Metodi per il calcolo di autovalori

e autovettori

e stabilità dei sistemi lineari

Tesina per il corso di calcolo
numerico 2
A.A. 2007/2008

Samuela Locci

Indice

Introduzione

Introduzione

La conoscenza degli autovalori e degli autovettori di una matrice quadrata è richiesta non solo nell'ambito di importanti teorie della matematica, ma anche in molte applicazioni, nelle quali si deve disporre di una loro buona approssimazione numerica.
Gli autovalori e gli autovettori vengono utilizzati nelle discipline più disparate che vanno dalla meccanica quantistica alla compressione di immagini, dallo studio dei fenomeni sismici allo studio della stabilità dei sistemi dinamici, dalla propagazione dei segnali a questioni di statistica, ecc…
Si pensi addirittura che Google, il notissimo motore di ricerca in Internet, usa il calcolo degli autovalori di matrici di grandissima dimensione per stabilire le correlazioni tra le pagine web ogni volta che viene effettuata una ricerca.
Data la grande importanza di conoscere gli autovalori di una matrice, nel corso dell' ultimo secolo sono stati ideati e perfezionati molti metodi per il loro calcolo, alcuni dei quali saranno trattati ed esaminati in questa tesina.
Nell' ultimo capitolo del testo è stata riportata un'applicazione degli algoritmi trattati per lo studio della stabilità dei sistemi lineari.

Capitolo 1

Approssimazione di autovalori e autovettori

1.1 Autovalori e autovettori

Si dicono autovalore ed autovettore di una matrice A uno scalare λ ed un vettore $x \neq 0$ che verifichino la relazione:

$$Ax = \lambda x$$

Tale equazione può essere riscritta nella forma

$$(A - \lambda I)x = 0$$

Affinché questo sistema lineare omogeneo ammetta una soluzione non nulla è necessario che il suo determinante

$$p_A(\lambda) = \det(A - \lambda I)$$

si annulli. Essendo $p_A(\lambda)$ un polinomio di grado n in λ, chiamato polinomio caratteristico di A, il Teorema fondamentale dell'algebra assicura l'esistenza di n autovalori, non necessariamente distinti, che possono essere determinati calcolando gli zeri di $p_A(\lambda)$.

Inoltre per ciascun autovalore λ_k una soluzione non nulla del sistema singolare:

$$(A - \lambda_k I)x = 0$$

fornisce il corrispondente autovettore, che, nel caso in cui la matrice $(A - \lambda_k I)$ abbia rango n-1 resta determinato a meno di una costante moltiplicativa.

Definiamo **spettro di una matrice** l'insieme dei suoi autovalori:

$$\sigma(A) = \{\lambda_1, \ldots\ldots\ldots\ldots \lambda_k\}$$

e raggio spettrale il massimo dei moduli degli autovalori di A

$$\rho(A) = \max|\lambda_k|$$

1.2 Localizzazione geometrica degli autovalori

Il calcolo degli autovalori ed autovettori di una matrice è un problema potenzialmente delicato, sia perché computazionalmente oneroso sia perché, in taluni casi, può rivelarsi estremamente instabile conducendo ad una forte propagazione degli errori.
Inoltre poiché gli autovalori non sono altro che gli zeri del polinomio caratteristico $p_A(\lambda)$, per $n \geq 5$ è necessario ricorrere a metodi iterativi per la loro valutazione. Una conoscenza preliminare della dislocazione dello spettro della matrice nel piano complesso può risultare vantaggiosa per innescare opportunamente tali metodi.
I ***teoremi dei cerchi di Geshgorin*** ci permettono di individuare un sottoinsieme del piano complesso che contenga tutti gli autovalori di una matrice.

Primo teorema di Gershgorin: *Assegnata una matrice quadrata A, definiamo cerchi riga gli insiemi*

$$R_i = \{ z \in C : |z - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}| \} \quad , \; i = 1, \ldots\ldots\ldots\ldots, n$$

Allora gli autovalori di A sono contenuti nei cerchi riga, ossia:

$$\sigma(A) \subseteq \bigcup_{i=1}^{n} R_i$$

Corollario1: *Assegnata una matrice quadrata A, definiamo cerchi colonna gli insiemi*

$$C_j = \left\{ z \in C : |z - a_{ij}| \leq \sum_{\substack{i=1 \\ i \neq j}}^{n} |a_{ij}| \right\} \qquad j = 1, \ldots\ldots\ldots\ldots, n$$

Allora

$$\sigma(A) \subseteq \bigcup_{j=1}^{n} C_j$$

Corollario2: *Se indichiamo con S_R e S_C l'unione dei cerchi riga e dei cerchi colonna cioè:*

$$S_R \subseteq \bigcup_{i=1}^{n} R_i \qquad S_C \subseteq \bigcup_{j=1}^{n} C_j$$

Allora:

$$\sigma(A) \subseteq S_R \cap S_C$$

Secondo teorema di Gershgorin: *supponiamo che l'insieme S_R sia costituito dall'unione di due sottoinsiemi disgiunti, cioè che:*

$$S_1 = \bigcup_{i=1}^{k} R_i \; , \; S_2 = \bigcup_{i=k+1}^{n} R_i \; , \; S_1 \cap S_2 = 0$$

Allora S_1 contiene esattamente k autovalori di A, contati con la loro molteplicità, e S_2 contiene i rimanenti n-k.

***Terzo teorema di Gershgorin**: Sia A una matrice irriducibile. Se un autovalore λ di A appartiene alla frontiera di S_R allora esso deve appartenere alla frontiera di ciascun cerchio riga R_i, i=1,....n.*

1.2.1 Applicazioni teoremi di Gershgorin

Vediamo ora alcune applicazioni dei teoremi di localizzazione esposti sopra.
Possiamo visualizzare graficamente i cerchi di Gershgorin di una matrice A attraverso la *function* gershgorin.m. Tale funzione visualizza in tre finestre grafiche distinte i cerchi di riga, i cerchi di colonna e l'intersezione dell'unione dei cerchi di riga e di colonna.
Consideriamo per cominciare la matrice:

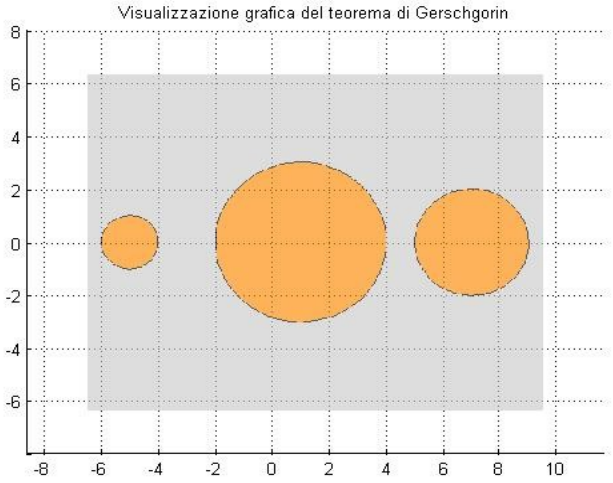
$$A=\begin{Vmatrix} 1 & 2 & -1 \\ 2 & 7 & 0 \\ -1 & 0 & -5 \end{Vmatrix}$$

Lo spettro di tale matrice è $\sigma(A)=\{$ -5.1712, 0.55888, 7.6123$\}$.
Utilizzando il comando gershgorin(A) otteniamo:

-cerchi riga: $R_1=\{|z||z-1|\leq 3\}$, $R_2=\{|z||z-7|\leq 2\}$, $R_3=\{|z||z+5|\leq 1\}$;

-cerchi colonna: $C_1=\{|z||z-1|\leq 3\}$, $C_2=\{|z||z-7|\leq 2\}$, $C_3=\{|z||z+5|\leq 1\}$;

La loro intersezione ed unione è riportata nella figura sottostante in color arancio:



Osservando tale figura notiamo che gli autovalori sono compresi negli intervalli: [-2, 4], [5, 9] e [-6, -4]. Possiamo inoltre vedere che, poiché la matrice A è simmetrica ed ha quindi autovalori reali, cerchi riga e cerchi colonna coincidono, quindi per localizzare gli autovalori potevamo limitarci a considerare esclusivamente i cerchi riga.

Consideriamo ora la seguente matrice complessa:

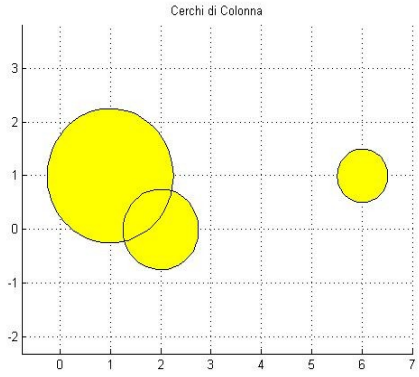
$$A=\begin{Vmatrix} 1+i & 0.5 & 0.5 \\ 1 & 2 & 0 \\ 0.25 & -0.25 & 6+i \end{Vmatrix}$$

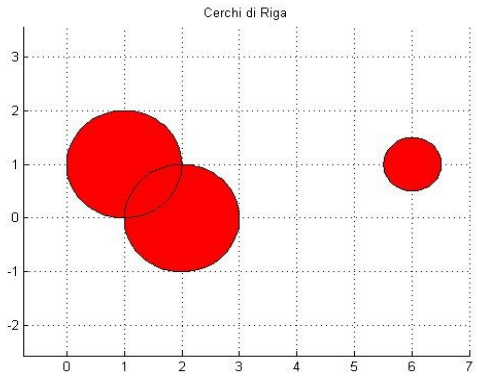
Lo spettro di tale matrice è $\sigma(A)=\{0.69+0.82i,\ 2.29+0.18i,\ 6.02+1.01i\}$.
Attraverso la *function* gershgorin.m otteniamo:

-cerchi riga: $R_1=\{|z\in\mathbb{C}:|z-(1+i)|\leq 1\}$, $R_2=\{|z\in\mathbb{C}:|z-2|\leq 1\}$, $R_3=\{|z\in\mathbb{C}:|z-(6+i)|\leq 0.5\}$;

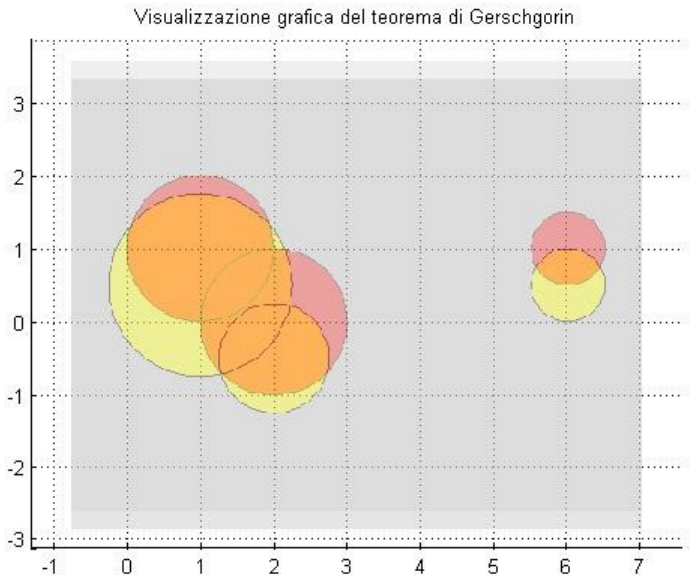
-cerchi colonna: $C_1=\{|z\in\mathbb{C}:|z-(1+i)|\leq 1.25\}$, $C_2=\{|z\in\mathbb{C}:|z-2|\leq 0.75\}$, $C_3=\{|z\in\mathbb{C}:|z-(6+i)|\leq 0.5\}$;

Riportati nelle immagini sottostanti:





Riportiamo ora, l'intersezione $S_R \cap S_C$ che è colorata in arancio:



Notiamo che gli autovalori sono tutti localizzati nella zona in arancio.

1.3 Metodo delle potenze

Il metodo delle potenze è un metodo iterativo particolarmente adatto per approssimare l'autovalore di modulo massimo di una matrice, ed il relativo autovettore associato.
Esso converge quando sono verificate le seguenti tre ipotesi:

- A è diagonalizzabile;
- il vettore iniziale $x^{(0)}$ ha una componente non nulla lungo l'autovettore v_1 corrispondente a λ_1;
- l'autovalore di modulo massimo è separato dagli altri

Consideriamo quindi una matrice A di ordine n diagonalizzabile e siano $\lambda_1,.....\lambda_n$ i suoi autovalori tali che:

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

supponiamo cioè che l'autovalore di modulo massimo λ_1 abbia molteplicità algebrica unitaria e che non esistano altri autovalori con lo stesso modulo.
L'algoritmo è allora il seguente: fissato un vettore iniziale $x^{(0)} \in C^n$, si genera la successione di vettori il cui termine k-esimo è:

$$x^{(k)} = A^k x^{(0)}$$

questa successione converge all'autovettore principale v_1 della matrice A.
Esaminiamo il comportamento per $k->\infty$ della successione $x^{(k)}$.

Nelle ipotesi fatte su A si ha che $x^{(0)}$ ha rappresentazione $x^{(0)} = \sum_{i=1}^{n} \alpha_i v_i$, e supponendo che $\alpha_1 \neq 0$ si ha: $x^{(k)} = \alpha_1 \lambda_1^k v_1 + \sum_{i=2}^{n} \alpha_i \lambda_1^k v_i = \alpha_1 \lambda_1^k \left(v_1 + \sum_{i=2}^{n} \frac{\alpha_i}{\alpha_1} \left(\frac{\lambda_i}{\lambda_1} \right)^k v_i \right)$

Da cui si vede che, poiché $\left| \lambda_i / \lambda_1 \right| < 1$ per $i \geq 2$, la direzione del vettore $x^{(k)}$ tende a quella di v_1.

Nella forma sopra esposta, l'algoritmo può presentare problemi di *underflow* o di *overflow* dovuti al fatto che λ_1^k può tendere a zero o all'infinito. E' necessaria, quindi, un'operazione di *normalizzazione* che può essere effettuata utilizzando a priori diversi tipi di norma di vettore.
L'iterazione assume la seguente forma:

$$\begin{cases} x^k = Aq^{(k-1)} \\ q^{(k)} = \dfrac{x^k}{\|x^k\|_2} \\ \lambda_1^{(k)} = \dfrac{(q^{(k)})^T A q^{(k)}}{(q^{(k)})^T q^{(k)}} \end{cases}$$

Sarà inoltre necessario introdurre un criterio di stop del tipo:
$\left| \lambda_1^{(k)} - \lambda_1^{(k-1)} \right| < \tau \left| \lambda_1^{(k)} \right|$ oppure $k > N$

Il metodo delle potenze può essere implementato in Matlab utilizzando la *function* potenze.m il cui listato, nella sua parte essenziale, è riportato di seguito.
I dati in ingresso **x0**, **tol**, **nmax** rappresentano rispettivamente il vettore iniziale, la tolleranza sotto la quale si accetta l'approssimazione calcolata e il numero massimo di iterazioni consentite.
La funzione restituisce lo scalare **lambda** che rappresenta l'approssimazione dell'autovalore di modulo massimo, il vettore **x1**, approssimazione del corrispondente autovettore e lo scalare **iter** corrispondente al numero di iterazioni impiegate.

```
function [lambda,x,iter]=potenze(A,tol,nmax,x0)
x0 = x0/norm(x0);
pro = A*x0;
lambda = x0'*pro;
err = tol*abs(lambda) + 1;
while  (err > tol*abs(lambda) &  iter <= nmax)
   x = pro;
   x = x/norm(x);
   pro = A*x;
   lambdanew = x'*pro;
   err = abs(lambdanew - lambda);
   lambda = lambdanew;
   iter = iter + 1;
end
```

1.3.1 Sperimentazione del metodo delle potenze

In questa sezione testeremo la validità dell'algoritmo e la sua convergenza. Iniziamo col generare la matrice A sulla quale saranno eseguiti i primi test, utilizzando i seguenti comandi Matlab:

D = diag ([λ_1,.....,λ_n]);
Q =orth(rand(n));
A =Q'*D*Q;

La matrice generata è simile alla matrice D scelta in partenza, pertanto gode di un'importantissima proprietà: ha i suoi stessi autovalori.
Scegliamo una matrice diagonale $5x5$ così costituita:

D=diag(λ_1, 3, 1, 7, 10);
Q=orth(rand(5));
A =Q'*D*Q;

Abbiamo indicato con λ_1 l'autovalore di modulo massimo.

Facciamo diverse prove utilizzando come vettore iniziale $x^{(0)}$=[1 1 1 1 1], come tolleranza tol=1e-14 e facendo variare il valore dell'autovalore di modulo massimo λ_1:

λ_1	N iterazioni	Errore	lambda
2000	4	4.77484718430787e-012	2000
1000	5	1.13686837721616e-013	1000
250	6	8.5265128291212e-014	250
50	12	4.2632564145606e-014	50
20	24	8.5265128291212e-014	20
11	159	9.76996261670138e-014	11
-10	55	1.15463194561016e-014	1.25021577449067

La tabella in alto mette in evidenza come il metodo delle potenze sia tanto più rapidamente convergente quanto più l'autovalore dominante sia ben separato dagli altri, ovvero tale che λ_2/λ_1 sia decisamente inferiore a uno.
Notiamo inoltre che se nello spettro della matrice A, ci sono due autovalori dominanti opposti $\lambda_2 = -\lambda_1$ allora il metodo delle potenze non converge poiché durante la costruzione dell'algoritmo avevamo supposto che esistesse uno ed un solo autovalore dominante. L'algoritmo ci fornisce erroneamente come autovalore dominante lambda=1.2502 anziché restituire il valore corretto lamda=10.
Tuttavia nel caso si abbia $\lambda_2 = -\lambda_1$ si può approssimare l'autovalore di modulo massimo applicando il metodo delle potenze alla matrice A^2.
Si ha infatti $\lambda_i(A^2)=[\lambda_i(A^2)]^2$, per $i=1,.....,n$ in modo $\lambda_1^2=\lambda_2^2$ e si ricade nel caso di autovalori coincidenti per cui il metodo converge.

Otteniamo infatti:

λ_1	N iterazioni	Errore	lambda
-10	24	2.70006239588838e-013	99.9999999999998

E' necessario mettere in evidenza che per l'applicazione di questo algoritmo la scelta del vettore iniziale $x^{(0)}$ è libera, tuttavia è buona norma prendere un vettore generato in maniera casuale $x^{(0)}$ = **rand(n,1)** in modo da ridurre la probabilità di violare la seconda ipotesi di convergenza dell'algoritmo.
A questo proposito consideriamo una matrice A generata dal comando Matlab:

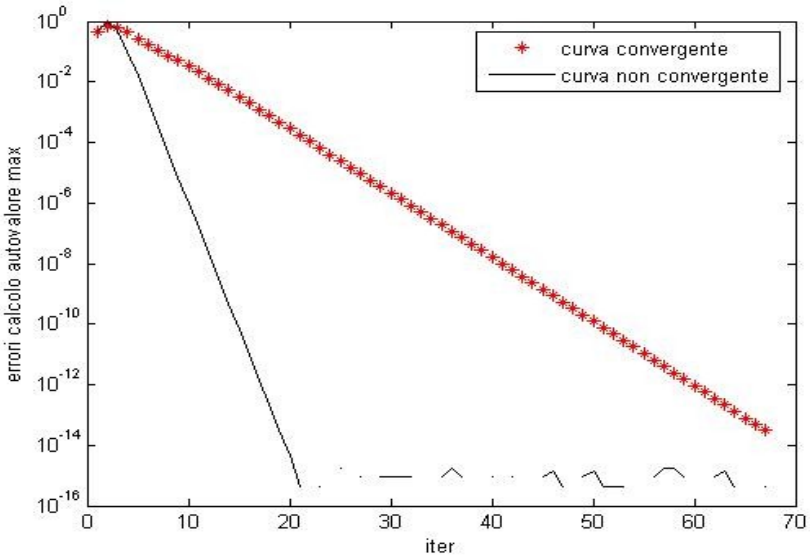
A= toeplitz([-3,1,0,0]);

Gli autovalori di questa matrice, generati col comando **eig** di Matlab, sono:

$\lambda_1 = -4.6180$; $\lambda_2 = -3.6180$; $\lambda_2 = -2.3819$; $\lambda_2 = -1.3819$;

Applicando il metodo delle potenze con vettore iniziale $x^{(0)}$ =[1 1 1 1] otteniamo invece:
- λ_1= -3.6180
- N° iterazioni = 19

quindi il metodo converge erroneamente al secondo autovalore più grande in modulo.
Questo accade perché l'autovettore principale è ortogonale a [1;1;1;1], e quindi il vettore iniziale ha componente nulla rispetto al primo autovettore.
Ponendo ora $x^{(0)}$ = rand(n,1) e riapplicando il metodo si ottiene la convergenza all'autovalore corretto: $\lambda_1 = -4.6180$ con un numero di iterazioni pari a 67.
Mostriamo nella figura sottostante l'andamento dell'errore nei due casi sopra esposti:



Testiamo infine l'algoritmo su una matrice A in forma compagna.
Per ottenerla, Matlab ci fornisce due funzioni:

- **poly** che riceve un vettore contenente gli autovalori e restituisce un vettore contenente i coefficienti del polinomio che ha come radici i valori contenuti nel vettore di input;
- **compan** genera la matrice compagna utilizzando i valori del vettore in ingresso per calcolare il valore degli elementi della prima riga.

Si ha:
p=poly([λ_1,.....λ_n])
A=compan(p)

Nella tabella sottostante sono riportati i risultati ottenuti:

λ_1,.....λ_n	N iterazioni	Errore	lambda
[1:5]	134	4.9737991503207e-014	5.00000000000021
[1:8]	221	7.28306304154103e-014	8.00000000000093
[1:10]	261	9.9475983006414e-014	9.99999999999735
[1:12]	411	5.86197757002083e-014	12.0000000000844
[1:14]	1654	3.73034936274053e-014	13.9999999980131
[1:20]	2001	6.12470358873907e-008	19.9999971368358

Analizzando la tabella notiamo che all'aumentare delle dimensioni della matrice, il metodo delle potenze converge all'autovalore di modulo massimo ma con moltissime iterazioni.
Per scoprire perché accade questo, bisogna fare un'analisi più accurata della stabilità del calcolo.
A questo proposito enunciamo il seguente teorema:

***Teorema di Bauer-Fike**: Sia $A \in C^{nxn}$ diagonalizzabile, cioè esista una matrice non singolare X tale che:*
$X^{-1}AX = D = diag(\lambda_1, ...\lambda_n)$
Fissata una matrice di perturbazione E, sia μ un autovalore di A+E. Allora
$$\min_{\lambda \in \sigma(A)} |\lambda - \mu| \leq k_2(X)\|E\|_2$$
dove $\sigma(A)$ rappresenta lo spettro della matrice A e k_2 il numero di condizionamento in norma a due.

Il Teorema di Bauer-Fike afferma che, se la matrice A è diagonalizzabile, $k_2(X)$ *rappresenta il numero di condizionamento del problema agli autovalori.*

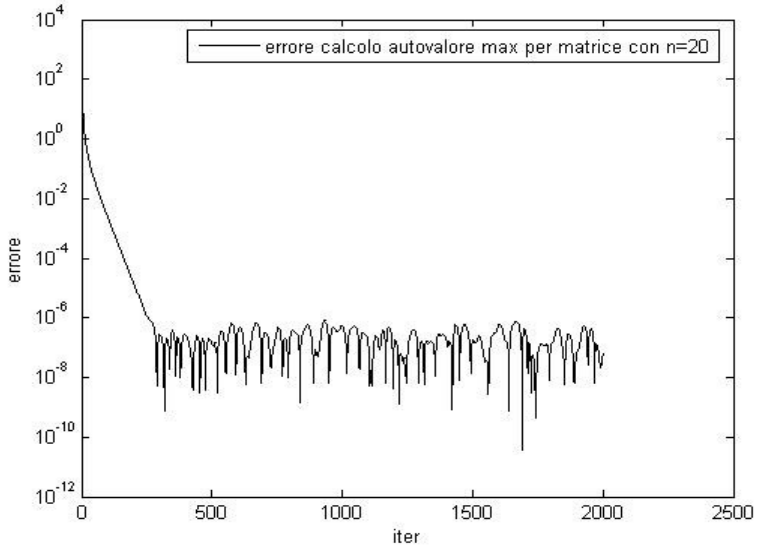
Vogliamo ora determinare il *numero di condizionamento* del nostro problema.
La prima cosa da fare è verificare se la nostra matrice A=compan(p) è diagonalizzabile.
E' noto che: se una matrice di ordine *n* ha n autovalori distinti, allora esiste un insieme di n autovettori linearmente indipendenti che costituiscono dunque una base per R^n.
Inoltre data una matrice A di dimensione nxn con autovalori λ_1,.....,λ_n , sia X=[$v_1|v_2|.....|v_n|$] una sua matrice modale (costituita dai suoi n autovettori).
La matrice Λ ottenuta:

$\Lambda = X^{-1}AX$ è diagonale.

Per le proposizioni riportate sopra possiamo dedurre che la nostra matrice A=compan(p), generata dal polinomio p=poly([λ_1,.....λ_n]), avendo autovalori tutti distinti è diagonalizzabile, pertanto risulta valido il teorema di *Bauer-Fike*.
Possiamo quindi concludere che $k_2(X)$ è numero di condizionamento del nostro problema; il suo valore è riportato nella tabella in basso:

λ_1,.....λ_n	$k_2(X)$
[1:5]	9485.49036761777
[1:8]	121498473.797356
[1:10]	130243291293.691
[1:12]	208338818562397
[1:14]	1.51540128507309e+018
[1:20]	8.24300784889221e+020

Dall'analisi della tabella notiamo che il valore di $k_2(X)$ è elevatissimo anche per matrici di piccola dimensione, oltre a presentare una significativa crescita all'aumentare delle dimensioni della matrice.
Da questo possiamo concludere che per una matrice in forma compagna il problema del calcolo degli autovalori risulta fortemente *mal condizionato*.
E' interessante osservare l'errore commesso nel calcolo dell'autovalore di modulo massimo, per una matrice di ordine elevato:



1.4 Metodo delle potenze inverse

Assegnato un numero $\mu \in C$, con $\mu \notin C^{nxn}$, vogliamo approssimare l'autovalore della matrice $A \in C^{nxn}$ più vicino a μ. A tal fine si può utilizzare il metodo delle potenze applicato alla matrice $(M_\mu)^{-1} = (A - \mu I)^{-1}$, ottenendo quello che viene chiamato metodo delle potenze inverse.
Il numero μ viene detto *shift*.
La matrice M_μ^{-1} ha per autovalori $\xi_i = (\lambda_i - \mu)^{-1}$; supponiamo che esista un intero m tale che:

$$|\lambda_m - \mu| < |\lambda_i - \mu| \quad \text{con} \quad i = 1,.....n \quad \text{e} \quad i \neq m$$

Questa ipotesi equivale ad assumere che l'autovalore λ_m più vicino a μ abbia molteplicità pari a 1.
L'autovalore di modulo massimo di M_μ^{-1} corrisponde pertanto a ξ_m, ed in particolare se $\mu = 0$, λ_m rappresenta l'autovalore di modulo minimo di A.
L'algoritmo è così strutturato:

$$\begin{cases} (A - \mu I)x^k = q^{(k-1)} \\ q^{(k)} = \dfrac{x^k}{\|x^k\|_2} \\ \sigma^{(k)} = \dfrac{(q^{(k)})^T A q^{(k)}}{(q^{(k)})^T q^{(k)}} \end{cases}$$

Gli autovalori di M_μ coincidono con quelli di A in quanto $M_\mu = X(\Lambda - \mu I_n)X^{-1}$ dove $\Lambda = diag(\lambda_1,.....\lambda_n)$.
La principale differenza rispetto al processo iterativo del metodo delle potenze tradizionale, risiede nel fatto che ad ogni passo k è necessario risolvere un sistema lineare avente matrice dei coefficienti: $M_\mu = A - \mu I$. Conviene dunque calcolare, per k=1, la fattorizzazione LU di M_μ, in modo da dover risolvere ad ogni passo solo due sistemi triangolari per un costo dell'ordine di n^2 *flops*.
Il metodo delle potenze può essere implementato in Matlab attraverso la *function* potinv.m il cui listato, nella sua parte essenziale, è riportato di seguito.
Il dato in ingresso **mu** contiene l'approssimazione iniziale per l'autovalore cercato.
Lo *script* restituisce **sigma** che rappresenta l'approssimazione dell'autovalore più prossimo a mu.
La fattorizzazione LU, con pivoting parziale, della matrice M_μ è stata effettuata usando la funzione Matlab intrinseca lu.

```
function [sigma, x, iter] = potinv (A, mu, tol, nmax, x0)
[L,U]=lu(A-mu*eye(n));
x0=x0/norm(x0);
err=tol*abs(sigma)+1;
while err > tol*abs(sigma) & iter <= nmax
   x = x/norm(x);
   z=L\x;
   pro=U\z;
   sigmanew = x'*pro;
   err = abs(sigmanew - sigma);
   sigma= sigmanew;
   iter = iter + 1; end
```

1.4.1 Sperimentazione del metodo delle potenze inverse

Per analizzare la convergenza del metodo delle potenze inverse facciamo i nostri test su una matrice del tipo:

D = diag ([λ_1,.....,λ_n]);
Q =orth(rand(n));
A =Q'*D*Q;

Consideriamo la seguente matrice diagonale $5x5$:

D=diag(2000, 35, 560, 78, 1200);

e generiamo la matrice A con i comandi riportati sopra.

Facciamo diverse prove utilizzando:
-vettore iniziale: $x^{(0)} = rand(n,1)$
-tolleranza: tol=1e-14
-μ variabile per ogni test

μ	N iterazioni	Errore	sigma
0	22	6.93889390390723e-017	35
10	17	2.63677968348475e-016	35
20	14	5.55111512312578e-017	35
30	9	3.60822483003176e-016	35
34.9	4	1.06581410364015e-014	35
77	5	1.22124532708767e-015	78
1999	4	4.44089209850063e-016	2000

Dalla tabella si può notare che tale metodo ha la proprietà fondamentale di convergere ad un generico autovalore di A, precisamente a quello più vicino allo shift μ.
La convergenza sarà tanto più rapida quanto più μ è vicino a λ_m.
Esso si può utilizzare convenientemente per migliorare una stima iniziale di μ di un autovalore di A, ottenuta ad esempio utilizzando l'algoritmo di localizzazione introdotto nella prima parte della tesina.
Ovviamente tale metodo non è applicabile su matrici singolari, in quanto queste non possono essere invertite.

1.5 L'algoritmo QR

L'algoritmo QR risulta essere il metodo più generale esistente per il calcolo di tutti gli autovalori di una matrice. Lo schema dell'algoritmo è particolarmente semplice:
Si parte ponendo $A_0 = A$. Nella generica iterazione si effettua la fattorizzazione QR della matrice A_k (ad esempio con l'algoritmo di Householder) e si calcola la nuova iterata rimoltiplicando i fattori in ordine inverso:

1. $A_0 = A$
2. for k=0,1,…… fino alla convergenza
    1. fattorizza $A_k = Q_k R_k$
    2. calcola $A_{k+1} = R_k Q_k$

Osserviamo che tutte le matrici della successione sono unitariamente simili:

$$A_{k+1} = R_k Q_k = Q_k^T Q_k R_k Q_k = Q_k^T A_k Q_k$$

e come tali hanno gli stessi autovalori.
Sotto opportune ipotesi la successione A_k converge ad una matrice triangolare superiore che ha come elementi diagonali gli autovalori di A.
L'ipotesi per la convergenza del metodo è che gli autovalori siano reali e distinti in modulo:

$$|\lambda_1| > |\lambda_2| > \ldots\ldots\ldots > |\lambda_n|$$

Se tali ipotesi non sono verificate, la successione A_k non converge ad una forma triangolare superiore; tuttavia è stato dimostrato che anche in questi casi il metodo può essere applicato con opportune varianti.
Sia ad esempio:

$$|\lambda_1| > \ldots\ldots > |\lambda_r| = |\lambda_{r+1}| > \ldots > |\lambda_n| > 0$$

dove λ_r e λ_{r+1} sono due numeri complessi coniugati, oppure due numeri reali. Allora la successione $\{A_k\}$ o meglio degli elementi diagonali non converge agli autovalori dello stesso modulo, ma gli autovalori dei blocchi diagonali convergono a λ_r e λ_{r+1}.
Situazioni analoghe si presentano quando la matrice A ha più autovalori di modulo uguale e in questo caso il metodo QR genera matrici R_k con struttura diagonale a blocchi, in cui gli autovalori dei blocchi diagonali convergono ad autovalori di A.

L'algoritmo QR può essere implementato utilizzando la *function* qrbase.m. Tale funzione prende in ingresso la matrice **A** ed il numero di iterazioni desiderate **iter**, mentre i parametri in uscita **T, Q** e **R** sono le matrici A_{k+1}, Q ed R dopo iter iterazioni del metodo.

1.6.1 Applicazione dell'algoritmo

Per testare questo algoritmo prendiamo per cominciare la matrice:

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

Applicando l'algoritmo otteniamo dopo 2001 iterazioni la seguente matrice triangolare a blocchi:

$$A_{2001}=\begin{Vmatrix}2 & 0.26726 & -1.3887\\ 0 & -0.5 & 0.86603\\ 0 & -0.86603 & -0.5\end{Vmatrix}$$

da cui si ricava immediatamente l'autovalore $\lambda = 2$, mentre gli altri due autovalori si ricavano come autovalori della seguente sottomatrice 2x2:

$$B=\begin{Vmatrix}-0.5 & 0.86603\\ -0.86603 & -0.5\end{Vmatrix}$$

cioè: $\lambda_{2,1}=-0.5\pm 0.866i$

Si può notare che la matrice assegnata, presentando due autovalori complessi e coniugati, non verifica le ipotesi di convergenza del metodo QR; tuttavia osserviamo che la successione A_{2001} converge almeno verso una forma triangolare a blocchi e questo ci consente comunque di ricavare tutto lo spettro di A.

Consideriamo ora una matrice con la struttura seguente:

D = diag ([λ_1,.....,λ_n]);
Q =orth(rand(n));
A =Q'*D*Q;

Modifichiamo gli elementi sulla diagonale di D riducendo il passo tra i vari termini ad ogni prova. Riportiamo i risultati ottenuti nella tabella seguente:

Diag[λ_1,.....λ_n]	N iterazioni	Errore
[20 16 12 8 4]	162	2.09133463581949e-015
[20 18 16 14 12]	338	1.50721186373355e-015
[20 19 18 17 16]	689	2.1359873651684e-015
[20 19.5 19 18.5 18]	1335	3.21955463672211e-015

Possiamo notare che più lo spettro di A risulta distribuito, ovvero più gli autovalori sono spaziati tra loro, più veloce risulta la convergenza del metodo.

Se invece gli autovalori risultano molto vicini tra loro, algoritmo QR risulta essere troppo lento, osserviamo infatti che nell'ultimo caso riportato nella tabella occorrono ben 1335 iterazioni per la convergenza del metodo.
Nella colonna a destra ho riportato l'errore commesso nell'approssimazione dello spettro di A. Tale errore è stato calcolato in questo modo:

$\lambda = [\ \lambda_1, \ldots\ldots \lambda_n\ ];$
$\tilde{\lambda} = [\ \tilde{\lambda}_1, \ldots\ldots \tilde{\lambda}_n\ ];$
$err = \dfrac{abs(\lambda - \tilde{\lambda})}{abs(\lambda)};$

dove λ rappresenta il vettore degli autovalori di A calcolati utilizzando la funzione intrinseca di Matlab eig, mentre $\tilde{\lambda}$ rappresenta l'approssimazione degli autovalori calcolata con la *function* qrbase.m.
Segnaliamo che la funzione eig implementa il metodo QR con opportune tecniche di *shift* in modo da accelerare la convergenza rispetto al metodo di base qualora A presenti autovalori molto vicini in modulo.

Consideriamo ora una matrice A in forma compagna che sappiamo essere generata come:
p=poly([$\lambda_1, \ldots\ldots\ldots\ldots \lambda_n$])
A=compan(p)

Ricordiamo che per una matrice di questo tipo il problema del calcolo degli autovalori risulta fortemente *mal condizionato*. Per questo motivo, è interessante confrontare i risultati che si ottengono implementando il metodo QR sia con la funzione intrinseca eig di Matlab e sia con la *function* qrbase.m.
Iniziamo i nostri test con una matrice di dimensione $5x5$:

$\lambda_1, \ldots\ldots \lambda_n$	N iterazioni	Autovalori: qrbase.m	Autovalori: eig	Autovalori attesi
[1:5]	144	5.0000000000001 3.99999999999985 3.00000000000005 2 0.999999999999999	4.99999999999983 4.00000000000034 2.99999999999983 2 1.00000000000001	5 4 3 2 1

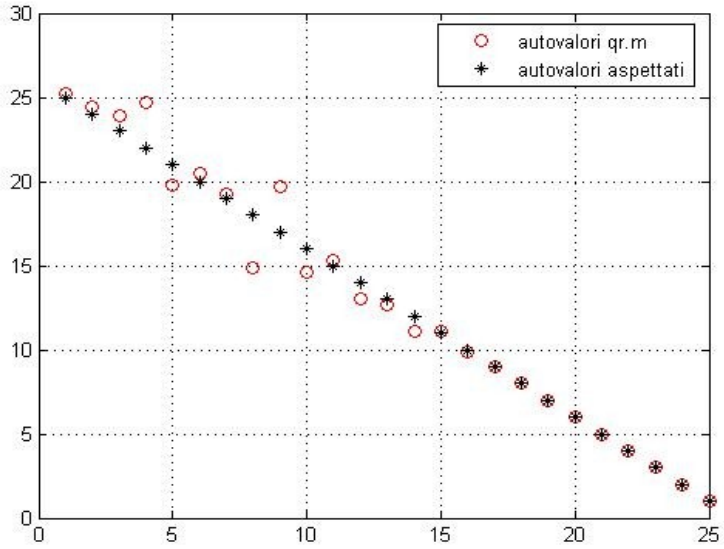
Nella tabella ho riportato le approssimazioni degli autovalori fornite rispettivamente dalla *function* qrbase.m e dalla funzione eig di Matlab, nella colonna a destra ho invece riportato i valori esatti degli autovalori della matrice A che ci attendevamo (essendo contenuti in poly([1:5])).
Analizzando i risultati si vede chiaramente che nel calcolo degli autovalori viene commesso un errore, piccolo ma esistente, sia con la funzione eig di Matlab sia con la *function* qrbase.m.
Questo errore è dovuto essenzialmente al *cattivo condizionamento* del problema.
Per completezza riportiamo di seguito i valori dell'errore commesso in entrambi i casi:

Erreig= 4.9620913438793e-14

Errqrbase.m= 2.27959065765307e-14

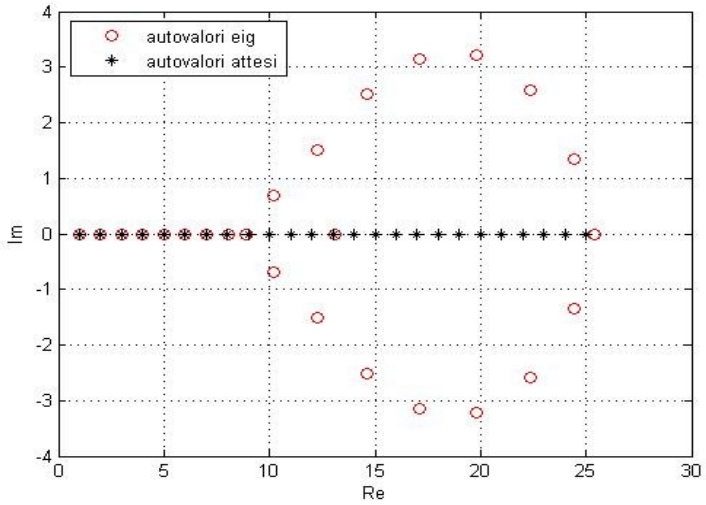
Consideriamo ora una matrice in forma compagna di dimensione $n = 25$ così costituita:
p=poly([1:25])
A=compan(p)

Confrontiamo, anche in questo caso, i risultati ottenuti implementando il metodo QR sia con la funzione intrinseca eig di Matlab e sia con la *function* qrbase.m.
Non costruiremo una tabella perché, a causa delle dimensioni della matrice, i valori da riportare sarebbero troppi; ma riporteremo i risultati ottenuti, attraverso due grafici.
Nel primo di questi abbiamo rappresentato in rosso gli autovalori calcolati applicando la *function* qrbase.m, invece in nero gli autovalori veri della matrice A che avremmo dovuto ottenere.



Osservando la figura si nota che il metodo QR da noi implementato ci fornisce una buona stima dello spettro di A fino all'autovalore $\lambda_{15} = 11$, dopodichè si discosta dai valori attesi.
E' necessario precisare che nel grafico qui sopra ho plotato gli elementi della diagonale della matrice A anche se, non sarei autorizzata a farlo visto che la sottodiagonale non è diventata piccola ma presenta alcune componenti decisamente non nulle.
Se applicassi l'algoritmo qr in modo rigoroso, ovvero senza trascurare il fatto che la sottodiagonale ottenuta contiene elementi non nulli, otterrei in realtà gli stessi autovalori che mi fornisce la funzione eig di Matlab.

Nel secondo grafico mostriamo, invece, i risultati ottenuti calcolando gli autovalori della nostra matrice A con la *function intrinseca* eig.



Osservando la figura, ci rendiamo subito conto che la stima degli autovalori ottenuta è molto lontana dall'effettivo spettro della matrice A. La *function* eig di Matlab ci fornisce addirittura sette autovalori complessi e coniugati, da questo né concludiamo che non è affidabile per calcolare "*stime esatte"* di autovalori per matrici in forma compagna di grandi dimensioni.

Capitolo 2

Stabilità dei sistemi lineari

2.1 Premessa

La stabilità è senza dubbio la proprietà più studiata dei sistemi dinamici. L'importanza di tale proprietà deriva dal fatto che la stabilità è una specifica imposta a quasi ogni sistema fisico controllato, perché implica la possibilità di lavorare intorno a certe condizioni nominali senza discostarsi troppo da esse.
In questo capitolo ci limiteremo ad analizzare la stabilità dei sistemi lineari.
E' fondamentale precisare che il caso dei sistemi lineari è particolare, per questo tipo di sistemi è possibile infatti parlare di stabilità come *caratteristica* del sistema. Questo non è invece possibile nel caso di sistemi non-lineari per cui la stabilità è una proprietà che dipende non solo dal sistema ma anche dalle sue condizioni.
Esistono diverse definizioni di stabilità, ma senza dubbio la più significativa è la stabilità alla Lyapunov. Questa definizione, che verrà enunciata nel seguito, è valida in generale sia per sistemi lineari che non lineari ma come detto prima la nostra trattazione si limiterà per semplicità solo ai sistemi lineari.

2.2 Stabilità alla Lyapunov

La teoria della stabilità alla Lyapunov è basata sulla nozione fondamentale di *stato di equilibrio.*

Dato un sistema lineare autonomo $\dot{x}(t) = Ax(t)$*, lo stato* x_e *è un punto di equilibrio se e solo se è soluzione del sistema lineare omogeneo:*

$Ax_e = 0$

Da ciò derivano immediatamente i seguenti risultati:

- *Se la matrice A è non singolare, l'unico stato di equilibrio del sistema è* $x_e = 0$ *ossia l'origine*
- *Viceversa, se A è singolare allora il sistema ha un numero infinito di stati di equilibrio che descrivono uno spazio lineare*

2.2.1 Stabilità dei punti di equilibrio

Dato un sistema lineare autonomo:

- *Se uno stato di equilibrio è stabile (instabile), ciò implica che anche tutti gli altri eventuali stati di equilibrio sono stabili (instabili)*
- *Se uno stato di equilibrio* x_e *è asintoticamente stabile, allora valgono i tre seguenti risultati:*

  - $x_e = 0$*, ovvero tale stato coincide con l'origine;*

  - x_e *è l'unico stato di equilibrio del sistema ;*

  - x_e *è globalmente asintoticamente stabile, ossia il suo dominio di attrazione coincide con l'intero spazio di stato.*

La precedente proposizione spiega perché nel caso di sistemi lineari è lecito parlare di sistema stabile, ovvero instabile, ovvero sistema asintoticamente stabile, anziché riferire tali proprietà al generico stato di equilibrio.
Nell'analisi dei sistemi lineari e stazionari esistono numerosi risultati in termini di stabilità.
Il criterio di analisi di stabilità utilizzato di solito è quello basato sul calcolo degli autovalori della matrice A.

Teorema: Criterio degli autovalori
Si consideri il sistema lineare e stazionario:

$$\dot{x}(t) = Ax(t)$$

- *Tale sistema è asintoticamente stabile se e solo se tutti gli autovalori della matrice A hanno parte reale negativa.*

- *Tale sistema è stabile se e solo se la matrice A non ha autovalori a parte reale positiva e gli eventuali autovalori a parte reale nulla hanno indice unitario*

- *Tale sistema è instabile se e solo se almeno un autovalore di A ha parte reale positiva, oppure parte reale nulla e indice maggiore di uno.*

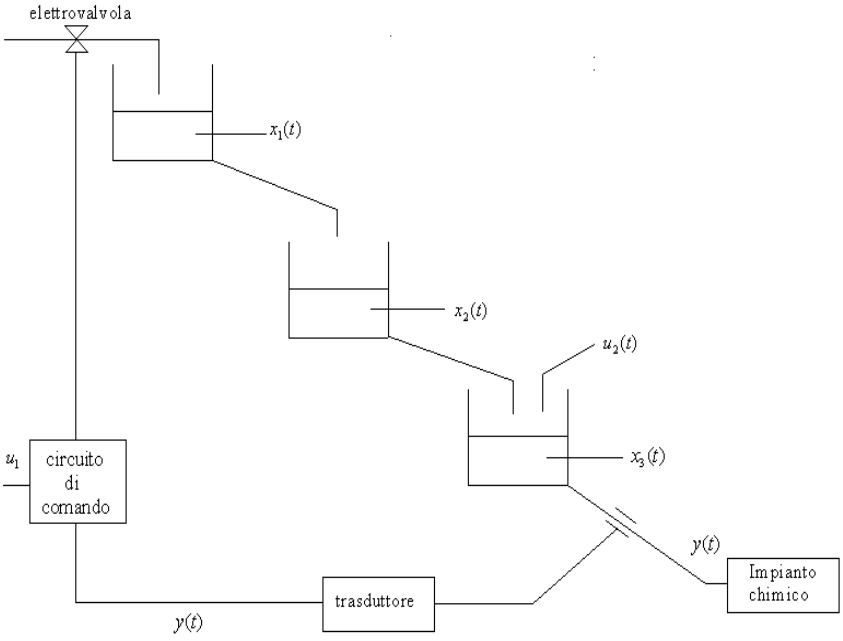
Il criterio agli autovalori fornisce una "*misura*" della stabilità del sistema.
E' infatti scontato che piccole variazioni di un parametro caratterizzante un sistema si ripercuotano in piccole variazioni degli elementi a_{ij} della matrice A e, quindi, in piccole variazioni degli autovalori λ_n del sistema.
Un sistema a tempo continuo asintoticamente stabile resterà pertanto tale anche se soggetto a piccole perturbazioni parametriche, se l'indicatore:

$$\min\{-\mathrm{Re}(\lambda_n)\}$$

è sufficientemente grande.

2.3 Applicazione pratica

Facciamo ora un esempio di applicazione del criterio degli autovalori per studiare la stabilità di un sistema reale.
Supponiamo di dover regolare l'alimentazione di un impianto chimico.
Lo schema al quale facciamo riferimento è il seguente:



Tale impianto ha portata di alimentazione $y(t)$ che deve essere mantenuta il più possibile costante e pari ad un valore desiderato u_1. La portata $y(t)$ è l'uscita di un serbatoio preceduto da altri due serbatoi uguali e alimentati da una portata $v(t)$ che può essere variata attraverso un'elettrovalvola.
All'equilibrio la portata di uscita $\hat{y}$ è uguale alla portata di alimentazione, per cui se $u_2(t)$ fosse identicamente nullo basterebbe fissare l'alimentazione v al valore costante u_1 per ottenere un

funzionamento corretto dell'intero impianto. Per compensare alle variazioni della portata $y(t)$, provocate dal disturbo $u_2(t)$ si pone:

$$v(t) = u_1 - k[y(t) - u_1] \qquad \text{con k>0}$$

Per compensare meglio i disturbi si tende a scegliere un valore di guadagno k molto elevato, ma questo è possibile solo se il sistema rimane asintoticamente stabile per alti valori di k.
Per vedere se ciò accade possiamo applicare il criterio degli autovalori.
Scriviamo le equazioni che rappresentano la dinamica del sistema:

$$\dot{x}_1(t) = -ax_1(t) + u_1 - k(ax_3(t) - u_1)$$
$$\dot{x}_2(t) = ax_1(t) - ax_2(t)$$
$$\dot{x}_3(t) = ax_2(t) + u_2(t) - ax_3(t)$$
$$y(t) = ax_3(t)$$

dove a è un coefficiente e $x_i(t)$ sono i tre volumi
La matrice di stato A risulta quindi:

$$A = \begin{Vmatrix} -a & 0 & -ka \\ a & -a & 0 \\ 0 & a & -a \end{Vmatrix}$$

Poniamo $a = 2$ e facciamo invece variare k.

CASO 1: $a = 2$ e k=3

La matrice A diventa quindi:

$$A = \begin{Vmatrix} -2 & 0 & -6 \\ 2 & -2 & 0 \\ 0 & 2 & -2 \end{Vmatrix}$$

Tale matrice è non singolare per cui possiamo subito affermare che l'origine è l'unico stato di equilibrio del sistema, dobbiamo ora vedere se questo è stabile o meno.
Per verificare la validità dei risultati che otterremo applicando gli algoritmi, andiamo a calcolare innanzitutto gli autovalori della matrice A con la *function* eig di Matlab.
Gli autovalori trovati sono:

$\lambda_1 =$ -4.88449914061482
$\lambda_{2,1} =$ -0.55775042969259 ± 2.49804953296681i

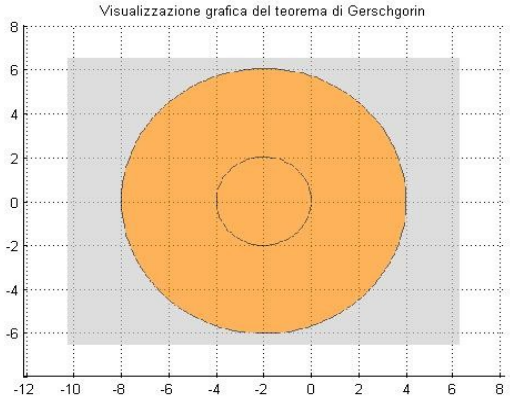
Supponiamo ora di non aver fatto l'operazione precedente e quindi di non conoscere nulla sullo spettro della matrice A.
In queste situazioni la prima cosa da fare è applicare l'algoritmo di localizzazione degli autovalori attraverso la *function* gershgorin.m.

Otteniamo i seguenti risultati:

-cerchi riga: $R_1 = \{|z| : |z+2| \leq 6\}$, $R_2 = \{|z| : |z+2| \leq 2\}$, $R_3 = \{|z| : |z+2| \leq 2\}$;

-cerchi colonna: $C_1 = \{|z| : |z+2| \leq 2\}$, $C_2 = \{|z| : |z+2| \leq 2\}$, $C_3 = \{|z| : |z+2| \leq 6\}$;

La loro intersezione ed unione è riportata nella figura sottostante in color arancio:

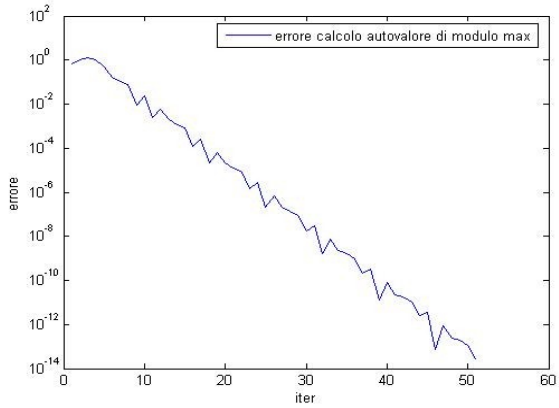


Notiamo che cerchi riga e cerchi colonna coincidono, pertanto gli autovalori sono localizzati nell'intervallo [-8, 4].

Una volta localizzata la zona del piano complesso che contiene gli autovalori, possiamo procedere al loro effettivo calcolo.

Applichiamo per cominciare il **metodo delle potenze** che, ricordiamo fornisce un'approssimazione dell'autovalore di modulo massimo.

Attraverso la *function* potenze.m troviamo che: dopo 51 iterazioni il metodo converge all'autovalore di modulo massimo $\lambda_1 =$ -4.884499 con un errore il cui andamento è riportato in basso:



Applichiamo ora l'algoritmo delle **potenze inverse** con $\mu = 0$ per trovare una stima dell'autovalore di modulo minimo.

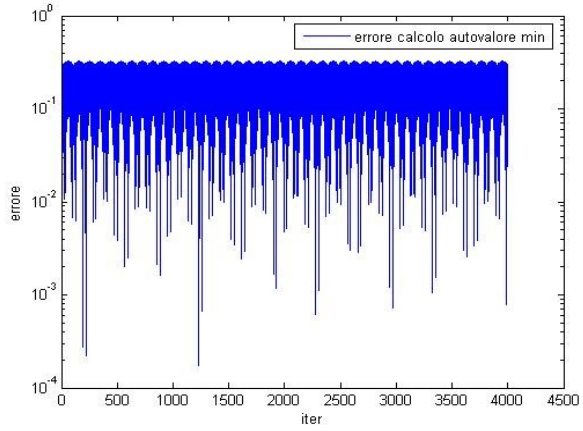
Otteniamo i seguenti risultati:

Numero iterazioni: 4001

Autovalore di modulo minimo trovato: -7.55679

Errore commesso: 0.0468

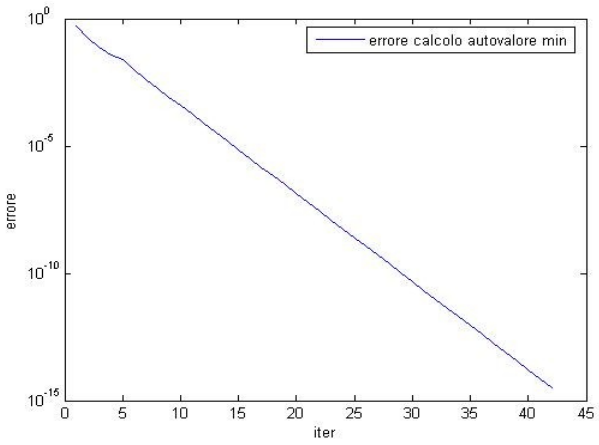
Riportiamo in basso l'andamento dell'errore:



Come possiamo notare l'algoritmo non converge al corretto autovalore di modulo minimo, ma ci fornisce erroneamente λ_m= -7.55679.

Riapplichiamo il metodo ponendo μ =i e vediamo cosa succede:
Numero iterazioni: 41
Autovalore di modulo minimo: -0.557750429692594 + 2.49804953296681i
Errore commesso: 3.347281e-015

Riportiamo in basso l'andamento dell'errore:



Vediamo che, con una opportuna modifica dello *shift,* l'algoritmo riesce a convergere all'autovalore di modulo minimo corretto.

Per verificare la stabilità del sistema, però, non è sufficiente conoscere gli autovalori di modulo massimo e minimo, ma ci serve tutto lo spettro della matrice A.
Applichiamo quindi l'algoritmo QR attraverso la *function* qrbase.m.

Dopo 2001 iterazioni otteniamo la seguente matrice:

$$A_{k+1}=\begin{Vmatrix} -4.884499 & 3.5019 & -1.4525 \\ 0 & -0.87596 & 1.6894 \\ 0 & -3.7537 & -0.23954 \end{Vmatrix}$$

da cui si ricava immediatamente l'autovalore $\lambda = -4.884499$, mentre gli altri due autovalori si ricavano come autovalori della seguente sottomatrice 2x2:

$$B=\begin{Vmatrix} -0.87596 & 1.6894 \\ -3.7537 & -0.23954 \end{Vmatrix}$$

cioè: $\lambda_{2,1} = -0.55775 \pm 2.498i$

Concludendo possiamo dire che per k=3 l'origine è un punto di equilibrio asintoticamente stabile, o equivalentemente che il sistema risulta essere asintoticamente stabile in quanto i suoi autovalori hanno tutti parte reale negativa.

CASO 2: $a=2$ e k=7

La matrice A diventa quindi:

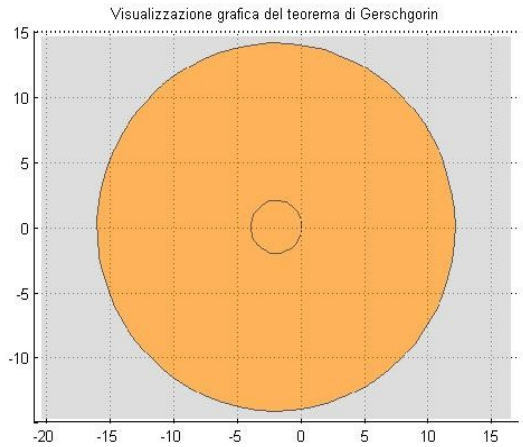
$$A=\begin{Vmatrix} -2 & 0 & -14 \\ 2 & -2 & 0 \\ 0 & 2 & -2 \end{Vmatrix}$$

Tale matrice è non singolare per cui possiamo subito affermare che l'origine è l'unico stato di equilibrio del sistema, dobbiamo ora vedere se questo è stabile o meno.
Per avere un'idea della localizzazione degli autovalori tracciamo i cerchi di Gershgorin:
Otteniamo i seguenti risultati:

-cerchi riga: $R_1=\{|z||z+2|\le 14\}$, $R_2=\{|z||z+2|\le 2\}$, $R_3=\{|z||z+2|\le 2\}$;

-cerchi colonna: $C_1=\{|z||z+2|\le 2\}$, $C_2=\{|z||z+2|\le 2\}$, $C_3=\{|z||z+2|\le 14\}$;

La loro intersezione ed unione è riportata nella figura sottostante in color arancio:



Notiamo che gli autovalori sono localizzati nella zona in arancio ovvero nell'intervallo [-16; 12].

Applichiamo ora l'algoritmo QR attraverso la *function* qrbase.m per conoscere lo spettro della matrice A.

Dopo 2001 iterazioni del metodo, risulta:

$$A_{k+1}=\begin{Vmatrix} -5.8259 & 3.1502 & -11.04 \\ 0 & -2.2941 & 5.732 \\ 0 & -2.765 & 2.12 \end{Vmatrix}$$

da cui si ricava immediatamente l'autovalore $\lambda = -5.8259$, mentre gli altri due autovalori si ricavano come autovalori della seguente sottomatrice 2x2:

$$B=\begin{Vmatrix} -2.2941 & 5.732 \\ -2.765 & 2.12 \end{Vmatrix}$$

cioè $\lambda_{2,1}$= -0.08705 +3.3133i

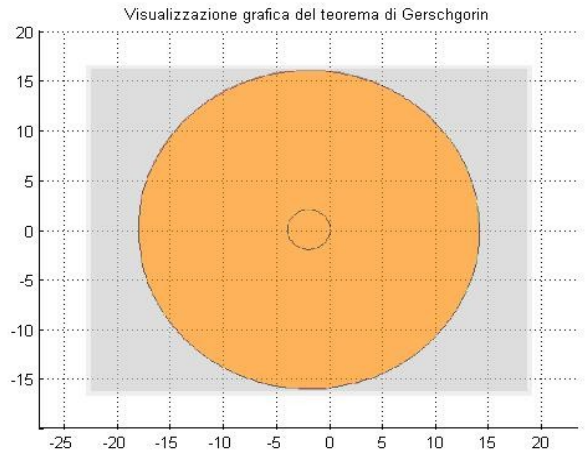
Poiché conosciamo tutti gli autovalori della matrice A possiamo affermare che per il criterio degli autovalori il sistema risulta asintoticamente stabile in quanto i suoi autovalori sono tutti a parte reale negativa.

CASO 3: $a=2$ e k=8

La matrice A è in questo caso:

$$A=\begin{Vmatrix} -2 & 0 & -16 \\ 2 & -2 & 0 \\ 0 & 2 & -2 \end{Vmatrix}$$

Tracciamo i cerchi di Gershgorin:



Gli autovalori sono localizzati nell'intervallo [-18, 14].

Possiamo osservare come all'aumentare del guadagno k, aumenta anche l' ampiezza dell'intervallo in cui sono compresi gli autovalori.
Applichiamo il metodo QR per calcolare gli autovalori della matrice:.

Dopo 2001 iterazioni del metodo, risulta:

$$A_{k+1} = \begin{Vmatrix} -6 & 13.416 & 2.5473e\text{-}12 \\ 0 & 1.2 & 1.833 \\ 0 & -7.3321 & -1.2 \end{Vmatrix}$$

ricaviamo immediatamente l'autovalore $\lambda = -5.8259$, gli altri vengono ricavati dalla sottomatrice B e sono pari a: $\lambda_{2,1} = 2.77555e\text{-}17 \pm 3.464102i$.

Il sistema per k=8 risulta instabile perché ha due autovalori a parte reale positiva. Quindi il guadagno dell'elettrovalvola non può essere troppo elevato altrimenti il sistema diventa instabile.
In conclusione possiamo affermare che per avere un funzionamento corretto dell'intero impianto si deve scegliere un circuito di comando con un guadagno k<8.