Analisi metodo iterativo Jacobi

In questa documentazione si mostreranno i casi di test relativi al metodo iterativo Jacobi.

Test di Accuratezza

Il test di accuratezza determina quanto la soluzione del sistema lineare trovata con Jacobi si avvicina a quella reale. I test sono stati effettuati utilizzando matrici sparse di grandi dimensioni non singolari e ben condizionate. E' stata implementata la funzione *CalcoloAccuratezza() che restituisce :* un grafico rappresentante la sparsità della matrice, l'*indice di condizionamento della matrice A, l'errore relativo, il residuo relativo e il numero di iterazioni impiegate per trovare la soluzione.*

Condizione Sufficiente di Convergenza $\rho(A) < 1$

Un metodo iterativo è convergente se il raggio spettrale della matrice di iterazione è minore di 1. Di seguito viene mostrato un esempio in Matlab che conferma la condizione.

```
A = spdiags(rand(100,1),0,100,100);
x = ones(100,1);
```

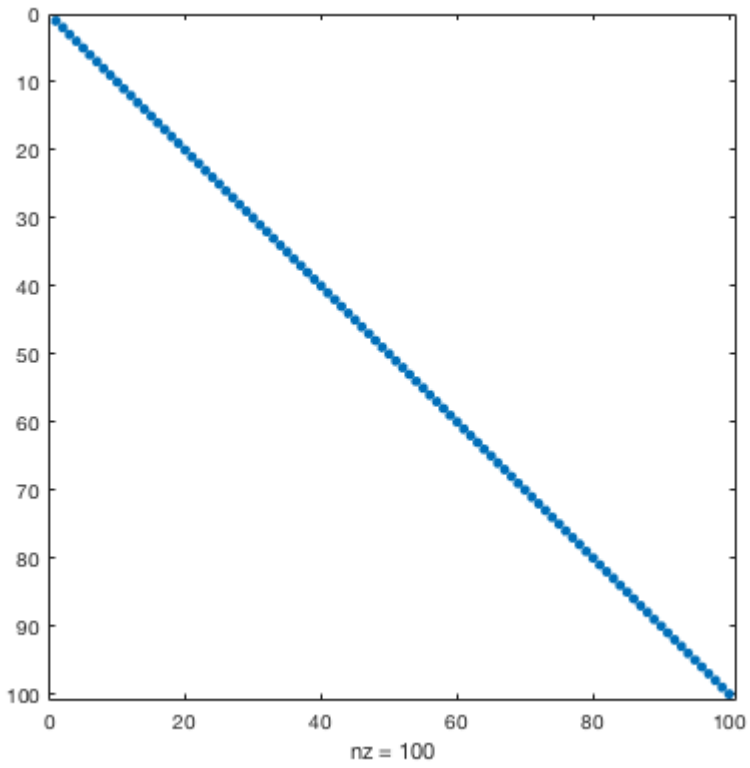
Il raggio spettrale è il massimo in modulo degli autovalori della matrice A.

```
rho = max(eig(A))
```

```
rho = 0.9984
```

Il raggio spettrale è minore di 1, soddisfa dunque l'ipotesi di convergenza. La funzione CalcoloAccuratezza() conferma che il metodo Jacobi è convergente per la matrice trattata ed in particolare il numero di iterazioni è 2.

```
[indice_cond,errore_rel,numiter,resid_rel] = CalcoloAccuratezza(A,x,eps,700)
```

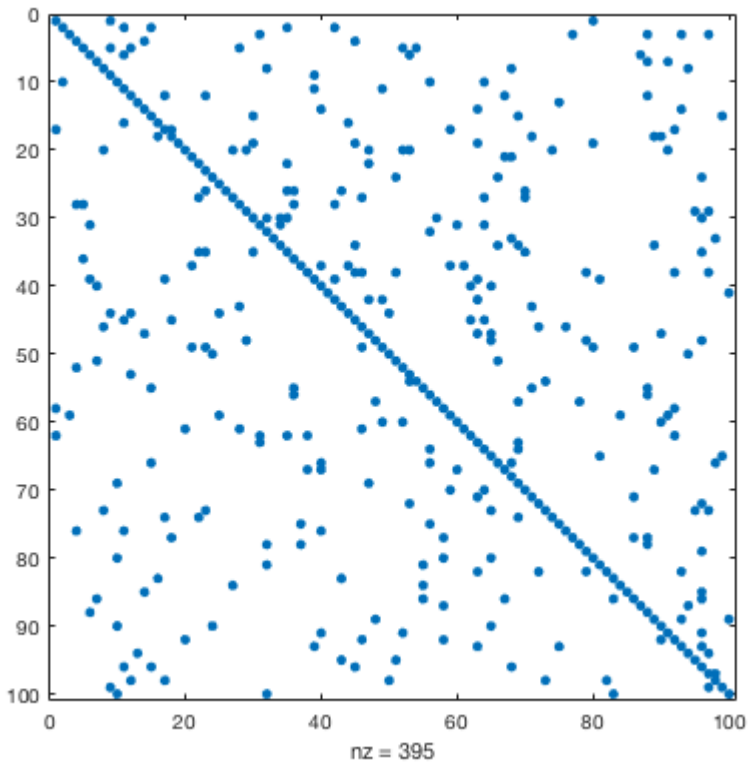



```
indice_cond = 817.0686
errore_rel = 0
numiter = 2
resid_rel = 0
```

Test con Matrice A con diagonale strettamente dominante $\rho(A) = \|A\|$.

La matrice ha diagonale principale strettamente dominante. Questa è una condizione sufficiente per la convergenza dell'algoritmo. Si sono richieste almeno 13 cifre significative corrette e al massimo 900 iterazioni.

```
A = sprand(100,100,0.03)+spdiags(ones(100,1),0,100,100)*3;
x = ones(100,1);
[indice_cond,errore_rel,numiter,resid_rel] = CalcoloAccuratezza(A,x,10^-13,900)
```




```
indice_cond = 8.0945
errore_rel = 1.6431e-14
numiter = 41
resid_rel = 4.6660e-15
```

Test con Matrice Malcondizionata e Sparsa '685_bus'

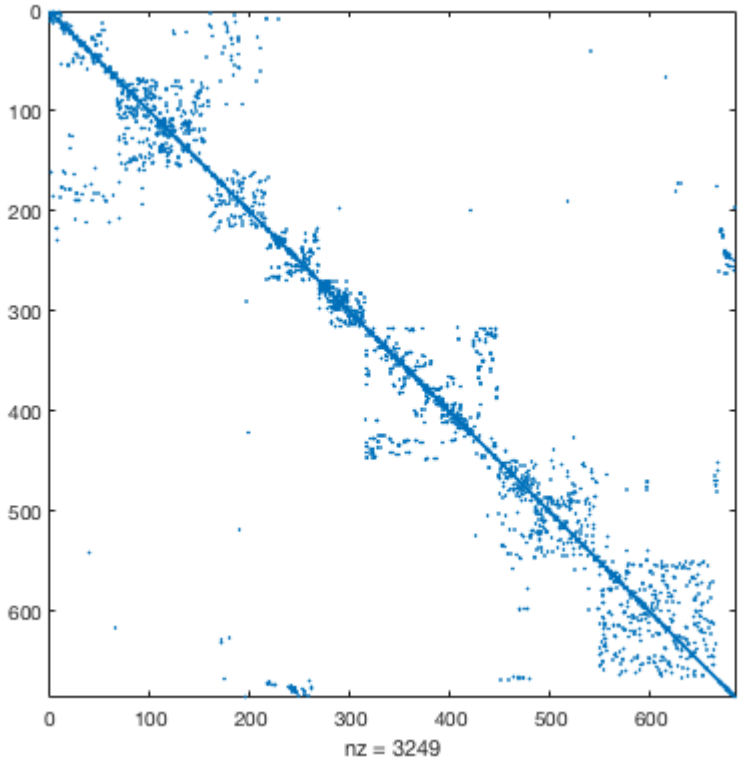
Questa matrice è stata prelevata dal seguente indirizzo web:

HB_685_bus

La matrice di input rappresenta un problema di rete elettrica avvenuto in Russia nel Luglio del 1985. Il numero degli zeri indica le interruzioni di corrente elettrica nei vari luoghi.

N.B : Il parametro 'Problem' estrae una struttura dal file .mat caricato, per caricare la matrice A basta eseguire il comando A = Problem.A

```
load('685_bus.mat','Problem');
A = Problem.A;
x = ones(length(A),1);
[indice_cond,errore_rel,numiter,resid_rel] = CalcoloAccuratezza(A,x,eps,800)
```

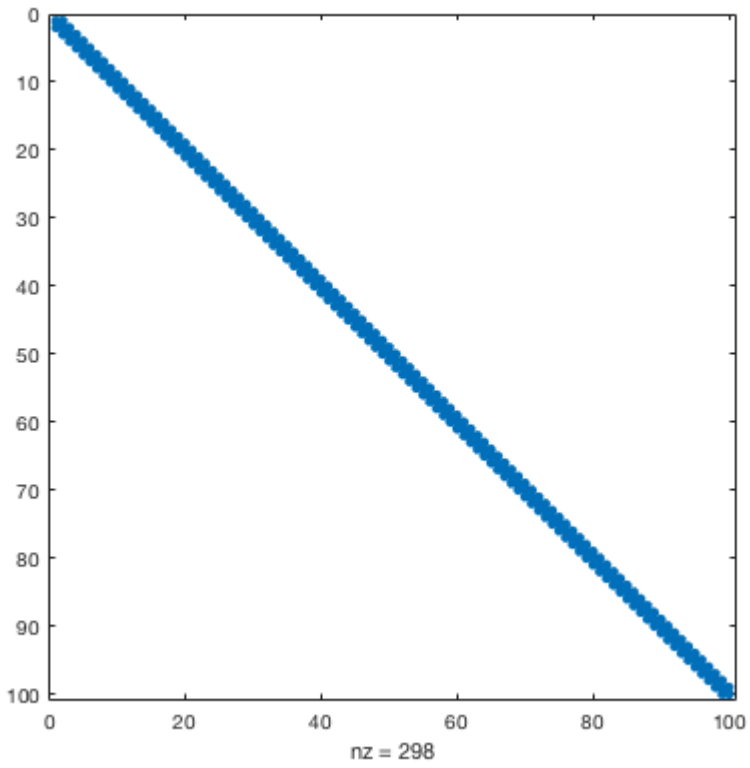



```
indice_cond = 5.3104e+05
errore_rel = 0.9200
numiter = 800
resid_rel = 1.5331e-04
```

L'errore relativo è circa 1, dunque non c'è nessuna cifra significativa corretta anche se ne abbiamo richieste 15 settando TOL=eps. Questo fenomeno avviene a causa del valore di condizionamento della matrice utilizzata.

Test con matrice A tridiagonale

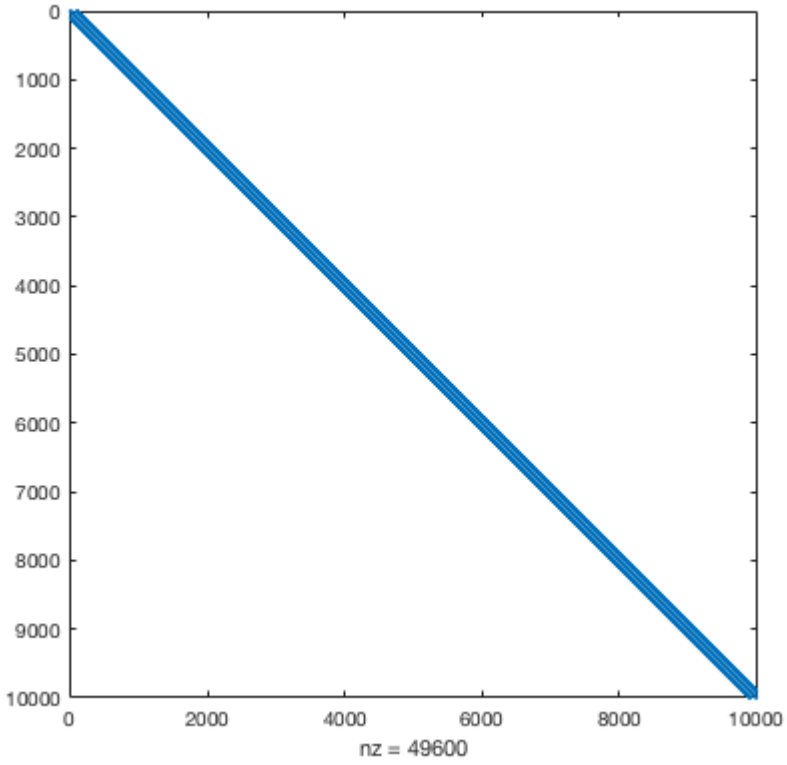
```
A = gallery('tridiag',100,10,35,22);
x = ones(100,1);
[indice_cond,errore_rel,numiter,resid_rel] = CalcoloAccuratezza(A,x,10^-8,900)
```




```
indice_cond = 22.3333
errore_rel = 4.3670e-09
numiter = 213
resid_rel = 1.7269e-09
```

Test con Matrice A di Poisson

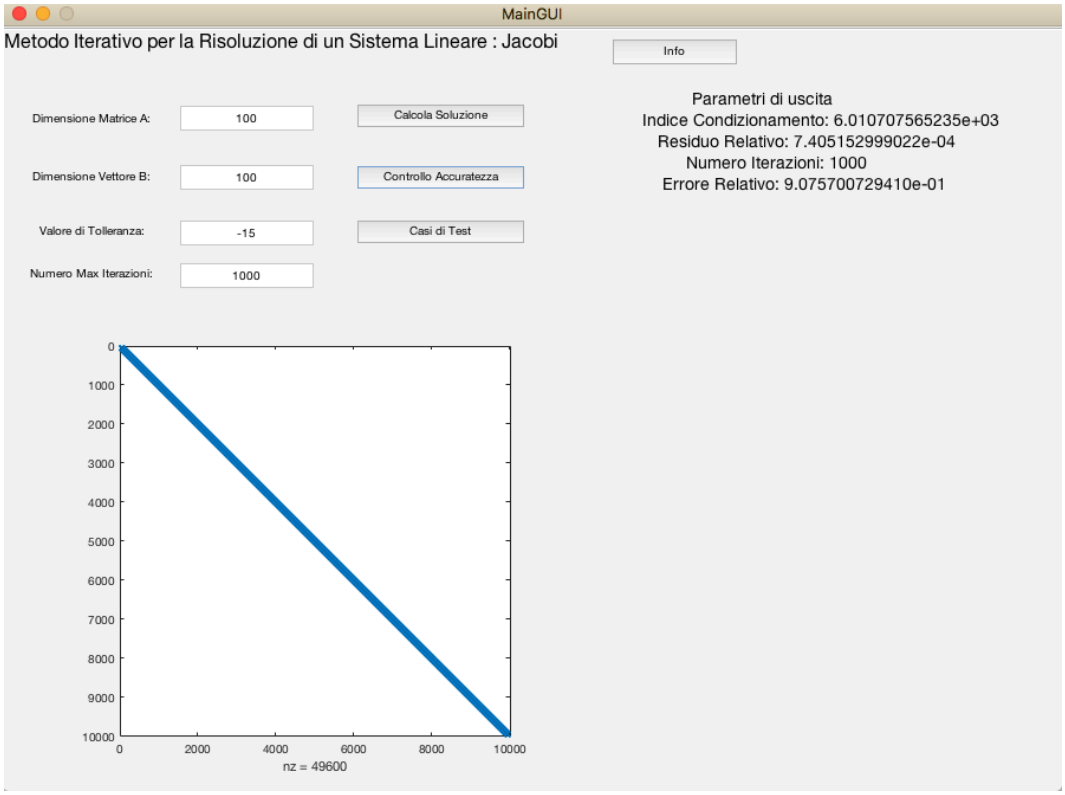
```
A = gallery('poisson',100);
x = ones(10000,1);
[indice_cond,errore_rel,numiter,resid_rel] = CalcoloAccuratezza(A,x,10^-12,30000)
```




```
indice_cond = 6.0107e+03
errore_rel = 8.0509e-07
numiter = 30000
resid_rel = 3.8955e-09
```

Contrariamente al caso della matrice 685_bus nella quale non vi era nessuna cifra significativa corretta nella soluzione, per la matrice di Poisson si nota che le cifre significative corrette restituite dalla soluzione sono 7 a fronte delle 12 che sono state richieste.

Esecuzione da interfaccia grafica



Valutazione delle Performance

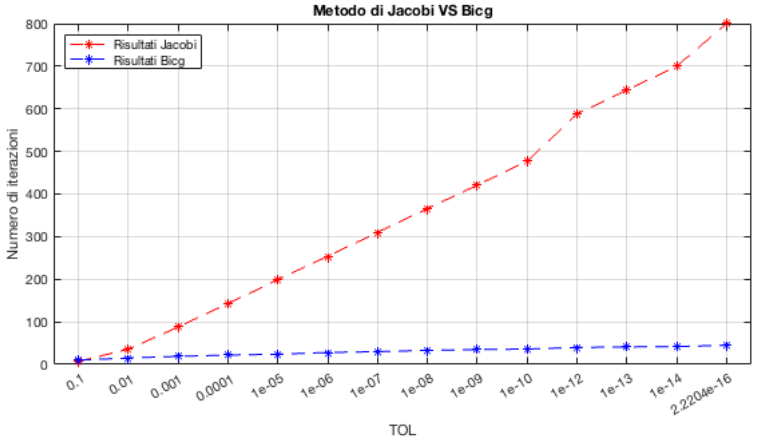
La valutazione delle performance dell'algoritmo mette alla luce le differenze del metodo di Jacobi implementato con la funzione del MATLAB "***bicg***". Entrambe richiedono gli stessi parametri di input (Matrice A, Vettore B , Tolleranza e Numero massimo di Iterazioni) e gli stessi parametri di Output (soluzione,numero iterazioni e residuo relativo). Le differenze tra i due metodi vengono mostrate su dei grafici che riportano il numero di iterazioni, l'indice di condizionamento della matrice in input ed errore e residuo relativo.

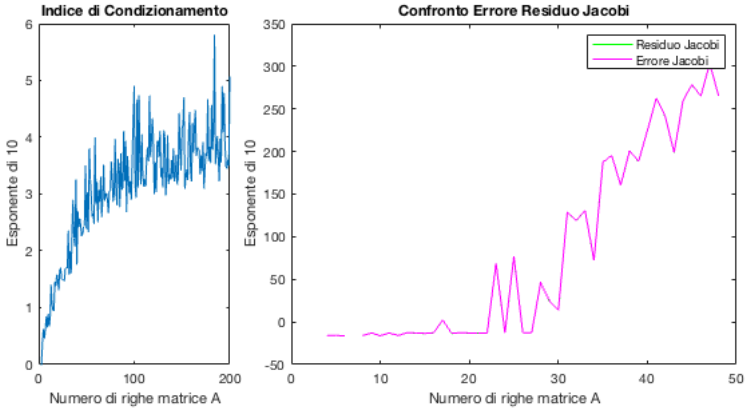
Sono stati scelti i seguenti parametri di input

- A : Matrice Sparsa di Poisson (presa dalla Gallery)
- b : Vettore casuale di elementi sparsi della stessa lunghezza di A
- TOL : Espresso come vettore di tolleranze da $10^{-1}\, a\, \mathrm{eps}$
- MAXITER : Numero massimo di iterazioni posto di default a 900

```
A = gallery('poisson',10);
b = sprand(length(A),1,0.2);
TOL=[10^-1 10^-2 10^-3  10^-4 10^-5 10^-6 10^-7 10^-8 10^-9 10^-10 10^-12 10^-13 10^-14 eps];
MAXITER = 900;
Valuta_Performance(A,b,TOL,MAXITER)
```

Grafico degli indici di condizionamento, Errore, Residuo e Numero Iterazioni per Jacobi/Bicg





Test di Robustezza

Per valutare la robustezza dell'algoritmo è stata implementata una test suite. I casi di test sono stati scelti a partire dalle condizioni di errore o warning che possono avere luogo. Ogni caso di test lo descriveremo

brevemente a partire dai parametri di input e dalla funzionalità che viene testata. Il numero di casi di test è: 15 ,implementati in una classe definita dal matlab *matlab.unittest.TestCase*. In tal modo si è automatizzato il processo di esecuzione dei test. Le istruzioni per eseguire i test saranno mostrate successivamente.

Di seguito seguono i casi di test più rilevanti che sono stati effettuati.

Test Case 1

Verifica se la matrice immessa non è sparsa

Input

- A = rand(10);
- b = A*ones(10,1);
- TOL = 10^-8;
- MAXITER = 700;

Test Case 3

Verifica se la matrice presenta uno zero sulla diagonale principale

Input

- A = sprand(10,10,0.1) + speye(10,10);
- A(1,1) = 0;
- b = A*ones(10,1);
- TOL = 10^-8;
- MAXITER = 700;

Test Case 4

Verifica se il vettore B non è un vettore

Input

- A = sprand(10,10,0.1) + speye(10,10);
- b = rand(10);
- TOL = 10^-8;
- MAXITER = 700;

Test Case 7

Verifica se il valore di tolleranza è negativo

- A = sprand(10,10,0.1) + speye(10,10);
- b = A*ones(10,1);
- TOL = -3;
- MAXITER = 10500;

Test Case 10

Verifica se il numero massimo di iterazioni è negativo

- A = sprand(10,10,0.1) + speye(10,10);
- b = A*ones(10,1);
- TOL = 10^-8;
- MAXITER = -1;

Test Case 11

Verifica se il numero massimo di iterazioni è molto grande

- A = sprand(10,10,0.1) + speye(10,10);
- b = A*ones(10,1);
- TOL = 10^-8;
- MAXITER = 10500;

Test Case 13

Verifica se la tolleranza e il numero massimo di iterazioni non sono specificati

Input

- A = sprand(10,10,0.1) + speye(10,10);
- b = A*ones(10,1);
- TOL = non specificato
- MAXITER = non specificato

Esecuzione Test suite

Vengono effettuati i test a partire dagli input definiti in precedenza.

```
result = runtests('CasiTest.m')
```

```
Running CasiTest
.......... .....
Done CasiTest
__________
result =
  1×15 TestResult array with properties:

    Name
    Passed
    Failed
    Incomplete
    Duration
    Details

Totals:
   15 Passed, 0 Failed, 0 Incomplete.
   1.4593 seconds testing time.
```

```
table(result)
```

```
ans = 15×6 table
```

	Name	Passed	Failed	Incomplete	Duration	Details
1	'CasiTest...	1	0	0	0.5500	1×1 struct
2	'CasiTest...	1	0	0	0.0318	1×1 struct
3	'CasiTest...	1	0	0	0.0421	1×1 struct
4	'CasiTest...	1	0	0	0.0309	1×1 struct
5	'CasiTest...	1	0	0	0.0796	1×1 struct
6	'CasiTest...	1	0	0	0.4498	1×1 struct
7	'CasiTest...	1	0	0	0.0086	1×1 struct
8	'CasiTest...	1	0	0	0.0439	1×1 struct
9	'CasiTest...	1	0	0	0.0400	1×1 struct
10	'CasiTest...	1	0	0	0.0079	1×1 struct
11	'CasiTest...	1	0	0	0.0447	1×1 struct
12	'CasiTest...	1	0	0	0.0155	1×1 struct
13	'CasiTest...	1	0	0	0.0711	1×1 struct
14	'CasiTest...	1	0	0	0.0260	1×1 struct
15	'CasiTest...	1	0	0	0.0173	1×1 struct

Riferimenti

- Testing in Matlab: https://it.mathworks.com/help/matlab/matlab-unit-test-framework.html
- Docenti.unina.it D'alessio Alessandra
- Wikipedia : Metodo di Jacobi

Autori

Giuseppe Napolano M63000856 Raffaele Formisano M63000912 Giuseppe Romito M63000936