Autovalori

Esercitazione facoltativa in Matlab

Analisi del comportamento degli autovalori di sistemi arbitrari perturbati

Autovalori

Esercitazione facoltativa in Matlab

Esercizio 1

Presa una matrice quadrata A di dimensione n=73 ottenuta tramite il comando:

```
A=diag(ones(1, n-1), 1) + eye(n);
```

(che genera una matrice triangolare superiore con la diagonale principale e la prima *sovradiagonale* non nulle) si chiedeva di determinare una seconda matrice B, pari alla prima ma perturbata del valore 10^{-n} nella posizione *(n, 1)*, questa parte nel nostro codice viene lasciata alla funzione:

```
function B = createPertubate(A, n)
    % Create pertubation:
    E = zeros(size(A));
    E(n, 1) = 10^-n;
    % Create pertubated Matrix:
    B = A+E;
end
```

***createPertubate** crea una matrice **E** di dimensioni pari ad **A**, modifica quindi il valore nella posizione stabilita dal testo con il valore della perturbazione e quindi la somma ad A per ottenere la matrice perturbata **B**.*

Non sono bello ma patcho

La relazione e il codice dell'esercitazione sono state portate a termine da:

Andrea Storace:

4186140

Andrea Straforini:

4338710

Elisa Zazzera:

4380663

Si chiedeva infine di calcolare gli autovalori delle matrici A e B e quindi confrontare le due matrici in termini di **autovalori** e **norma**, questo task viene eseguito dalla funzione:

```
function [Ceig, norm_comparison, T] = compute_eig_n_norm(A, B, n)
% Computing A eigenvalues:
Aeig = eig(A);
% Computing B eigenvalues:
Beig = eig(B);
% eigenvalue comparison vector:
Ceig = zeros(size(Beig));

% compute eigenvalues comparison:
for i=1:n
    Ceig(i) = (Beig(i)-Aeig(i));
end
% Computing norms:
norm1 = norm(B-A)/norm(A);
norm2 = norm(Beig-Aeig)/norm(Aeig);

% Computing norm comparison:
norm_comparison = norm1-norm2;
T = table(Aeig, Beig, Ceig);
end
```

***conpute_eig_n_norm** calcola e confronta gli autovalori delle matrici **A** e **B** passate come argomento alla funzione, come metodo di confronto abbiamo preferito una semplice valutazione dell'errore assoluto data la dimensione della perturbazione, infatti il confronto sul valore dell'errore relativo avrebbe probabilmente restituito valori inconsistenti. Al contrario il confronto che viene poi eseguito tra le norme matriciali è di tipo relativo.*

Quindi di ripetere lo stesso confronto per le matrici A^tA e B^tB.

Dal momento che la dimensione della perturbazione non viene sentita dalla macchina poiché si ha una precisione

```
eps=2.2204e-16
```

a fronte di una perturbazione dell'ordine di 10^{-75} come si può vedere da una parte dei valori restituiti dal programma nella figura 1

Aeig	Beig	Ceig
1	1	0
1	1	0
1	1	0
1	1	0
1	1	0
...	...	...
1	1	0
1	1	0
1	1	0

Figura 1

Dove *Aeig* è la colonna degli autovalori di A, *Beig* è la colonna degli autovalori di B e *Ceig* l'errore assoluto derivante dal sistema perturbato. (consultare il file eigenvalue.xlsx per tutti i risultati ottenuti) L'errore relativo alla norma è l'unico che ci dà un'indicazione di come la perturbazione influisca sulla matrice:

```
norm_comparison =   5.0011e-74
```

Per lo stesso motivo anche i risultati ottenuti per le matrici A^tA e B^tB sono del tutto simili, per quanto riguarda gli autovalori:

Aeig	Beig	Ceig
0,000457	0,000457	0
0,004109	0,004109	0
0,011408	0,011408	0
0,022338	0,022338	0
0,036882	0,036882	0
0,055011	0,055011	0
...	...	...
3,992697	3,992697	0
3,998173	3,998173	0

Figura 2

Mentre per quanto riguarda il confronto tra le norme, si vede una leggera diminuzione del valore:

```
norm_comparison =2.5011e-74
```

Esercizio 2

Questo esercizio prendeva in esame una matrice simmetrica A_G, matrice di adiacenza rappresentante il grafo in figura, l'inizializzazione della tabella avviene tramite la chiamata `zeros(n)` che crea una matrice *11x11* nella quale viene poi posizionato il valore 1 nelle posizioni indicate dalle coppie di nodi adiacenti:

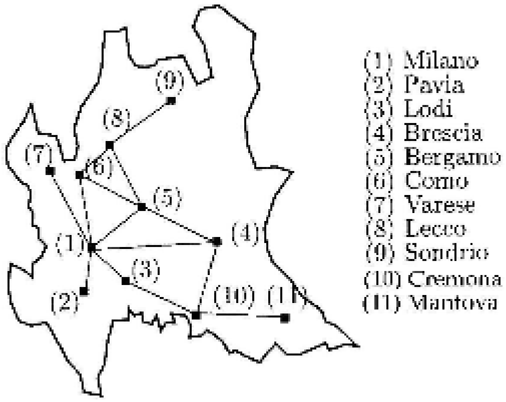


Figura 3

```
% this vector contains
pairs of nodes which are adjacent:
adj_nodes = [1 7; 1 6 ; 1 5 ; 1 2 ; 1 3 ; 1 4 ; 3 10 ; 10 11 ; 10 4 ; 4 5 ; 5 6;
5 8 ; 8 9];

% create adjency matrix:
for i=1:13
   Ag = place_simmetrically(Ag,adj_nodes(i,1), adj_nodes(i,2));
end
```

Dove:

```
function [A] = place_simmetrically(A,x,y)
    A(x,y) = 1;
    A(y,x)= 1;
end
```

place_symmetrically *posiziona in posizioni simmetriche il valore 1, che indica un arco di collegamento tra i nodi rappresentati dagli indici di posizione. In alcune matrici di adiacenza il valore posizionato è il peso dell'arco (ad esempio la lunghezza dell'arco), nel nostro caso il valore 1 è comodo poiché sommando il numero di archi uscenti dal nodo otteniamo un valore pari al grado di connessione come spiegato nel paragrafo successivo.*

Con questa ultima funzione termina il punto *a* del secondo esercizio. Il punto *b* prevedeva di calcolare la matrice diagonale D, che ha come elementi il numero di archi connessi a ciascun nodo, la matrice D viene quindi costruita con la chiamata `D = diag(sum(Ag))` dove **sum** restituisce il vettore colonna contenente la somma di ciascuna riga della matrice Ag (e quindi il numero di archi collegati a ciascun nodo), calcolata la matrice D, viene chiesto di calcolare la matrice AgD^{-1} ed ottenere i suoi autovalori. La matrice AgD^{-1} viene calcolata con `AgDi = Ag/D` per maggiore precisione e velocità da parte del programma.

Autovalore	Nodo
1	Milano
0,778393493	Pavia
0,64207143	Lodi
0,278540846	Brescia
-0,893763485	Bergamo
-0,817510716	Como
-0,621639834	Varese
-0,366091734	Lecco
6,40837E-17	Sondrio
-1,20529E-17	Cremona
-1,20529E-17	Mantova

Figura 4

Gli autovalori della matrice ottenuta rispecchiano l'*importanza* del nodo rispettivo, infatti il nodo 1(Milano) ha in modulo il l'autovalore più grande, a seguire i nodi 5(Bergamo) e 6(Como) da notare che mentre l'importanza del nodo 1(Milano) dipende **solo** dal suo grado di connessione, l'importanza degli altri nodi dipende sia dal loro grado di connessione, sia dalla distanza dal nodo 1, motivo per cui i nodi 2(Pavia) e 3(Lodi) hanno importanza minore rispetto ai nodi 5(Bergamo) e 6(Como).

Esercizio 3

Il terzo esercizio prevedeva, nel primo punto, di calcolare gli autovalori della matrice $A = \begin{pmatrix} 1 & -1 & 2 \\ -2 & 0 & 5 \\ 6 & -3 & 6 \end{pmatrix}$ tramite il metodo delle potenze, un metodo iterativo che determina l'autovalore di modulo massimo di una data matrice con l'ausilio di un vettore di coefficienti. Prima di testare il metodo ci siamo quindi assicurati che gli autovalori della matrice data non fossero uguali in modulo con il comando:

```
eig(A)

-1.0000

 5.0000

 3.0000
```

Questo indica che il metodo è applicabile. Il metodo iterativo come mostrato a pagina 24 delle note:

$y_0 = (1, \ldots, 1)^T$

$\forall m = 0 \ldots max$

$w_{m+1} = Ay_m$ autovettore

$\lambda_1^{(m+1)} = \frac{y_m^T w_{m+1}}{y_m^T y_m}$ autovalore associato all'autovettore

$y_{m+1} = \frac{w_{m+1}}{\|w_{m+1}\|}$ il nuovo termine della successione deve essere normalizzato dal momento che la potenza λ^m può tendere sia a 0 che a $+\infty$ a seconda che l'autovalore sia maggiore o minore di uno!

Viene implementato dalla funzione

```
function [conv_speed, lam] = mp(v, A)
    prevlam = 0;
    conv_speed = 0;
    while true
        % Computing egeinvector
        w=A*v;
        % Computing egeinvalue:
        lam = (v'*w)/(v'*v);
        % updating v:
        v = w/norm(w);
        if(abs(lam-prevlam)<eps)
            break;
        end
        prevlam = lam;
        conv_speed = conv_speed+1;
    end
end
```

***mp** implementa il metodo delle potenze per il calcolo dell'autovalore dominante di una matrice passata in argomento tramite l'ausilio di un vettore di partenza per il calcolo della successione. La convergenza del metodo è garantita solamente dalla matrice passata in argomento, che deve avere autovalori distinti in modulo (al contrario il ciclo non termina, scardinando una delle caratteristiche della successione) mentre il criterio di arresto a riga 38 garantisce la restrizione a un numero finito di passi per il calcolo dell'autovalore.*

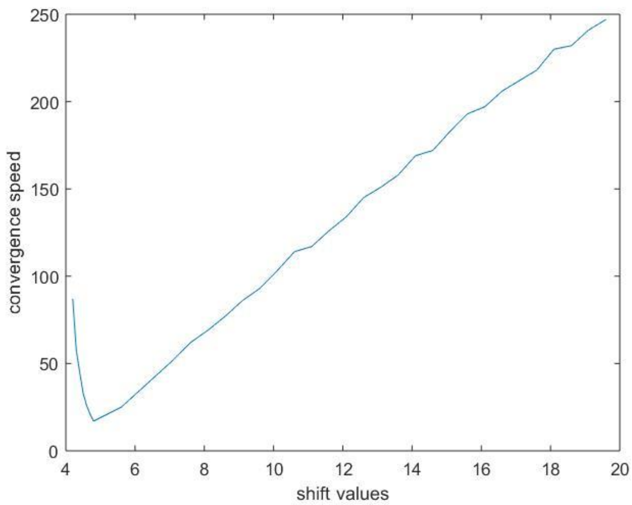
Il secondo punto dell'esercitazione richiedeva di calcolare sempre l'autovalore massimo, ma col metodo delle potenze inverse, metodo utilizzato per calcolare gli autovalori intermedi data una matrice di dimensioni arbitrarie. In questo caso la convergenza del metodo dipende esclusivamente dalla scelta dello shift che si utilizza per il calcolo, che deve essere un valore vicino a quello cercato che, nel nostro caso, può essere qualsiasi valore strettamente maggiore di **4**.

Definito il valore di shift, il metodo delle potenze, come mostrato a pagina 24 delle note, viene implementato dalla funzione:

```
function [ conv_speed, lam ] = inv_mp(v, A)
    shift=4.5;
    I=eye(size(A));
    prev=0;
    conv_speed=0;
    while true
     W=inv(A-shift*I)*v;
     lam_aux=(v' * W)/(v'*v);
     v=W/(norm(W));
     lam=shift+(1/lam_aux);
     % stop if more accurate than eps
     if (abs(lam - prev)<eps)
        break;
     end
     prev=lam;
     conv_speed=conv_speed+1;
    end
end
```

***mp** implementa il metodo delle potenze inverse per il calcolo di un autovalore diverso da quello dominante di una matrice passata in argomento tramite l'ausilio di un vettore di partenza per il calcolo della successione. La convergenza del metodo è garantita dal valore di shift che deve avere un valore molto vicino a quello dell'autovalore cercato, mentre il criterio di arresto a riga 58 garantisce la restrizione a un numero finito di passi per il calcolo dell'autovalore.*

Per entrambe i metodi, il calcolo dell'autovalore ha una velocità di convergenza maggiore con il secondo vettore $v_2 = (3, 10, 4)$ come si può vedere nella tabella in figura 6: circa il doppio in entrambe i casi, notare inoltre, come il metodo delle potenze inverse abbia una velocità di convergenza minore di circa la metà del metodo delle potenze, questo dipende dal valore di shift utilizzato, nel nostro caso **4,5**, la velocità inizia a diminuire per valori sempre più vicini all'autovalore dominante (5) per poi ricrescere, come si può vedere in **figura 5** . Nell'intorno del valore 5 dovrebbe verificarsi una discontinuità eliminabile, data dal fatto che, scegliendo tale valore come shift, `det(A-shift*I)=0` e quindi no ammetterebbe inversa, fondamentale per il metodo in uso.



metodo	componenti vettore	velocità
mp	1, 1, 1	72
mp	3, 10, 4	143
inv_mp	1, 1, 1	33
inv_mp	3, 10, 4	67

Figure 5, 6