Serie 3
Autovalori e Autovettori

Una questione di autovalori: Google PageRank

Dalla sua nascita, Google ha impiegato brevissimo tempo per imporsi come il più utilizzato motore di ricerca rispetto a tutti i concorrenti. Il successo è dipeso dalla accuratezza delle risposte, molto superiore a quelle di altri motori di ricerca. Google infatti stila una propria classifica (*ranking*) dell'importanza delle pagine web, per ordinare i risultati proposti; molto spesso questa classifica è proprio quella desiderata dall'utente.

L'algoritmo che ordina le pagine web per importanza è denominato *PageRank* ed è stato sviluppato da S. Brin e L. Page presso la Stanford University[1]. Il principio su cui si basa è il seguente:

- se una pagina web A ha un collegamento (*link*) verso una pagina B, questo è interpretato come un voto di A in favore di B (cioè alza B in graduatoria);
- i votanti non sono tutti uguali: il voto di chi è alto in classifica (cioè ha ricevuto molti link) vale maggiormente di quello di chi è in basso.

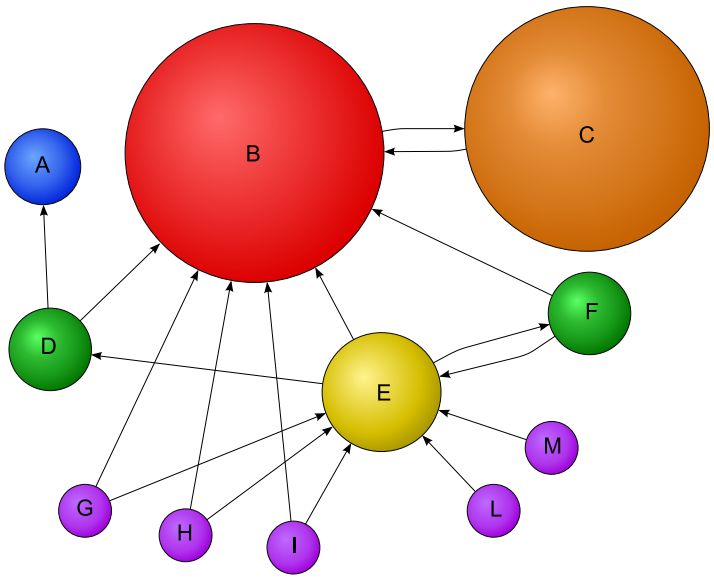

Figura 1: Diagramma schematico del *ranking*: ogni sfera rappresenta una pagina web, ogni freccia un link, la dimensione delle sfere è l'importanza attribuita alla pagina.

Questo principio è descritto schematicamente in Fig. 1. La pagina B riceve molti link e quindi è considerata molto importante. La pagina C è considerata più importante di E, anche

[1] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine", *Computer Networks and ISDN Systems*, vol. 30, no. 1-7, pp. 107– 117, 1998. Proceedings of the Seventh International World Wide Web Conference.

se riceve meno link, perchè il voto di B conta molto di più dei tanti "piccoli" voti che riceve E (ad es. se il "Washington Post" o il "New York Times" pubblicassero un link al sito web di "Calcolo Numerico ed Elementi di Analisi", questo varrebbe molto di più dei links dai siti web personali di docenti ed esercitatori del corso!).

Dal punto di vista formale, si definisce il *ranking* (importanza relativa) $r(p)$ della pagina p come la somma di tutti i *ranking* $r(q)$ di tutte le pagine q che puntano p:

$$r(p) = \sum_{q \to p} \frac{r(q)}{\#q}; \tag{1}$$

la somma è ponderata da $\#q$, numero di link presenti nella pagina q (se una pagina q ha un solo link verso p è probabile che un lettore interessato lo segua, viceversa se la pagina q ha moltissimi link è poco probabile che il lettore scelga proprio quello verso p). È facile notare che si tratta di un problema in forma implicita: per conoscere il ranking di p si devono conoscere quelli delle altre pagine, che a loro volta si basano su quello di p.

Fortunatamente il problema è affrontabile in modo relativamente semplice, una volta scritto in forma matriciale. Siano $\{p_1, p_2, \dots p_N\}$ tutte le pagine web censite e sia $A \in \mathbb{R}^{N \times N}$ la matrice di connessione, il cui elemento $a_{i,j}$ è dato da:

$$a_{i,j} = \begin{cases} \dfrac{1}{\#p_j} & \text{se esiste un link da } p_j \text{ a } p_i \\ 0 & \text{altrimenti.} \end{cases}$$

dove $\#p_j$ è il numero di link presenti sulla pagina p_j.
Si noti che gli $a_{i,j}$ possono essere interpretati come una distribuzione di probabilità, in particolare rappresentano la probabilità che un persona che clicca a caso giunga alla pagina p_i a partire dalla pagina p_j. La matrice gode della proprietà:

$$\sum_{i=1}^{N} a_{i,j} = 1. \tag{2}$$

Se il *ranking* delle pagine web p_i è rappresentato dal vettore colonna *PageRank* $\mathbf{r} = [r_1, r_2, \dots r_n]^T$, allora l'equazione (1) equivale al seguente problema:

$$\mathbf{r} = A\mathbf{r}.$$

Il *PageRank* è quindi l'autovettore corrispondente all'autovalore 1 del problema agli autovalori associato. Si può dimostrare che se i λ_i sono gli autovalori di A allora $|\lambda_i| \leq 1$. Inoltre $\lambda_1 = 1$ ha molteplicità uno[2].

Dato che il numero di pagine censite N è dell'ordine di grandezza delle decine di miliardi, il calcolo con un metodo diretto dell'autovettore *PageRank* è computazionalmente troppo oneroso, anche per chi dispone di risorse di calcolo eccezionali. Si utilizza quindi un metodo iterativo, che restituisce una soluzione approssimata, basato sul metodo delle potenze. In questo caso particolare, arrestare il numero di iterazioni al passo k significa aver considerato solo le pagine p_j che distano da p_i al più k click[3].

[2]Ciò vale sotto l'ipotesi che ogni pagina web abbia almeno un link. Un trucco per soddisfare l'ipotesi per le pagine senza link potrebbe essere quello di attribuire loro un link ad ogni pagina esistente. In questo laboratorio non si affronta questo caso.

[3]Per ragioni di semplicità di esposizione tralasciamo la possibiltà di introdurre un fattore di rilassamento (*damping fatctor*), che comunque non modifica il principio base dell'algoritmo.

1 Metodo delle potenze

Consideriamo una matrice $A \in \mathbb{C}^{n\times n}$ e supponiamo che i suoi autovalori siano così ordinati:

$$|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq |\lambda_4| \geq \cdots \geq |\lambda_n|.$$

Denotiamo con $\mathbf{x}_1$ l'autovettore di norma unitaria associato all'autovalore dominante λ_1. Se gli autovettori di A sono linearmente indipendenti, λ_1 e $\mathbf{x}_1$ possono essere calcolati tramite la seguente procedura, nota come *metodo delle potenze*:

1. si considera un vettore arbitrario $\mathbf{x}^{(0)} \in \mathbb{C}^n$ e si pone

$$\mathbf{y}^{(0)} = \mathbf{x}^{(0)}/||\mathbf{x}^{(0)}||, \quad \lambda^{(0)} = (\mathbf{y}^{(0)})^H A\mathbf{y}^{(0)};$$

2. si calcola:

$$\begin{aligned} &\text{per } k = 1, 2 \ldots \\ &\mathbf{x}^{(k)} = A\mathbf{y}^{(k-1)}, \qquad \mathbf{y}^{(k)} = \frac{\mathbf{x}^{(k)}}{||\mathbf{x}^{(k)}||}, \qquad \lambda^{(k)} = (\mathbf{y}^{(k)})^H A\mathbf{y}^{(k)}; \end{aligned}$$

   dove $\mathbf{y}^H = \overline{\mathbf{y}}^T$.

3. si termina l'algoritmo quando $|\lambda^{(k)} - \lambda^{(k-1)}| < \epsilon|\lambda^{(k)}|$, dove ϵ è una tolleranza assegnata.

Questo metodo genera una successione di vettori $\{\mathbf{y}^{(k)}\}$ di norma unitaria tali da allinearsi alla direzione dell'autovettore $\mathbf{x}_1$, per $k \to \infty$. Si dimostra inoltre che $\lambda^{(k)} \to \lambda_1$, per $k \to \infty$. Si noti che il calcolo di $\lambda^{(0)}$ serve soltanto per testare la condizioni di uscita al passo $k = 1$.

Esercizio 1

1. Costruire una matrice di connessione $A \in \mathbb{R}^{100\times 100}$ (dove $N = 100$ rappresenta il numero di pagine) che soddisfa la proprietà (2).
   (*Suggerimento*: scrivere una matrice di elementi $a_{i,j} = 0, 1$ dove uno indica l'esistenza di un link dalla pagina p_j alla pagina p_i e zero l'assenza di link. Se non si vuole studiare un caso reale, questa matrice può essere generata in modo casuale tramite la funzione Matlab® `randi`. Successivamente modificare A (normalizzando le colonne) in modo che rispetti la (2)).

2. Costruire una matrice di connessione $B \in \mathbb{R}^{5\times 5}$ relativa a 5 pagine web rappresentato in Fig. 2 (si segua il suggerimento al punto precedente).

3. Scrivere una funzione `eigpower` che implementa il metodo delle potenze per la ricerca dell'autovalore di modulo massimo e dell'autovettore associato. Una possibile intestazione potrebbe essere:

```
function [lambda,x,iter]=eigpower(A,tol,nmax,x0)
```

   dove i parametri di ingresso sono:

   - la matrice `A`;
   - la tolleranza `tol`;

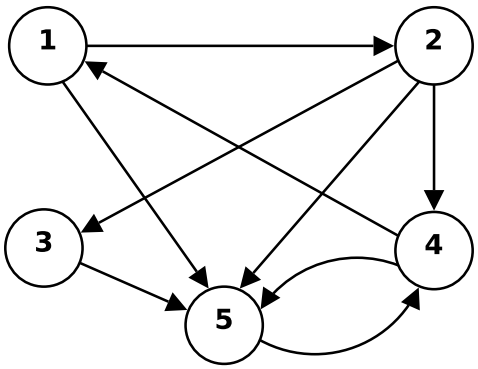

Figura 2: Schema di un web di 5 pagine. Ogni cerchio rappresenta una pagina web, ogni freccia un link.

- il numero massimo di iterazioni `nmax`;
- il vettore iniziale `x0`.

I parametri di uscita associati sono l'autovalore di modulo massimo `lambda`, l'autovettore associato `x` ed il numero di iterazioni effettuate `iter`.

Si noti che il comando `'` di Matlab® applicato a vettore o matrice ritorna il trasposto complesso coniugato (ovviamente se i vettori o matrici sono a valori reali, questa operazione coincide con la semplice trasposizione); per esempio, dato un vettore $\mathbf{v}$, eseguire `v'` in Matlab® equivale a calcolare $\mathbf{v}^H = \overline{\mathbf{v}}^T$.

4. Tramite la funzione `eigpower` si verifichi che A e B ammettono 1 come autovalore di modulo massimo e si calcoli il corrispondente autovettore (*PageRank*). Si assuma una tolleranza di 10^{-6}, un numero massimo di iterazioni pari a 100 ed un vettore iniziale $\mathbf{x}^{(0)} = (1/N, 1/N, \ldots 1/N)^T$.

2 Metodo delle potenze inverse

Dato che gli autovalori della matrice A^{-1} sono i reciproci di quelli di A, è possibile utilizzare il metodo delle potenze per approssimare anche l'autovalore di A di modulo minimo λ_n ($0 < |\lambda_n| < |\lambda_{n-1}| \leq |\lambda_{n-2}| \leq \cdots \leq |\lambda_1|$). Si tratta del *metodo delle potenze inverse*:

1. si considera un vettore arbitrario $\mathbf{x}^{(0)} \in \mathbb{C}^n$ e si pone

$$\mathbf{y}^{(0)} = \mathbf{x}^{(0)}/||\mathbf{x}^{(0)}||, \quad \mu^{(0)} = (\mathbf{y}^{(0)})^H A^{-1}\mathbf{y}^{(0)};$$

2. si calcola:

$$\begin{aligned} &\text{per } k = 1, 2 \ldots \\ &\mathbf{x}^{(k)} = A^{-1}\mathbf{y}^{(k-1)}, \qquad \mathbf{y}^{(k)} = \frac{\mathbf{x}^{(k)}}{||\mathbf{x}^{(k)}||}, \qquad \mu^{(k)} = (\mathbf{y}^{(k)})^H A^{-1}\mathbf{y}^{(k)}; \end{aligned}$$

   dove $\mathbf{y}^H = \overline{\mathbf{y}}^T$.

3. si termina l'algoritmo quando $|\mu^{(k)} - \mu^{(k-1)}| < \epsilon|\mu^{(k)}|$, dove ϵ è una tolleranza assegnata.

Il metodo fornisce un'approssimazione di $\mu = 1/\lambda_n$, cioè $1/\mu^{(k)} \to \lambda_n$ per $k \to \infty$.

La matrice A^{-1} non viene realmente calcolata. Ad ogni passo si risolve invece il sistema lineare $A\mathbf{x}^{(k)} = \mathbf{y}^{(k-1)}$. Potrebbe quindi essere conveniente fattorizzare una volta sola, inizialmente, la matrice A con una opportuna fattorizzazione, così che ad ogni iterazione vengano risolti sistemi più semplici. Si noti inoltre che gli autovettori di A ed A^{-1} sono gli stessi ed è quindi possibile (e molto conveniente) scrivere il quoziente di Rayleigh che definisce $\mu^{(k)}$ come $(\mathbf{y}^{(k)})^H A\mathbf{y}^{(k)}$, cioè con la matrice A al posto di A^{-1}; in questo modo $\mu^{(k)}$ convergerà direttamente a λ_n, per $k \to \infty$, e non al suo reciproco. L'algoritmo delle *metodo delle potenze inverse* si scrive dunque in maniera più conveniente come segue.

1. si considera un vettore arbitrario $\mathbf{x}^{(0)} \in \mathbb{C}^n$ e si pone

$$\mathbf{y}^{(0)} = \mathbf{x}^{(0)}/||\mathbf{x}^{(0)}||, \quad \lambda^{(0)} = (\mathbf{y}^{(0)})^H A\mathbf{y}^{(0)};$$

2. si calcola:

$$\begin{aligned} &\text{per } k = 1, 2 \dots \\ &\text{risolvere } A\,\mathbf{x}^{(k)} = \mathbf{y}^{(k-1)}, \qquad \mathbf{y}^{(k)} = \frac{\mathbf{x}^{(k)}}{||\mathbf{x}^{(k)}||}, \qquad \lambda^{(k)} = (\mathbf{y}^{(k)})^H A\mathbf{y}^{(k)}; \end{aligned}$$

   dove $\mathbf{y}^H = \overline{\mathbf{y}}^T$.

3. si termina l'algoritmo quando $|\lambda^{(k)} - \lambda^{(k-1)}| < \epsilon|\lambda^{(k)}|$, dove ϵ è una tolleranza assegnata.

Osseviamo dunque che, per $k \to +\infty$, abbiamo $\lambda^{(k)} \to \lambda_n$ e $\mathbf{y}^{(k)} \to \mathbf{x}_n$, l'autovettore associato all'autovalore λ_n di modulo minimo di A.

Esercizio 2

1. Scrivere una funzione `invpower` che implementa il metodo delle potenze per la ricerca dell'autovalore di modulo minimo e dell'autovettore associato. Una possibile intestazione potrebbe essere:

```
function [lambda,x,iter]=invpower(A,tol,nmax,x0)
```

   dove `lambda` è l'autovalore di modulo minimo, mentre gli altri parametri di ingresso e di uscita hanno lo stesso significato attribuito dalla funzione `eigpower`. Si risolva il sistema lineare $A\mathbf{x}^{(k)} = \mathbf{y}^{(k-1)}$ tramite la fattorizzazione LU della matrice A e il metodo di sostituzione in avanti e all'indietro (utilizzare le funzioni `fwsub` e `bksub` implementate nei laboratori della Serie 2).

2. Applicare la funzione scritta alla matrice ottenuta con il comando `toeplitz( 1 : 4 )`, usando tolleranza pari a 10^{-6}, numero massimo di iterazioni pari a 100 e vettore iniziale $\mathbf{x}^{(0)} = (1, 2, 3, 4)^T$. Provare poi ad usare la funzione `invpower` ponendo $\mathbf{x}^{(0)} = (1, 1, 1, 1)^T$ e commentare il risultato.

3 Metodo delle potenze inverse con *shift*

Il *metodo delle potenze inverse* può essere utilizzato anche per calcolare l'autovalore più vicino ad un dato numero $\mu \in \mathbb{C}$.

Denotiamo con $\lambda_\mu(A)$ tale autovalore; costruendo la matrice $M = A - \mu I$ notiamo che M ha autovalori $\lambda(M) = \lambda(A) - \mu$. Pertanto, per calcolare $\lambda_\mu(A)$ è sufficiente approssimare l'autovalore di M $\lambda_{\min}(M)$ di modulo minimo con il metodo delle potenze inverse e quindi porre $\lambda_\mu(A) = \lambda_{\min}(M) + \mu$.

Il numero μ viene detto *shift* e il metodo viene di conseguenza denominato *metodo delle potenze inverse con shift*. L'algoritmo è il seguente:

1. si considera un vettore arbitrario $\mathbf{x}^{(0)} \in \mathbb{C}^n$ e si pone $\mathbf{y}^{(0)} = \mathbf{x}^{(0)}/||\mathbf{x}^{(0)}||$ e $\lambda^{(0)} = (\mathbf{y}^{(0)})^H A \mathbf{y}^{(0)}$;
2. si pone $M = A - \mu I$;
3. si calcola, per $k = 1, 2, \ldots$
$$\mathbf{x}^{(k)} = M^{-1}\mathbf{y}^{(k-1)}, \qquad \mathbf{y}^{(k)} = \frac{\mathbf{x}^{(k)}}{||\mathbf{x}^{(k)}||}, \qquad \lambda^{(k)} = (\mathbf{y}^{(k)})^H A \mathbf{y}^{(k)};$$
4. si termina l'algoritmo quando $|\lambda^{(k)} - \lambda^{(k-1)}| < \varepsilon|\lambda^{(k)}|$, dove ε è una tolleranza assegnata.

Come nel metodo delle potenze inverse, la matrice M^{-1} non è realmente calcolata, ma ad ogni passo si risolve il sistema lineare $M\mathbf{x}^{(k)} = \mathbf{y}^{(k-1)}$. Anche in questo caso è conveniente fattorizzare una volta sola, inizialmente, la matrice M con una opportuna fattorizzazione, così che ad ogni iterazione vengano risolti sistemi più semplici. Si noti inoltre che gli autovettori di M^{-1} coincidono con quelli di A.

Localizzazione geometrica degli autovalori

Per applicare il metodo delle potenze inverse con *shift* è necessario localizzare geometricamente gli autovalori di A nel piano complesso. A tal fine possiamo utilizzare i cerchi di Gershgorin, definiti come:

$$\mathcal{R}_i = \{z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|\}$$
$$\mathcal{C}_i = \{z \in \mathbb{C} : |z - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ji}|\}.$$

$\mathcal{R}_i$ è chiamato l'i-esimo *cerchio riga*, mentre $\mathcal{C}_i$ l'i-esimo *cerchio colonna*. Definendo

$$\mathcal{S}_\mathcal{R} = \bigcup_{i=1}^{n} \mathcal{R}_i \quad \text{e} \quad \mathcal{S}_\mathcal{C} = \bigcup_{i=1}^{n} \mathcal{C}_i,$$

rispettivamente, le zone del piano determinate dall'unione dei cerchi riga (colonna), valgono le seguenti proprietà:

- tutti gli autovalori di A appartengono alla regione $\mathcal{S}_\mathcal{R} \bigcap \mathcal{S}_\mathcal{C}$;
- se $\mathcal{S}_1 = \bigcup_{i=1}^{m} \mathcal{R}_i$ e $\mathcal{S}_2 = \bigcup_{i=m+1}^{n} \mathcal{R}_i$ sono disgiunti ($\mathcal{S}_1 \bigcap \mathcal{S}_2 = \emptyset$), allora $\mathcal{S}_1$ contiene esattamente m autovalori ed $\mathcal{S}_2$ i restanti $n - m$.

Tale proprietà può essere equivalentemente formulata sui cerchi colonna. In particolare, quest'ultima proprietà evidenzia che se un cerchio riga (o colonna) di Gershgorin è isolato, allora contiene uno ed un solo autovalore.

Esercizio 3

Si consideri la matrice:

$$B = \begin{bmatrix} 10 & -1 & 1 & 0 \\ 1 & 1 & -1 & 3 \\ 2 & 0 & 2 & -1 \\ 3 & 0 & 1 & 5 \end{bmatrix}$$

1. Usare la `function gershcircles` per localizzare gli autovalori.
2. Calcolare gli autovalori con la `function eig` di Matlab® .
3. Usare la `function invpower` per calcolare l'autovalore di modulo minimo.
4. Osservato che la `function invpower` non converge e che gli autovalori di modulo minimo sono complessi e coniugati:
   - scrivere la `function invpowershift` che implementa il metodo delle potenze inverse con *shift* (a tal fine, modificare opportunamente la `function invpower`);
   - trovare *shift* opportuni che permettano di calcolare gli autovalori di modulo minimo;
   - utilizzando un valore di *shift* opportuno dedotto dai cerchi di Gershgorin, utilizzare la `function invpowershift` per calcolare l'autovalore di modulo massimo, e confrontare il numero di iterazioni necessario con il metodo delle potenze dirette.

4 Calcolo dello spettro di una matrice

I metodi che consentono di approssimare simultaneamente tutti gli autovalori sono basati sull'idea di trasformare A in una matrice simile (ovvero che ha gli stessi autovalori) ma con struttura triangolare, in modo che gli autovalori siano rappresentati dagli elementi diagonali. A tal fine, l'algoritmo di riferimento (implementato anche nella `function eig` di Matlab®) è il metodo di *iterazione QR*, che consiste nel costruire una successione di matrici $A^{(k)}$ simili alla matrice A, secondo il seguente algoritmo: posto $A^{(0)} = A$, si calcolano per $k = 0, 1, \ldots,$ con la fattorizzazione QR (`function qr` presente in Matlab®) le matrici quadrate $Q^{(k+1)}$ e $R^{(k+1)}$ tali che

$$Q^{(k+1)} R^{(k+1)} = A^{(k)}$$

e si pone quindi $A^{(k+1)} = R^{(k+1)} Q^{(k+1)}$.

Si può dimostrare che le matrici $A^{(k)}$ sono tutte simili tra loro e, di conseguenza, essendo $A^{(0)} = A$, hanno in comune con quest'ultima gli autovalori. Inoltre, se tutti gli autovalori di A sono reali e distinti in modulo,

$$\lim_{k\to\infty} A^{(k)} = \begin{bmatrix} \lambda_1 & a_{1,2}^{(k)} & \cdots & \cdots & a_{1,n}^{(k)} \\ 0 & \lambda_2 & a_{2,3}^{(k)} & & \vdots \\ & \ddots & \ddots & \ddots & \vdots \\ & & \ddots & \lambda_{n-1} & a_{n-1,n}^{(k)} \\ 0 & & & 0 & \lambda_n \end{bmatrix}$$

ovvero $A^{(k)}$ tende a diventare triangolare superiore. Si decide quindi di arrestare l'algoritmo quando i termini sottodiagonali diventano piccoli, ovvero quando $\max_{i>j} |a_{i,j}{}^{(k)}| \leq \varepsilon$, dove ε è una tolleranza fissata. La velocità di convergenza a zero dei coefficienti $a_{i,j}^{(k)}$ con $i > j$ e $k \to \infty$ dipende da $\max_i |\lambda_{i+1}/\lambda_i|$. Sotto l'ulteriore ipotesi che A sia simmetrica, $A^{(k)}$ tende a una matrice diagonale.
N.B.: Al contrario dei metodi delle potenze, questo metodo non fornisce una approssimazione degli autovettori di A.

Esercizio 4

1. Si scriva la `function qrbasic` che implementa l'algoritmo di *iterazione QR* per il calcolo di tutti gli autovalori di una generica matrice A. L'intestazione dell'algoritmo sarà

   ```
   function D = qrbasic(A,tol,nmax)
   ```

   dove `D` è un vettore contenente i valori degli autovalori calcolati, `A` è la matrice in ingresso, `tol` è la tolleranza per il test di arresto e `nmax` è il numero massimo di iterazioni. Si calcoli la fattorizzazione QR necessaria per l'algoritmo tramite la `function qr` di Matlab® .

2. Si consideri la seguente famiglia di matrici:

$$A(\alpha) = \begin{bmatrix} \alpha & 2 & 3 & 13 \\ 5 & 11 & 10 & 8 \\ 9 & 7 & 6 & 12 \\ 4 & 14 & 15 & 1 \end{bmatrix}, \qquad \alpha \in \mathbb{R}$$

   - Con $\alpha = 30$, si effettui il calcolo degli autovalori di $A(30)$ con la `function qrbasic`. Si riporti il numero di iterazioni necessario per giungere a convergenza imponendo una tolleranza pari a 10^{-10}.
   - Con la stessa tolleranza, ma $\alpha = -30$, si effettui il calcolo degli autovalori di $A(-30)$. Si riporti il numero di iterazioni necessario per giungere a convergenza, e si commenti il risultato ottenuto alla luce delle stime teoriche.

Esercizio 5

Una delle applicazioni della Singular Value Decomposition (SVD) è il suo utilizzo nell'*image compression*, un insieme di tecniche che permettono di rappresentare una generica immagine con una sua versione che occupi meno spazio in memoria. In generale (escluse le cosiddette compressioni *lossless*), questo risultato viene ottenuto a scapito della qualità dell'immagine ottenuta, ovvero tralasciando una parte dell'informazione contenuta nell'immagine di partenza.

Consideriamo ora per semplicità esclusivamente immagini in scala di grigi, in cui il valore di ogni singolo pixel è un numero positivo che rappresenta la luminosità del pixel stesso (al valore 0 corrisponde dunque il colore nero). In questo contesto, possiamo interpretare un'immagine A di dimensione m pixel per n pixel come una matrice $A \in \mathbb{R}^{m \times n}$.

1. Caricare in memoria, in Matlab, una delle immagini del dataset di esempio (e.g.: `street1.jpg`, `street2.jpg`, `peppers.png`, `ngc6543a.jpg`), utilizzando i seguenti comandi:

```
IM = imread( "nomefile.png" ); % carico l'immagine
IM = rgb2gray( IM ); % converto l'immagine in scala di grigi
imshow( IM ) % visualizzo l'immagine
A = double( IM ); % converto in formato a virgola mobile
```

2. Si calcoli la SVD della matrice A utilizzando il comando Matlab `svd`, ottenendo così le matrici $U \in \mathbb{R}^{m \times m}$, $S \in \mathbb{R}^{m \times n}$, e $V \in \mathbb{R}^{n \times n}$ tali che $USV^T = A$. Definita ora A_k la matrice ridotta di rango $k \leq \min\{m, n\}$ (per A a rango pieno) ottenuta troncando i fattori della SVD di A, si assembli tale matrice per il valore $k = 10$, per esempio.

3. Si confronti l'immagine così ottenuta con l'immagine di partenza. N.B.: per visualizzare A_k con il comando `imshow` è necessario prima convertire A_k con il comando `uint8`.

4. Si calcoli il *rapporto di compressione* ottenuto, inteso come il rapporto tra lo spazio in memoria occupato da A e quello occupato dalla sua SVD troncata al rango k.

Esercizio aggiuntivo

Esercizio 6

Si consideri la matrice (simmetrica e definita positiva):

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}$$

Si definisca un algoritmo per stimare il numero di condizionamento $K(A)$ della matrice A che preveda l'uso dei metodi delle potenze e potenze inverse (si osservi che per A simmetrica e definita positiva, allora $K(A) = \frac{\lambda_{max}}{\lambda_{min}}$, dove λ_{max} e λ_{min} sono rispettivamente il massimo e minimo degli autovalori di A). In particolare:

1. Si scriva la `function sdpcond` che implementa l'algoritmo iterativo per la stima del numero di condizionamento $K(A)$ di una generica matrice simmetrica e definita positiva. L'intestazione dell'algoritmo sarà

   ```
   function K = sdpcond(A, tol, nmax)
   ```

   dove `A` è la matrice in ingresso, `tol` è la tolleranza per un opportuno test di arresto, e `nmax` è il numero massimo di iterazioni.

2. Si usi la `function sdpcond` per la stima di $K(A)$, utilizzando un numero massimo di iterazioni pari a 200 e una tolleranza di 10^{-8}.

3. Considerata una generica matrice simmetrica e definita positiva di dimensione $n \times n$, si stimi il numero di operazioni effettuato per l'approssimazione del suo numero di condizionamento tramite l'algoritmo implementato in funzione della dimensione n del sistema e del numero di iterazioni k effettuate.