdot |x_k|}{|y^H x|}.$$

Soluzione

Determinare l'elemento $\alpha = h_{k+1,k}$ di modulo minimo fra quelli in una sezione della sotto-diagonale.

Dato che H_0 è triangolare a blocchi, vale $Sp(H_0) = Sp(H_1) \cup Sp(H_2)$.
Se (x_1, λ) è eigenpair di H_1 e (x_2, ξ) lo è di H_2 i vettori

$$\tilde{x_1} = \begin{pmatrix} x_1 \\ 0 \end{pmatrix} \text{ e } \tilde{x_2} = \begin{pmatrix} (H_1 - \mu I)^{-1} H_{12} x_2 \\ x_2 \end{pmatrix}$$

sono, rispettivamente, autovettori di H_0 relativamente a λ e ξ.

Problema

La risoluzione del sistema $H_1 - \mu I$ comporta perdita di precisione.

Poniamo allora $\tilde{x_2} = \begin{pmatrix} 0 \\ x_2 \end{pmatrix}$.

Metodo di Newton

Dato (x, λ) eigenpair approssimato, calcolare al prim'ordine (y, μ) tali che $H(x+y) = (\lambda+\mu)(x+y)$ e sommarle a (x, λ) equivale a svolgere un passo del metodo di Newton applicato a F_s:

$$\begin{pmatrix} x \\ \lambda \end{pmatrix} + \begin{pmatrix} y \\ \mu \end{pmatrix} = \begin{pmatrix} x \\ \lambda \end{pmatrix} - \begin{pmatrix} H - \lambda I & -x \\ e_s^T & 0 \end{pmatrix}^{-1} \begin{pmatrix} Hx - \lambda x \\ 0 \end{pmatrix}.$$

Soluzione

Per la formula di Sherman-Morrison-Woodbury, ad ogni passo

- calcoliamo $(H - \lambda I)^{-1} x = x'$;
- $x \leftarrow \frac{1}{x'_s} x'$ e $\lambda \leftarrow \lambda + \frac{1}{x'_s}$.

Condizioni d'arresto

L'algoritmo accetta un eigenpair (x, λ) come soluzione di F_s quando

$$\frac{||Hx - \lambda x||_2}{||H||_2 \cdot ||x||_2} < tol = ||H||_2 \cdot eps.$$

Infatti se $r = \lambda x - Hx$, data la matrice $\widehat{H} = H + \frac{1}{x^H x} r x^H$ vale

$$\frac{||\widehat{H} - H||_2}{||H||_2} = \frac{1}{||H||_2} \cdot \left|\left| \frac{r x^H}{x^H x} \right|\right|_2 \leq \frac{||Hx - \lambda x||_2}{||H||_2 \cdot ||x||_2} < tol.$$

Deflazione mediante trasformazioni elementari

Se $(x, \lambda)_i$ per $i = 1, \dots r$ sono eigenpair approssimati che causano doppioni, perdiamo $r-1$ soluzioni. È necessario ripetere le iterazioni con una matrice H' tale che $Sp(H') = Sp(H) - \{\lambda\}$ e valori iniziali $(x', \lambda)_i$ per $i = 2, \dots, r$.

Soluzione

Data $M = E(-1, x - e_n, e_n)$ e

$$\tilde{H} = M^{-1} H M = \left(\begin{array}{c|c} H' & 0 \\ \hline * & \lambda \end{array}\right),$$

la matrice H' fa al caso nostro.

Deflazione mediante trasformazioni elementari

La matrice H' differisce dalla $n-1$-esima sotto-matrice principale di testa di H solo per la sua ultima colonna, infatti se h_{n-1} denota la $n-1$-esima colonna di H e

$$x' = \begin{pmatrix} \frac{x_1}{x_n} \\ \vdots \\ \frac{x_{n-1}}{x_n} \end{pmatrix},$$

basta modificare la sotto-matrice di H solo nella sua ultima colonna, ponendola uguale a

$$h_{n-1} - h_{n,n-1}x'.$$

Calcolarla ha quindi costo lineare.

Vantaggi e svantaggi della deflazione elementare

- La matrice H' è poco costosa da calcolare
- Si può applicare anche nel caso di autovalori complessi, tuttavia se l'autovalore è complesso anche H' lo è e questo aumenta il costo computazionale
- Può essere mal condizionato

Soluzione

- Ricondursi ad un problema agli autovalori generalizzato
- Wilkinson propone di permutare opportune righe di H e dà una stima del residuo degli eigenpair proporzionale a $n||H||_\infty \cdot eps$

La struttura dell'algoritmo

Nel passo base restituiamo gli eigenpair calcolati con la routine `eig`.
Nel passo ricorsivo:

- `minmodulo`: determinare l'elemento α sulla sotto-diagonale in cui introdurre lo zero
- `dc`: determinare i valori iniziali $(x, \lambda)_i$, cioè gli eigenpair delle due sotto-matrici
- `iterazione`: per ciascun $(x, \lambda)_i$ iterare fino a convergenza:

$$\begin{pmatrix} H - \lambda I & -x \\ e_s^T & 0 \end{pmatrix} \begin{pmatrix} y \\ \mu \end{pmatrix} = \begin{pmatrix} \lambda x - Hx \\ 0 \end{pmatrix} \text{ e } \begin{cases} x \leftarrow x + y \\ \lambda \leftarrow \lambda + \mu \end{cases}$$

- `deflazione`: controllare la presenza di eventuali doppioni (`check_def`), nel caso deflazionare (`deflazione_elementare`)

La struttura dell'algoritmo

Nel passo base restituiamo gli eigenpair calcolati con la routine `eig`.
Nel passo ricorsivo:

- `minmodulo`: determinare l'elemento α sulla sotto-diagonale in cui introdurre lo zero
- `dc`: determinare i valori iniziali $(x, \lambda)_i$, cioè gli eigenpair delle due sotto-matrici
- `iterazione`: per ciascun $(x, \lambda)_i$ iterare fino a convergenza:

$$\begin{pmatrix} H - \lambda I & -x \\ e_s^T & 0 \end{pmatrix} \begin{pmatrix} y \\ \mu \end{pmatrix} = \begin{pmatrix} \lambda x - Hx \\ 0 \end{pmatrix} \text{ e } \begin{cases} x \leftarrow x + y \\ \lambda \leftarrow \lambda + \mu \end{cases}$$

- `deflazione`: controllare la presenza di eventuali doppioni (`check_def`), nel caso deflazionare (`deflazione_elementare`)

La struttura dell'algoritmo

Nel passo base restituiamo gli eigenpair calcolati con la routine `eig`.
Nel passo ricorsivo:

- `minmodulo`: determinare l'elemento α sulla sotto-diagonale in cui introdurre lo zero
- `dc`: determinare i valori iniziali $(x, \lambda)_i$, cioè gli eigenpair delle due sotto-matrici
- `iterazione`: per ciascun $(x, \lambda)_i$ iterare fino a convergenza:

$$\begin{pmatrix} H - \lambda I & -x \\ e_s^T & 0 \end{pmatrix} \begin{pmatrix} y \\ \mu \end{pmatrix} = \begin{pmatrix} \lambda x - Hx \\ 0 \end{pmatrix} \text{ e } \begin{cases} x \leftarrow x + y \\ \lambda \leftarrow \lambda + \mu \end{cases}$$

- `deflazione`: controllare la presenza di eventuali doppioni (`check_def`), nel caso deflazionare (`deflazione_elementare`)

La struttura dell'algoritmo

Nel passo base restituiamo gli eigenpair calcolati con la routine `eig`.
Nel passo ricorsivo:

- `minmodulo`: determinare l'elemento α sulla sotto-diagonale in cui introdurre lo zero
- `dc`: determinare i valori iniziali $(x, \lambda)_i$, cioè gli eigenpair delle due sotto-matrici
- `iterazione`: per ciascun $(x, \lambda)_i$ iterare fino a convergenza:

$$\begin{pmatrix} H - \lambda I & -x \\ e_s^T & 0 \end{pmatrix} \begin{pmatrix} y \\ \mu \end{pmatrix} = \begin{pmatrix} \lambda x - Hx \\ 0 \end{pmatrix} \text{ e } \begin{cases} x \leftarrow x + y \\ \lambda \leftarrow \lambda + \mu \end{cases}$$

- `deflazione`: controllare la presenza di eventuali doppioni (`check_def`), nel caso deflazionare (`deflazione_elementare`)

La struttura dell'algoritmo

Nel passo base restituiamo gli eigenpair calcolati con la routine `eig`.
Nel passo ricorsivo:

- `minmodulo`: determinare l'elemento α sulla sotto-diagonale in cui introdurre lo zero
- `dc`: determinare i valori iniziali $(x, \lambda)_i$, cioè gli eigenpair delle due sotto-matrici
- `iterazione`: per ciascun $(x, \lambda)_i$ iterare fino a convergenza:

$$\begin{pmatrix} H - \lambda I & -x \\ e_s^T & 0 \end{pmatrix} \begin{pmatrix} y \\ \mu \end{pmatrix} = \begin{pmatrix} \lambda x - Hx \\ 0 \end{pmatrix} \text{ e } \begin{cases} x \leftarrow x + y \\ \lambda \leftarrow \lambda + \mu \end{cases}$$

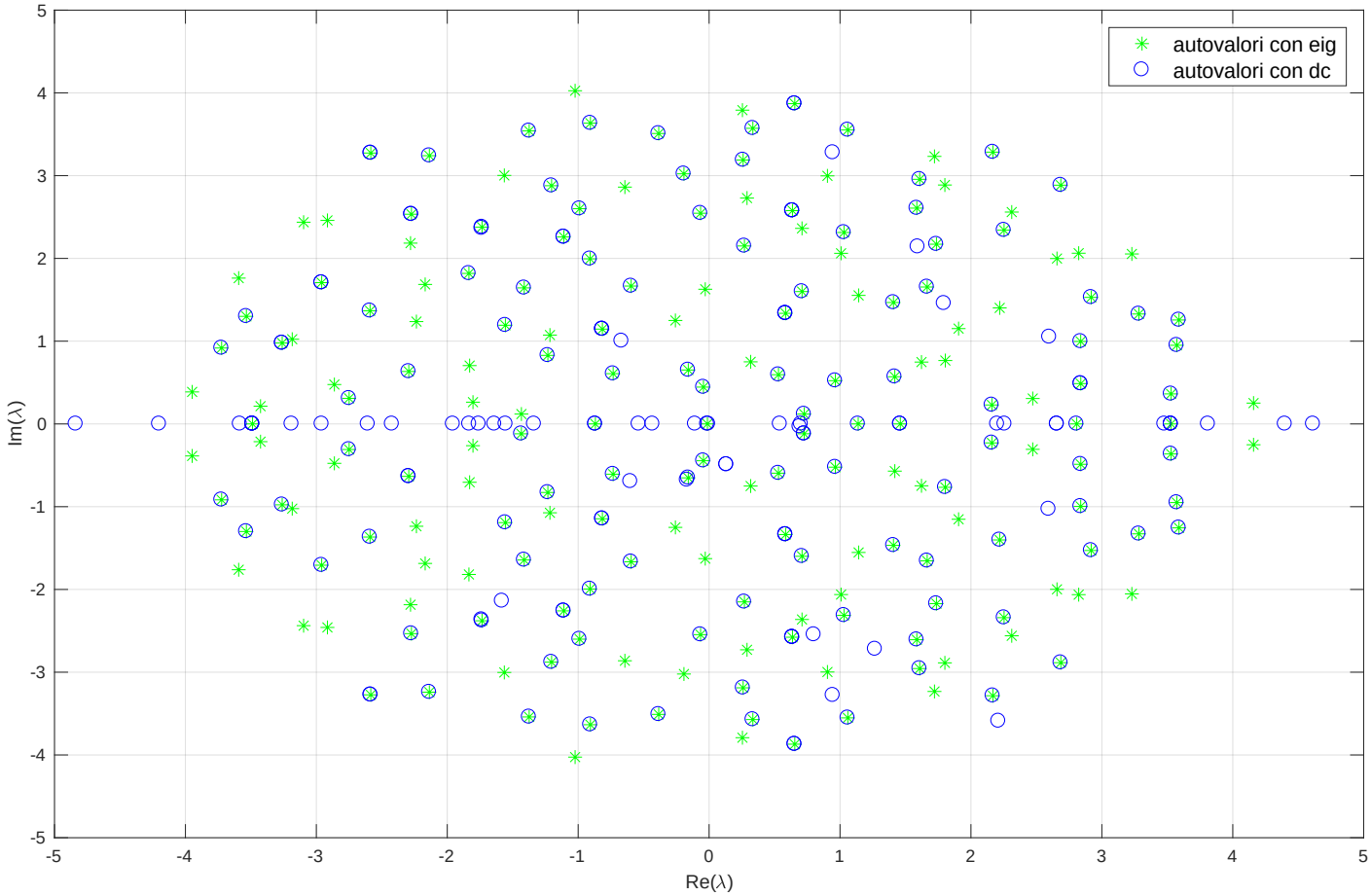
- `deflazione`: controllare la presenza di eventuali doppioni (`check_def`), nel caso deflazionare (`deflazione_elementare`)

Sperimentazione

- Scelta del passo base
- Scelta dei punti iniziali delle iterazioni
- Perdita di alcune soluzioni
- Confronto con `eig` sulla crescita del tempo di esecuzione
- Analisi teorica del costo computazionale

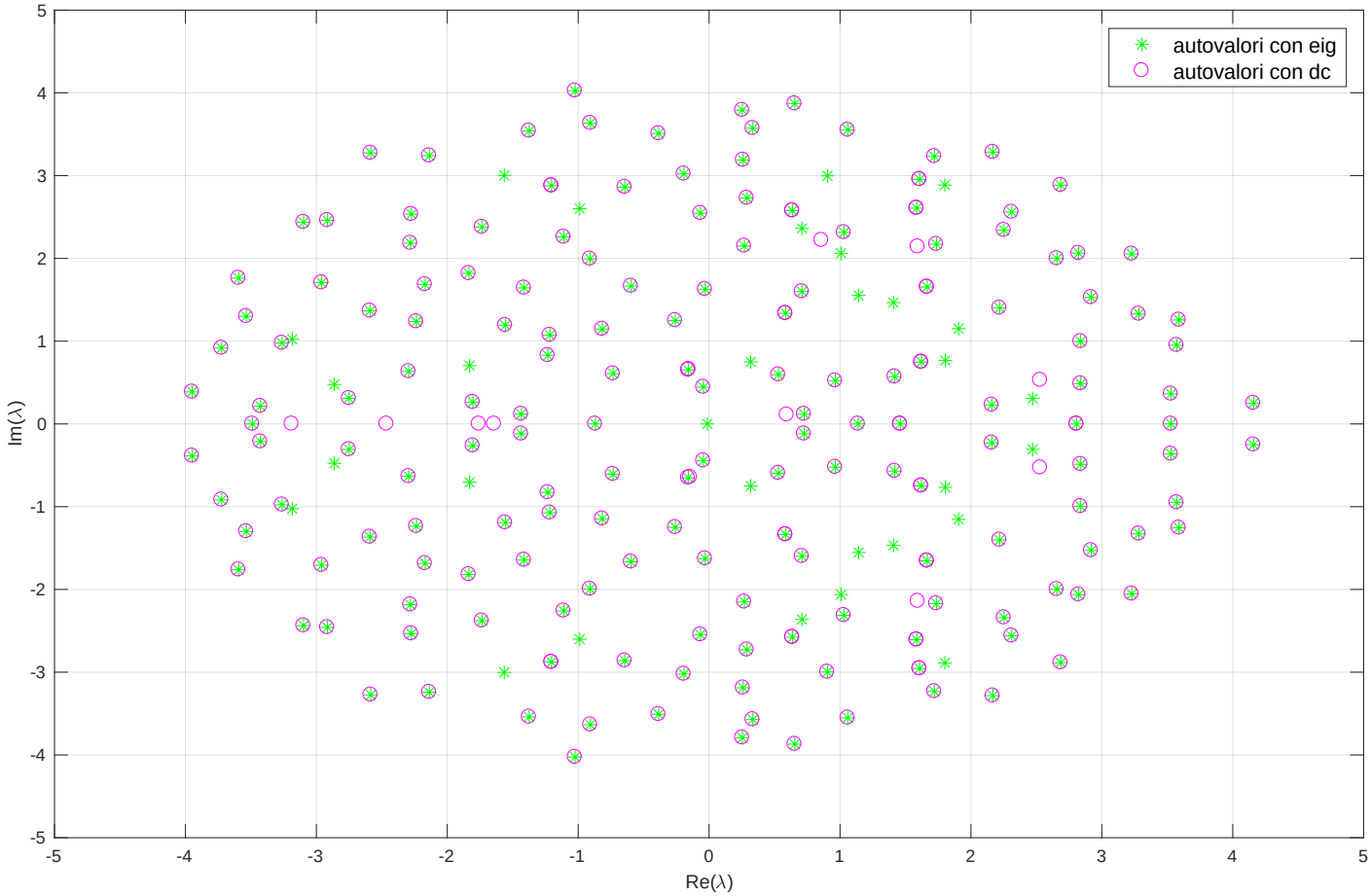
Scelta del passo base

Dimensione minima accettata pari a $\frac{n}{20}$



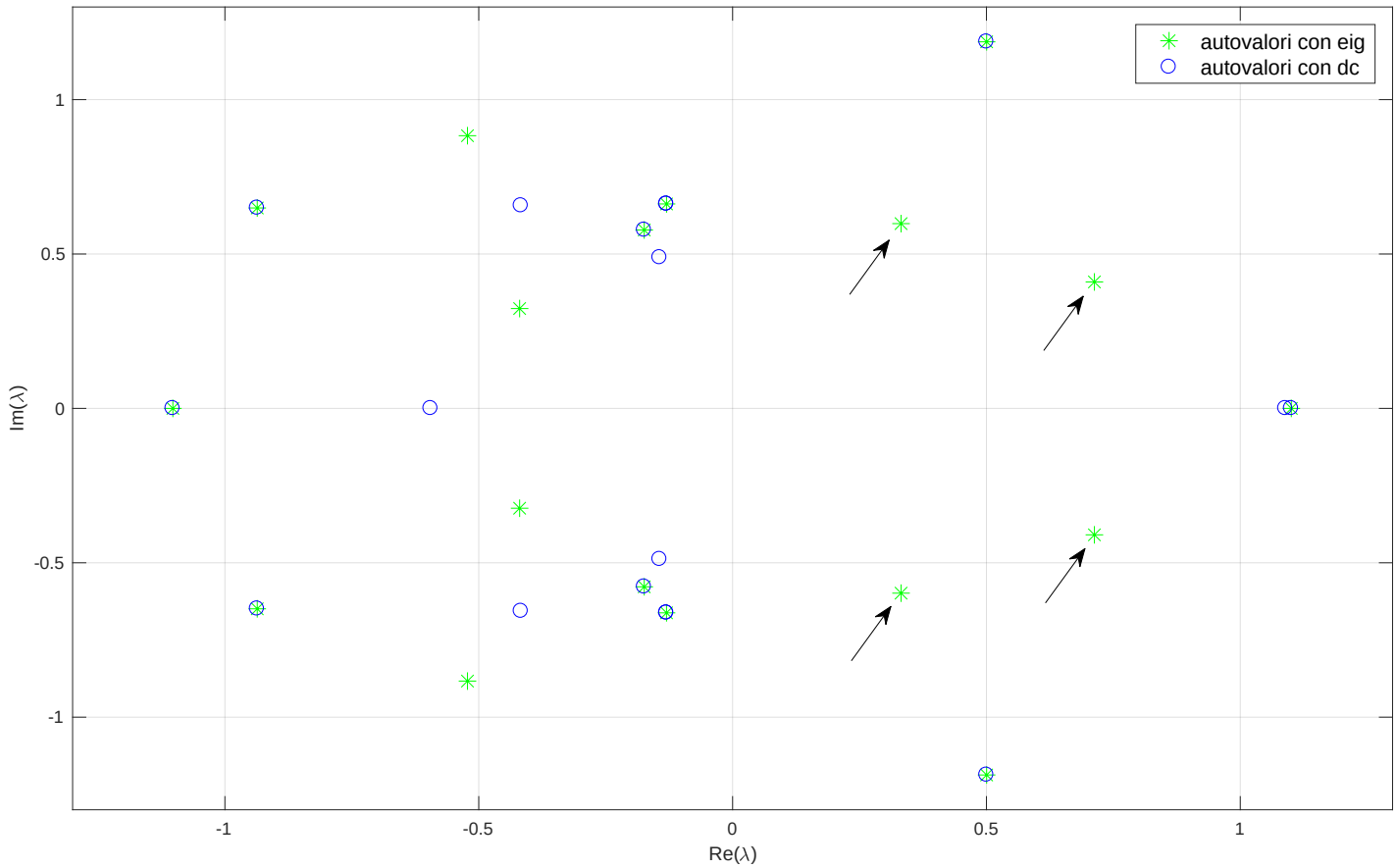
Scelta del passo base

Dimensione minima accettata pari a $\frac{n}{100}$



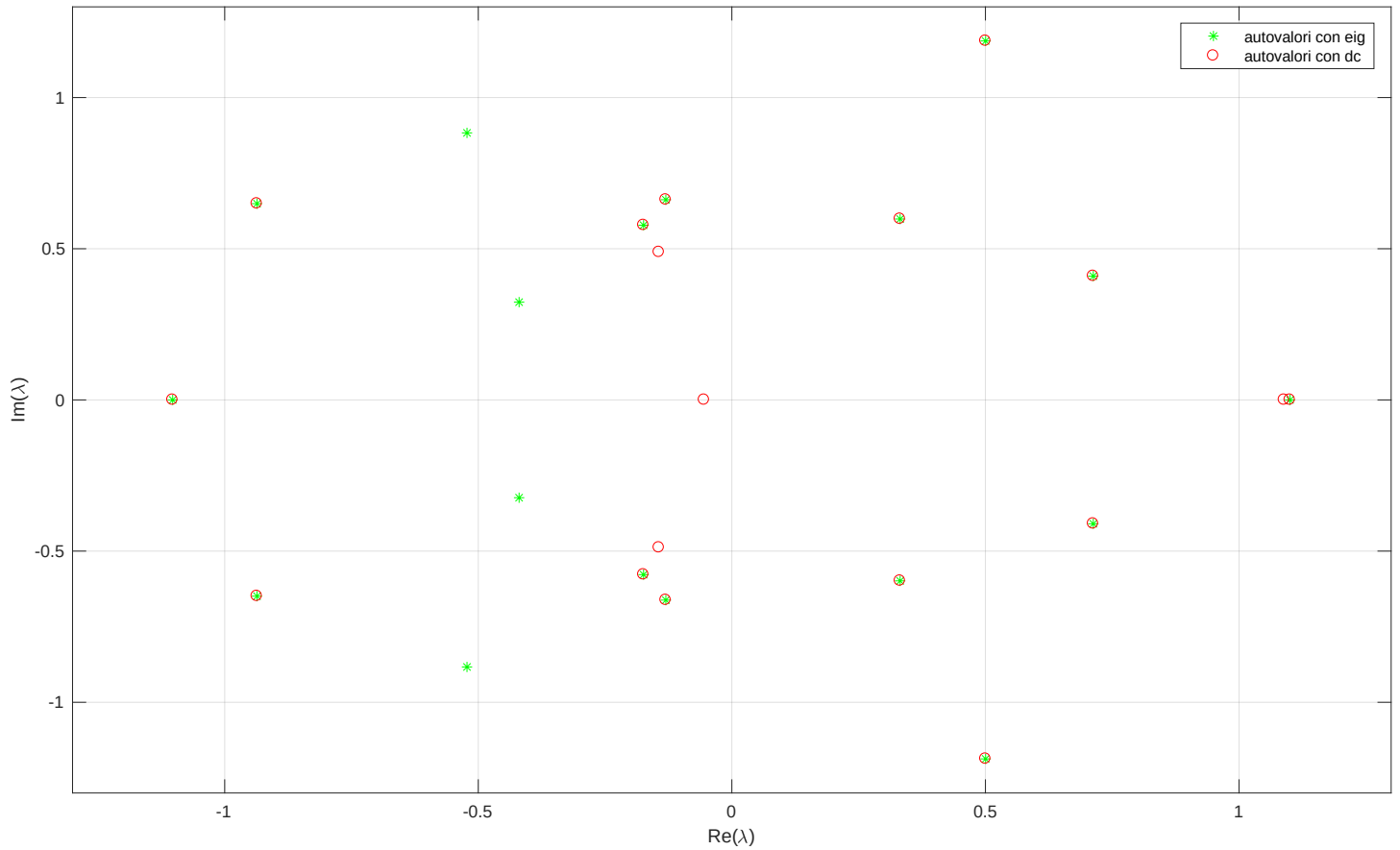
Scelta dei punti iniziali

Punti iniziali esatti

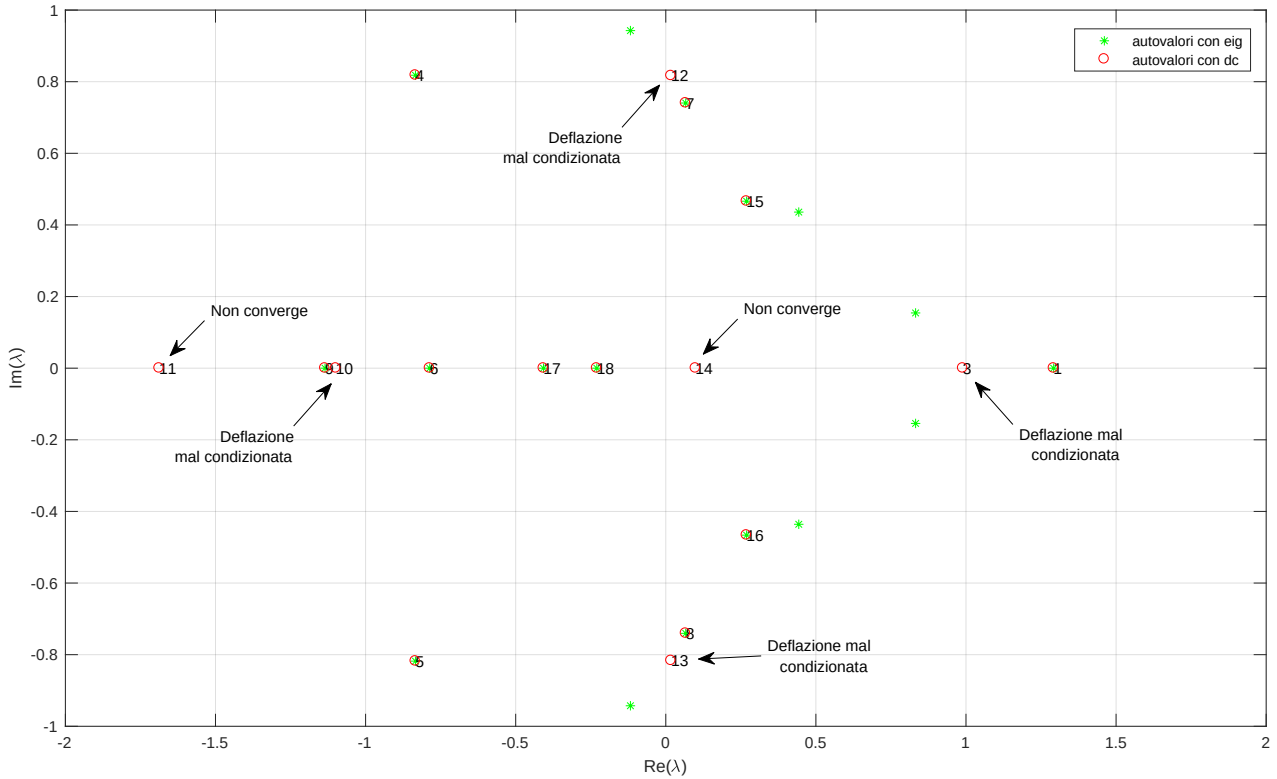


Scelta dei punti iniziali

Punti iniziali ottenuti anteponendo degli zeri



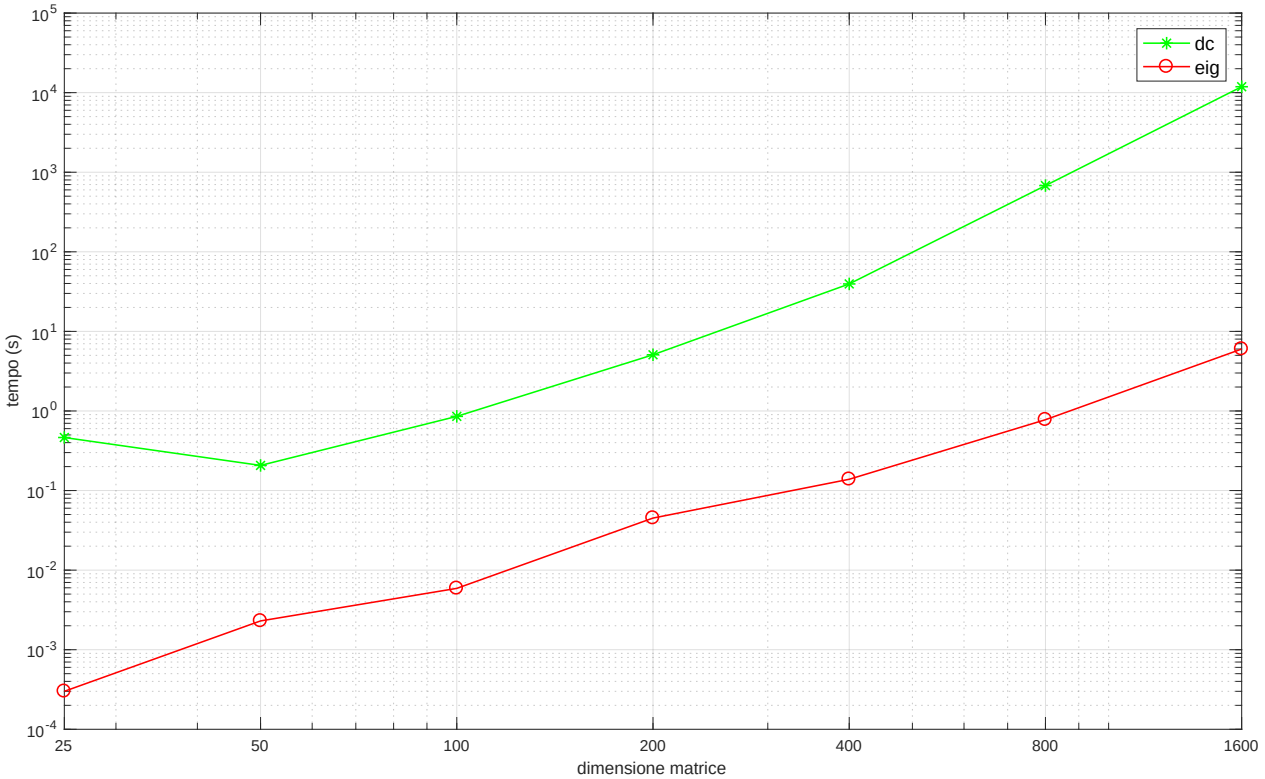
Perdita di alcune soluzioni



Autovalore	`rcond(J)`	Iterazioni	Residuo
11	0.0043	1001	0.2722
14	0.0064	1001	0.0927

Autovalore	`cond(M)`
3, 10	Inf
12, 13	Inf

Confronto tempi di esecuzione



Metodo	Rapporto fra l'i-esimo e l'$i-1$-esimo tempo					
`dc`	-1.1652	2.0477	2.5706	2.9579	4.1052	4.1268
`eig`	2.9386	1.3591	2.9343	1.6197	2.4861	2.9499

Analisi teorica del costo computazionale

Supponendo che la matrice sia di dimensione $n = 2^r$ e che lo split avvenga sempre a metà, a meno di confronti, il costo computazionale dell'algoritmo descritto per $H \in \mathbb{R}^{n \times n}$ è

$$c(n) = \begin{cases} c_0 & \text{passo base} \\ 2 \cdot c(\frac{n}{2}) + n \cdot I(n) + D(n) & \text{passo induttivo,} \end{cases}$$

dove $I(n) = P(4n^2) + 1$ e $D(n) \leq 4Pn^3 + 3n^2 + n$.
Data $f(n) = n \cdot I(n) + D(n)$, $\gamma = \frac{1}{4}$ è tale che $2f(\frac{n}{2}) \leq \gamma f(n)$, dunque per il teorema fondamentale delle ricorrenze si ha

$$c(n) = O(n^3).$$

Riassumendo

- Abbiamo descritto il metodo riconducendoci ad un problema di zeri di funzione sfruttando il metodo di Newton con opportuni punti iniziali
- Abbiamo svolto una sperimentazione sia su basse dimensioni per testarne la precisione sia su alte dimensioni per valutarne la velocità
- Tuttavia il metodo si dimostra essere poco stabile e sconveniente sia in termini di tempo che di precisione
- Probabilmente un'implementazione in parallelo riesce ad essere più efficiente e offre la possibilità di attuare migliori strategie correttive