Lezioni di Calcolo Numerico
Lezione 11: Autovalori e valori singolari di matrici

Alberto Tibaldi

8 giugno 2018

Indice

2 Metodi numerici per il calcolo di autovalori e autovettori

I metodi numerici per il calcolo degli autovalori e degli autovettori di una matrice si possono classificare in due grandi famiglie, che si differenziano per il tipo di risultato desiderato. Queste sono:

- i metodi che permettono di calcolare solo uno degli autovalori della matrice;
- i metodi che permettono di calcolare simultaneamente tutti gli autovalori della matrice.

È utile poter disporre di entrambe queste categorie di metodi. Se infatti per esempio avessimo a che fare con matrici molto grandi e, per qualche ragione, ci interessassero solo uno o due autovalori, un algoritmo appartenente alla prima famiglia sarebbe consigliabile in quanto meno dispendioso.

Verranno ora presentati un metodo appartenente alla prima categoria (più una sua variante), e un metodo appartenente alla seconda. In entrambi i casi, i metodi si baseranno su algoritmi iterativi, ossia in cui si arriva a definire delle successioni numeriche convergenti agli autovalori di interesse; per questo motivo, non sarà possibile sapere *a priori* quanti passi saranno necessari, ma si dovranno stabilire criteri di terminazione.

2.1 Metodo delle potenze

Sia $\underline{\underline{A}}$ una matrice a n righe e colonne diagonalizzabile, e siano λ_k, per $k = 1, 2, ..., n$ i suoi autovalori. Il metodo che stiamo per descrivere, detto **metodo delle potenze**, permette di calcolare solo un autovalore: **quello avente il massimo modulo**. Nonostante esistano situazioni in cui potrebbe essere importante calcolare proprio l'autovalore di massimo modulo, in generale questa ipotesi è ovviamente limitante. Nella sezione successiva introdurremo però una variante di questo metodo, che ci permetterà di calcolare un generico autovalore a partire da un *guess*, ovvero da una stima iniziale.

Poiché ci interessa l'autovalore di massimo modulo, è conveniente *etichettare* i vari autovalori per ordine decrescente di modulo, ossia definire λ_1, λ_2, e così via, in modo che:

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq ... |\lambda_2| \geq |\lambda_n| .$$

Detto a parole, il primo autovalore, λ_1, sarà quello di massimo modulo, ossia quello che valuteremo. Notiamo che la prima diseguaglianza è stretta: $|\lambda_2|$ deve essere strettamente minore di $|\lambda_1|$

affinché il metodo funzioni[1]. Data questa ipotesi, λ_1 è certamente reale: se infatti fosse complesso, allora anche il suo complesso coniugato sarebbe un autovalore, e quindi avremmo due autovalori di massimo modulo, andando contro l'ipotesi di partenza; quindi, questo metodo funziona solo con autovalori reali.

Dal momento che per ipotesi la matrice $\underline{\underline{A}}$ è diagonalizzabile, vale quanto ripreso nelle precedenti sezioni: la matrice dei suoi autovettori, $\underline{\underline{S}}$, è non singolare. In altre parole, lo spazio delle colonne di $\underline{\underline{S}}$ è una base per $\mathbb{R}^n$; detto ancora in altre parole, sommando le varie colonne di $\underline{\underline{S}}$ usando opportuni coefficienti moltiplicativi per ciascun vettore (ossia, prendendo una combinazione lineare dello spazio delle colonne), è possibile descrivere un qualsiasi vettore di n componenti, $\underline{z} \in \mathbb{R}^n$. Se invece $\underline{\underline{S}}$ fosse stata singolare, quindi di rango non massimo, non avremmo avuto la garanzia di poter scrivere un generico vettore $\underline{z}$ come combinazione lineare delle sue colonne. Quindi, data $\underline{x}_i$ la i-esima colonna di $\underline{\underline{S}}$, ovvero l'autovettore associato all'i-esimo autovalore λ_i, il generico vettore $\underline{z} \in \mathbb{R}^n$ si può scrivere come:

$$\underline{z} = \sum_{i=1}^{n} \alpha_i \underline{x}_i = \alpha_1 \underline{x}_1 + \alpha_2 \underline{x}_2 + \ldots + \alpha_n \underline{x}_n,$$

dove i vari α_i sono i coefficienti della combinazione lineare. Data dunque $\underline{\underline{A}}$ la matrice di partenza, è possibile calcolare $\underline{\underline{A}}\,\underline{z}$:

$$\underline{\underline{A}}\,\underline{z} = \underline{\underline{A}} \sum_{i=1}^{n} \alpha_i \underline{x}_i = \underline{\underline{A}} \left(\alpha_1 \underline{x}_1 + \ldots + \alpha_n \underline{x}_n \right) = \alpha_1 \underline{\underline{A}}\,\underline{x}_1 + \ldots + \alpha_n \underline{\underline{A}}\,\underline{x}_n = \sum_{i=1}^{n} \alpha_i \underline{\underline{A}}\,\underline{x}_i.$$

Questa proprietà deriva semplicemente dal fatto che stiamo parlando di applicazioni lineari, e quindi è possibile scambiare il segno di somma con la moltiplicazione per la matrice. A questo punto, dal momento che gli $\{\underline{x}_i\}$ sono gli autovettori di $\underline{\underline{A}}$, è possibile usare il truccone di *togliere la matrice e mettere l'autovalore al suo posto*! Quindi, otteniamo

$$\sum_{i=1}^{n} \alpha_i \underline{\underline{A}}\,\underline{x}_i = \sum_{i=1}^{n} \alpha_i \lambda_i \underline{x}_i.$$

A questo punto, proviamo a porci una domanda diversa: immaginiamo di moltiplicare il vettore $\underline{z}$ di partenza invece che per $\underline{\underline{A}}$, per $\underline{\underline{A}}^2$; cosa succede? Vediamo:

$$\underline{\underline{A}}^2 \underline{z} = \underline{\underline{A}} \left(\underline{\underline{A}}\,\underline{z} \right) = \underline{\underline{A}} \sum_{i=1}^{n} \alpha_i \lambda_i \underline{x}_i = \sum_{i=1}^{n} \alpha_i \lambda_i \underline{\underline{A}}\,\underline{x}_i = \sum_{i=1}^{n} \alpha_i \lambda_i \left(\lambda_i \underline{x}_i \right) = \sum_{i=1}^{n} \alpha_i \lambda_i^2 \underline{x}_i.$$

Come si può immaginare, se invece di avere $\underline{\underline{A}}^2$ avessimo $\underline{\underline{A}}^m$, otterremmo:

$$\underline{\underline{A}}^m \underline{z} = \sum_{i=1}^{n} \alpha_i \lambda_i^m \underline{x}_i = \alpha_1 \lambda_1^m \underline{x}_1 + \alpha_2 \lambda_2^m \underline{x}_2 + \ldots + \alpha_n \lambda_n^m \underline{x}_n.$$

Adesso, applichiamo un altro truccone: raccogliamo λ_1^m da questa espressione:

$$\alpha_1 \lambda_1^m \underline{x}_1 + \alpha_2 \lambda_2^m \underline{x}_2 + \ldots + \alpha_n \lambda_n^m \underline{x}_n = \lambda_1^m \underbrace{\left[\alpha_1 \underline{x}_1 + \left(\frac{\lambda_2}{\lambda_1} \right)^m \underline{x}_2 + \ldots + \left(\frac{\lambda_n}{\lambda_1} \right)^m \underline{x}_n \right]}_{\underline{y}^{(m)}} =$$

$$= \lambda_1^m \underline{y}^{(m)},$$

dove $\underline{y}^{(m)}$ è stato definito come l'intera parentesi quadra[2].

Siamo a questo punto pronti a trarre le conclusioni: a partire dalla nostra ipotesi $|\lambda_1| > |\lambda_i|$, con $i \geq 2$, i vari termini contenuti nelle parentesi quadre di $\underline{y}^{(m)}$ saranno tutti minori di 1:

[1] in qualche modo MATLAB® usa degli *sporchi trucchi* che permettono di mitigare questo problema, ma che rischiano di danneggiare le proprietà fisico/matematiche dell'autovalore; al termine della sezione aggiungeremo qualche dettaglio extra su questa cosa

[2] si noti che si è usato l'indice m tra parentesi tonde per indicare che questo è un m-esimo vettore $\underline{y}$, non un vettore elevato a potenza!

$$\frac{\lambda_i}{\lambda_1} < 1.$$

Ma, quindi, al crescere di m, si ha che

$$\lim_{m\to\infty} \left(\frac{\lambda_i}{\lambda_1}\right)^m = 0,$$

e, quindi, si avrà che

$$\lim_{m\to\infty} \underline{\underline{A}}^m \underline{z} = \lim_{m\to\infty} \lambda_1^m \underline{y}^{(m)} = \lim_{m\to\infty} \lambda_1^m \left[\alpha_1 \underline{x}_1 + \sum_{i=2}^{n} \left(\frac{\lambda_i}{\lambda_1}\right)^m\right] = \lambda_1^m \alpha_1 \underline{x}_1.$$

In altre parole, tutti i contributi degli autovettori diversi dal primo tendono a sparire per $m \to \infty$.

Possiamo a questo punto sfruttare tutto ciò che abbiamo introdotto fino a ora per proporre un algoritmo iterativo in grado di calcolare l'autovalore di massimo modulo. Sostanzialmente, ogni iterazione consisterebbe nella moltiplicazione di un certo vettore per la matrice $\underline{\underline{A}}$, e dunque sarebbe un po' come se a ogni iterazione facessimo in modo da far crescere il m del $\underline{\underline{A}}^m$. Partiamo da un $\underline{z}$ generico[3], e a forza di moltiplicarlo per $\underline{\underline{A}}$, il *metodo delle potenze* lo *ripulirà* di tutti i contributi degli autovettori che non siano quelli associati all'autovalore di massimo modulo[4]. Possiamo quindi riassumere l'algoritmo nei seguenti passi:

1. Dal momento che un autovettore è definito a meno di una costante moltiplicativa, ma anche che ogni iterazione del metodo delle potenze introduce un fattore moltiplicativo λ_1, al fine di evitare problemi di overflow o underflow è utile **normalizzare** il vettore su cui si lavora; il primo passo della m-esima iterazione dunque è definire il vettore su cui lavoriamo, $\underline{w}^{(m)}$, come

$$\underline{w}^{(m)} = \frac{\underline{z}^{(m)}}{\left\|\underline{z}^{(m)}\right\|_2}. \tag{1}$$

2. Dal momento che, al crescere di m, il vettore $\underline{w}^{(m)}$ tende proprio all'autovettore $\underline{x}_1$, ogni iterazione consiste nel definire una nuova stima dell'autovettore *più ripulita*, prendendo quello attuale e moltiplicandolo per la matrice $\underline{\underline{A}}$:

$$\underline{z}^{(m+1)} = \underline{\underline{A}}\, \underline{w}^{(m)}. \tag{2}$$

Come detto, questo è un metodo iterativo, quindi a priori non è chiaro quando ci si debba fermare. Si possono introdurre due criteri: prima di tutto, si può stabilire *a priori* di voler eseguire esattamente $m_{\max}$ iterazioni del metodo, sperando che esse siano sufficienti per avere una buona approssimazione dell'autovettore (e quindi dell'autovalore). Alternativamente, è possibile introdurre un **criterio di stop**, per esempio verificare quanto la stima dell'autovalore alla m-esima iterazione si discosti da quella alla $(m-1)$-esima iterazione: se al crescere di m questa differenza rimarrà al di sotto di una certa tolleranza, il metodo potrà essere fermato, in quanto non sta più effettuando *ripuliture*. Tuttavia, come appena descritto, il metodo delle potenze lavora sugli autovettori, dunque calcolare gli autovalori non è necessario, tant'è che né in (1) né in (2) appare λ_1. Volendo implementare un criterio di stop basato sull'autovalore, è possibile calcolare la sua stima alla m-esima iterazione[5], $\lambda_1^{(m)}$, utilizzando $\underline{w}^{(m)}$ e il quoziente di Rayleigh introdotto nelle precedenti sezioni:

[3] c'è il rischio che questo $\underline{z}$ generico non contenga, tra i vari contributi, l'autovettore $\underline{x}_1$, ossia che, nello scrivere la combinazione lineare, si abbia il coefficiente $\alpha_1 = 0$; questo, è risaputo, non è un vero problema, dal momento che la presenza di errori di arrotondamento e fenomeni del genere, di solito problematici, fanno sì da far apparire una *piccola componente* dell'autovettore $\underline{x}_1$ che, quindi, verrà esaltata dal metodo delle potenze che privilegerà lei e sopprimerà tutte le altre!

[4] quanto veloce sia questo procedimento di *ripulitura*, è determinato da quanto grande è λ_2/λ_1; se infatti $|\lambda_2| \simeq |\lambda_1|$, allora la successione decade più lentamente!

[5] e ribadisco che la notazione indica che questo è l'autovalore alla m-esima iterazione, **non** che è elevato a potenza m, perché uso le parentesi tonde all'apice per distinguere; è solo una notazione!

$$\lambda_1^{(m)} = \frac{(\underline{w}^{(m)})^{\mathrm{T}} \underline{\underline{A}}\, \underline{w}^{(m)}}{(\underline{w}^{(m)})^{\mathrm{T}} \underline{w}^{(m)}} = (\underline{w}^{(m)})^{\mathrm{T}} \underbrace{\underline{\underline{A}}\, \underline{w}^{(m)}}_{\underline{z}^{(m+1)}},$$

dove il denominatore è pari a 1 in virtù del fatto che $\underline{w}^{(m)}$ viene normalizzato in (1). Questa espressione si può ancora modificare, come suggerito nell'espressione, sostituendo (2) e arrivando a trovare

$$\lambda_1^{(m)} = (\underline{w}^{(m)})^{\mathrm{T}} \underline{z}^{(m+1)}. \tag{3}$$

Segue un esempio di codice implementante il metodo delle potenze.

```
clear
close all
clc

m_max=100; % fisso un massimo numero di iterazioni
z=[1 2 3]'; % guess iniziale per l'autovettore
toll=1e-10; % tolleranza per il calcolo dell'autovalore; quando lo scarto
            % l'autovalore rispetto alla stima al passo precedente e`
            % inferiore alla tolleranza, si puo` bloccare il metodo

A=[1 2 0;1 0 0 ;0 1 0]; % matrice di cui calcolare l'autovalore di massimo
                        % modulo

Deltalambda = inf; % per far partire il ciclo, impongo che Deltalambda sia
                   % infinito; cosi`, faremo almeno 1 iterazione!

lambda = 0; % inizializzo lambda a 0 per la prima iterazione; 0 e` una
            % scelta arbitraria

m = 1; % inizializzo indice iterazioni

while m<=m_max & Deltalambda > toll

    w = z/norm(z,2); % passo 1: normalizzo l'autovettore al passo attuale
    z=A*w; % passo 2: moltiplico A per ottenere la nuova stima dell'autovettore
    lambda(m+1) = w'*z; % quoziente di Rayleigh in forma ridotta

    % Calcolo scarto rispetto a precedente stima
    Deltalambda = abs(lambda(m+1)-lambda(m))./abs(lambda(m+1));

    m = m+1; % incremento indice iterazione

end

lambda(end) % stampo su schermo l'autovalore appena trovato
eig(A) % stampo gli autovalori trovati con il comando MATLAB per verifica
```

Prima di concludere, è opportuno rettificare e/o completare alcune affermazioni.

- Se abbiamo più autovalori uguali tra loro, ossia $\lambda_1 = \lambda_2 = ... = \lambda_k$, non solo in modulo ma anche in segno, si può vedere che il metodo delle potenze riesce, molto faticosamente, a convergere.
- Se invece $\lambda_1 = -\lambda_2$, ossia se esistono due autovalori di uguale modulo massimo ma segno opposto, in generale il metodo delle potenze non converge.
- Se λ_1 e λ_2 sono complessi coniugati, il metodo descritto non converge, anche se è possibile in qualche modo modificarlo per mitigare il problema.
- Si può dimostrare che per una matrice simmetrica $\underline{\underline{A}}$ la convergenza è più rapida, poiché va come $(\lambda_2/\lambda_1)^{2m}$: il quadrato di quella per matrici non simmetriche.

2.2 Metodo delle potenze inverse

Il grosso limite del metodo delle potenze è il fatto che esso può calcolare solamente l'autovalore di massimo modulo. Tuttavia, è possibile modificare questo metodo in modo tale da fargli approssimare non l'autovalore di massimo modulo, ma un generico λ tale che sia il più vicino possibile a una certa stima p dataci in qualche modo[6].

[6] per esempio dai cerchi di Gershgorin!

Partiamo dalla solita equazione del problema agli autovalori

$$\underline{\underline{A}}\,\underline{x} = \lambda \underline{x}$$

che ci dice che $\underline{x}$ è l'autovettore della matrice $\underline{\underline{A}}$ associato all'autovalore λ. Immaginiamo che p sia una *stima* dell'autovalore λ, ossia un numero abbastanza simile a esso; è lecito sottrarre ad ambo i membri dell'equazione appena scritta $p\underline{x}$, ottenendo

$$\underline{\underline{A}}\,\underline{x} - p\underline{x} = \lambda \underline{x} - p\underline{x},$$

che si può riscrivere, raccogliendo, come

$$\left(\underline{\underline{A}} - p\underline{\underline{I}}\right)\underline{x} = (\lambda - p)\underline{x}.$$

Questa equazione ci dice che il numero $\lambda - p$ è autovalore della matrice $\underline{\underline{A}} - p\underline{\underline{I}}$. Dal momento che gli autovalori dell'inversa di una matrice sono uguali ai reciproci degli autovalori della matrice di partenza, possiamo dire che il numero μ definito come

$$\mu = \frac{1}{\lambda - p}$$

sia autovalore della matrice $(\underline{\underline{A}} - p\underline{\underline{I}})^{-1}$, peraltro sempre con lo stesso autovettore $\underline{x}$:

$$\left(\underline{\underline{A}} - p\underline{\underline{I}}\right)^{-1}\underline{x} = \frac{1}{\lambda - p}\underline{x} \triangleq \mu \underline{x}. \tag{4}$$

Tutte queste bellissime considerazioni sugli autovalori e autovettori non sono un tentativo di ammazzare il tempo, ma hanno uno scopo ben preciso. Se infatti la stima p è abbastanza buona, allora abbiamo che $\lambda - p$ è un numero piccolo, ma quindi

$$|\mu| = \left|\frac{1}{\lambda - p}\right|$$

sarà un numero grande, dal momento che sarà il reciproco di un numero piccolo. Nel dettaglio, se p sarà una stima abbastanza buona, potremo dire che il μ corrispondente, in convergenza, sarà **l'autovalore di massimo modulo della matrice** $(\underline{\underline{A}} - p\underline{\underline{I}})^{-1}$. Ma noi, guarda caso, abbiamo appena imparato un metodo che ci permette di calcolare l'autovalore di massimo modulo di una matrice: **il metodo delle potenze**! Se quindi, invece di applicare il metodo delle potenze alla matrice $\underline{\underline{A}}$, lo applicassimo alla matrice $(\underline{\underline{A}} - p\underline{\underline{I}})^{-1}$, potremmo trovare l'autovalore più vicino alla stima p di partenza. Infatti, a partire da (4), potremmo ottenere λ, a partire da μ, come

$$\mu = \frac{1}{\lambda - p} \implies \mu(\lambda - p) = 1 \implies \mu\lambda - \mu p = 1 \implies \mu\lambda = 1 + \mu p,$$

che finalmente ci porta a

$$\lambda = p + \frac{1}{\mu}.$$

Proprio come fatto per il metodo delle potenze, possiamo sfruttare tutte queste considerazioni per scrivere un algoritmo.

1. Proprio come per il metodo delle potenze, per ragioni numeriche è conveniente normalizzare la stima dell'autovettore, lavorando quindi su un $\underline{w}^{(m)}$ normalizzato anziché su $\underline{z}^{(m)}$:

$$\underline{w}^{(m)} = \frac{\underline{z}^{(m)}}{\left\|\underline{z}^{(m)}\right\|_2}. \tag{5}$$

   Fin qui, nulla di nuovo.

2. Il metodo delle potenze inverse ha lo scopo di trovare l'autovalore μ di massimo modulo della matrice $(\underline{\underline{A}} - p\underline{\underline{I}})^{-1}$, anziché di trovare il λ_1 di $\underline{\underline{A}}$; di conseguenza, l'iterazione del metodo delle potenze inverse sarà identica alla (2), dove però al posto di $\underline{\underline{A}}$ dovremo sostituire $(\underline{\underline{A}} - p\underline{\underline{I}})^{-1}$:

$$\underline{z}^{(m+1)} = \left(\underline{\underline{A}} - p\underline{\underline{I}}\right)^{-1} \underline{w}^{(m)}.$$

Tuttavia, per quanto questa espressione sia corretta, non è molto furba da un punto di vista computazionale; infatti, questa contiene l'inversa di una matrice :-(. Ciò che invece potremmo fare è moltiplicare da sinistra ambo i membri per $(\underline{\underline{A}} - p\underline{\underline{I}})$, ottenendo

$$\left(\underline{\underline{A}} - p\underline{\underline{I}}\right) \underline{z}^{(m+1)} = \underline{w}^{(m)}. \tag{6}$$

Da un punto di vista teorico, le due espressioni sono esattamente coincidenti. Da un punto di vista pratico, però, questa è molto più interessante, dal momento che è un sistema lineare avente come soluzione la stima dell'autovettore per la successiva iterazione $\underline{z}^{(m+1)}$, avente come matrice di sistema $\left(\underline{\underline{A}} - p\underline{\underline{I}}\right)$ (che ha il vantaggio di non richiedere inversioni per essere valutata), e come termine noto la stima dell'autovettore alla precedente iterazione $\underline{w}^{(m)}$. Quindi, anziché invertire questa matrice, potremo calcolare la fattorizzazione PA = LU **una volta soltanto, come passo preliminare dell'algoritmo**, e **a ogni iterazione del metodo delle potenze inverse, sarà sufficiente risolvere due sistemi triangolari**.

Segue un esempio di codice implementante questo metodo.

```
clear
close all
clc

m_max=100; % fisso un massimo numero di iterazioni
toll=1e-10; % tolleranza per il calcolo dell'autovalore; quando lo scarto
            % l'autovalore rispetto alla stima al passo precedente e`
            % inferiore alla tolleranza, si puo` bloccare il metodo

A=[1 -2 0;0 2 0 ;1 1 3]; % matrice di cui calcolare l'autovalore di massimo
                         % modulo

z=[1 1 1]'; % guess iniziale per l'autovettore
p=0.5; % guess iniziale per l'autovalore che ci interessa approssimare

n = size(A); % calcolo la dimensione della matrice
w = z/norm(z); % normalizzo z come nel metodo delle potenze

lambda=p; % ha senso inizializzare lambda, per la prima iterazione, come il
          % guess su di esso!

Deltalambda = inf; % per far partire il ciclo, impongo che Deltalambda sia
                   % infinito; cosi`, faremo almeno 1 iterazione!

[L,U,P] = lu(A-p*eye(n)); % calcolo, una volta soltanto, la fattorizzazione
                          % PA = LU della matrice

m = 1; % inizializzo indice iterazioni

while m<=m_max & Deltalambda > toll

    w = z/norm(z); % passo 1: normalizzo l'autovettore al passo attuale
    z = U\(L\(P*w)); % passo 2: risolvo il sistema lineare (A-pI)*z = w ;
                     % grazie alla fattorizzazione LU, che ho calcolato
                     % solo una volta esternamente al ciclo, questo compito
                     % si riduce alla soluzione di due sistemi triangolari!

    % calcolo mu, il massimo autovalore di (A-p*eye(n))^(-1)
    mu=w'*z; % quoziente di Rayleigh per calcolo mu

    lambda(m+1) = p+1/mu; % formula per calcolare lambda da mu e p

    % calcolo mu, il massimo autovalore di (A-p*eye(n))^(-1)
    mu=w'*z; % quoziente di Rayleigh per calcolo mu
    lambda(m+1) = p+1/mu; % formula di slide 40

    % Calcolo scarto rispetto a precedente stima
    Deltalambda = abs(lambda(m+1)-lambda(m))./abs(lambda(m+1));
```

```
    m = m+1; % incremento indice iterazione

end
m

lambda(end) % stampo su schermo l'autovalore appena trovato
eigs(A,1,p) % con il comando eigs posso calcolare solo pochi (o in questo
            % caso solo 1 autovalore) tale per cui esso sia il piu` vicino
            % alla stima iniziale p; uso questo comando per verifica
```

Seguono ora alcune osservazioni conclusive sul metodo delle potenze inverse.

- Se scegliamo come stima iniziale $p = 0$, allora il metodo delle potenze inverse permette di approssimare l'autovalore di minimo modulo della matrice $\underline{\underline{A}}$, nell'ipotesi che esista un solo autovalore reale di minimo modulo.
- Una variante a questo algoritmo potrebbe consistere, dopo un certo numero di iterazioni, nel sostituire la stima p con un'altra stima, più raffinata, in quanto ottenuta dalle prime iterazioni del metodo. Il vantaggio potrebbe essere che, con una stima più raffinata, magari il numero di iterazioni necessarie per convergere potrebbe diminuire. Tuttavia, la controindicazione nella cosa sarebbe la necessità di ricalcolare, una volta sostituita p, la fattorizzazione PA = LU, aumentando il costo computazionale.
- Nel caso un po' sfortunato in cui $p = \lambda$, allora c'è il rischio che il numero $(p-\lambda)^{-1}$ sia infinito, e questo potrebbe causare problemi; c'è pure da dire che, se anche capitasse una cosa del genere, vorrebbe dire che non c'è nessun autovalore da trovare: lo avremmo già!

2.3 Metodo QR

Presentiamo ora, in qualità di rappresentante della famiglia dei metodi atti ad approssimare simultaneamente tutti gli autovalori, il **metodo QR**, basato sulla omonima fattorizzazione di matrici. Come i metodi delle potenze, anche questa tecnica è iterativa.

Alla prima iterazione, definiamo

$$\underline{\underline{A}}_1 = \underline{\underline{A}}.$$

Quindi calcoliamo, da $\underline{\underline{A}}_1$, la sua fattorizzazione QR:

$$\underline{\underline{A}}_1 = \underline{\underline{Q}}_1 \underline{\underline{R}}_1,$$

dove ricordiamo che $\underline{\underline{Q}}_1$ è una matrice ortogonale e $\underline{\underline{R}}_1$ una matrice triangolare superiore (assumendo che $\underline{\underline{A}}$ non sia singolare). Il metodo si basa sul definire la matrice di partenza dell'iterazione successiva, $\underline{\underline{A}}_2$, come

$$\underline{\underline{A}}_2 = \underline{\underline{R}}_1 \underline{\underline{Q}}_1,$$

e, in generale,

$$\underline{\underline{A}}_{k+1} = \underline{\underline{R}}_k \underline{\underline{Q}}_k.$$

A questo punto, notiamo un trucchetto furbo; dal momento che $\underline{\underline{Q}}_1$ è ortogonale, abbiamo che

$$\underline{\underline{Q}}_1^{\mathrm{T}} \underline{\underline{Q}}_1 = \underline{\underline{I}}.$$

Questo significa che è concesso *far apparire* dove vogliamo il prodotto $\underline{\underline{Q}}_1^{\mathrm{T}} \underline{\underline{Q}}_1$. Per esempio, nella definizione di $\underline{\underline{A}}_2$;-)

$$\underline{\underline{A}}_2 = \underline{\underline{R}}_1 \underline{\underline{Q}}_1 = \underline{\underline{I}}\, \underline{\underline{R}}_1 \underline{\underline{Q}}_1 = \underline{\underline{Q}}_1^{\mathrm{T}} \underline{\underline{Q}}_1 \underline{\underline{R}}_1 \underline{\underline{Q}}_1.$$

Tuttavia, dal momento che

$$\underline{\underline{Q}}_1 \underline{\underline{R}}_1 = \underline{\underline{A}}_1,$$

questa scrittura ci dice che

$$\underline{\underline{A}}_2 = \underline{\underline{Q}}_1^T \underline{\underline{A}}_1 \underline{\underline{Q}}_1 .$$

Questo ci dimostra che $\underline{\underline{A}}_2$ è una matrice **simile** a $\underline{\underline{A}}_1$: infatti, esiste una matrice $\underline{\underline{Q}}_1$ non singolare[7] che permette di passare da $\underline{\underline{A}}_1$ a $\underline{\underline{A}}_2$; questo è quello che di solito viene definito un *cambio di base*. Essendo $\underline{\underline{A}}_2$ simile a $\underline{\underline{A}}_1$, essa ha i suoi stessi autovalori. In generale, ovviamente, $\underline{\underline{A}}_k$ sarà simile a $\underline{\underline{A}}_{k-1}$, e così via, ma quindi anche a $\underline{\underline{A}}_1$, ossia alla matrice di partenza $\underline{\underline{A}}$.

L'utilità di questo algoritmo risiede nel fatto che, iterando questo metodo, si arriva a una matrice $\underline{\underline{A}}_\infty$ che soddisfa le seguenti proprietà:

- se la matrice $\underline{\underline{A}}$ è simmetrica, $\underline{\underline{A}}_\infty$ è diagonale, e, quindi, gli elementi sulla sua diagonale sono gli autovalori della matrice $\underline{\underline{A}}$, grazie alla proprietà di similitudine appena ricordata;
- se la matrice $\underline{\underline{A}}$ non è simmetrica, ma ha autovalori reali, allora $\underline{\underline{A}}_\infty$ è triangolare superiore e, di nuovo, gli elementi sulla sua diagonale sono gli autovalori della matrice $\underline{\underline{A}}$;
- se $\underline{\underline{A}}$ non è simmetrica e ha alcuni autovalori complessi coniugati, $\underline{\underline{A}}_\infty$ è **quasi triangolare superiore**, ossia presenta lungo la diagonale sottomatrici 1×1 contenenti gli autovalori reali di $\underline{\underline{A}}$, e sottomatrici 2×2 i cui autovalori sono autovalori complessi coniugati di $\underline{\underline{A}}$.

Questo è grosso modo l'algoritmo su cui si basa il comando `eig` di MATLAB®; nella forma qui presentata, la convergenza è garantita se gli autovalori della matrice sono distinti in modulo.

Segue infine uno script in cui viene implementato il metodo QR in questa versione di base.

```
clear
close all
clc

k_max=100; % fisso un massimo numero di iterazioni
toll=1e-14; % tolleranza per il calcolo dell'autovalore; quando lo scarto
            % l'autovalore rispetto alla stima al passo precedente e`
            % inferiore alla tolleranza, si puo` bloccare il metodo

n=10; % come esempio usiamo una matrice n x n di Hilbert
A=hilb(n); % matrice di Hilbert

% Inizializzo alcune variabili
err=inf; % errore tra gli autovalori al passo precedente e attuale
Eold=inf; % vettore degli autovettori al passo precedente
k=0; % numero di iterazioni del metodo

while(k<k_max & err>toll)

    [Q,R]=qr(A); % calcolo fattorizzazione QR
    A=R*Q; % matrice alla iterazione successiva

    E=diag(A); % la matrice A diventer diagonale o triangolare; di
               % conseguenza, i suoi autovalori diventeranno gli elementi
               % sulla diagonale

    err=norm(E-Eold,2); % calcolo errore per esempio in norma 2 rispetto
                        % autovalori stimati alla iterazione precedente

    Eold=E; % aggiorno il vettore degli autovalori rispetto al passo
            % precedente

    k=k+1; % aggiorno il contatore delle iterazioni

end

k % stampo sul prompt il numero di iterazioni
[Sref,Eref]=eig(A); % calcolo autovalori e autovettori "di riferimento"

figure
hold on
grid on
box on
plot(real(E),imag(E),'bo') % disegno gli autovalori ottenuti in parte reale
                           % e in parte immaginaria
plot(real(diag(Eref)),imag(diag(Eref)),'r*') % autovalori di riferimento
```

[7] in questo caso addirittura ortogonale!

```
xlabel('Re(\lambda)')
ylabel('Im(\lambda)')
title('Autovalori matrice')
legend('Metodo QR base','Funzione eig')
axis equal
axis square
```