Analisi Algoritmo Risolve

In questa documentazione si mostreranno i casi di test relativi all'algoritmo risolve.

Test di accuratezza

Il test di accuratezza da informazioni su qual'è l'errore prodotto dall'algoritmo risolve sulla soluzione calcolata. Infatti via software andiamo a risolvere un problema che è perturbato dove i dati del sistema hanno errore di round off. Non si potrà mai avere una soluzione del sistema che sia vera ma vogliamo capire quanto essa ci si avvicini . Si utilizzerà la funzione Calcolo_Accuratezza che a partire da un problema con relativa soluzione(**parametri di input**) calcola l'errore relativo della soluzione. La funzione restituirà oltre all'errore relativo anche l'indice di condizionamento.Quest'ultimo è strettamento legato all'errore che possiamo avere, infatti,ci da una misura di quanto l'errore di round off sui dati di input venga amplificato nell'errore di uscita.

Si analizzeranno tre casi:

1. **Matrice triangolare superiore:** utilizzo l'algoritmo back_substitution per risolvere il sistema.
2. **Matrice triangolare inferiore:** utilizzo l'algoritmo di forward_substitution per risolvere il sistema.
3. **Matrice piena:**Utilizzo l'algoritmo di Gauss con pivot parziale virtuale per calcolare la soluzione.

In ognuno dei 3 casi potremo avere che la matrice di input è mal condizionata o ben condizionata,nell'ultimo caso come si osserverà dal codice si impone che la matrice sia non singolare ponendo gli elementi della diagonale almeno pari ad 1.

```
%% Matrice Triangolare Superiore ben condizionata
A = triu(rand(200)) + 2*diag(ones(200,1));
x = ones(200,1);
b = A*x;
[indice_cond, err ] = Calcolo_Accuratezza(A,x,b,'sup')
```

```
indice_cond = 366.0049
err = 1.7871e-13
```

```
%% Matrice Triangolare Superiore mal condizionata
A = triu(rand(200));
x = ones(200,1);
b = A*x;
[indice_cond,err] =Calcolo_Accuratezza (A,x,b,'sup')
```

```
indice_cond = 3.6529e+18
err = 1.0535e+23
```

In tal caso si sono perse tutte le cifre significative del risultato.

```
%% Matrice Triangolare inferiore ben condizionata
A = tril(rand(200)) + 2*diag(ones(200,1));
x = ones(200,1);
```

```
b = A*x;
[indice_cond, ~] = Calcolo_Accuratezza(A,x,b,'inf')
```

```
indice_cond = 361.7838
```

```
%% Matrice Triangolare inferiore mal condizionata
A = tril(rand(200));
x = ones(200,1);
b = A*x;
[indice_cond, err] = Calcolo_Accuratezza(A,x,b,'inf')
```

```
indice_cond = 3.7506e+18
err = 8.9478e+27
```

Per le matrici piene che prevedono l'utilizzo dell'algoritmo di Gauss con pivoting si verifica la validità del **teorema di Wilkinson**.

```
%%Matrice Piena
A = rand(200);
x = ones(200,1);
b = A*x;
[indice_cond, err, residuo] =Calcolo_Accuratezza (A,x,b,'full')
```

```
indice_cond = 2.9050e+03
err = 6.4389e-14
residuo = 6.7471e-16
```

Il residuo e l'errore in tal caso sono quasi uguali la loro differenza è circa pari all'indice di condizionamento.

```
%%Matrice di Hilbert:molto malcondizionata
A =hilb(10);
x = ones(10,1);
b = A*x;
[indice_cond, err, residuo] = Calcolo_Accuratezza(A,x,b,'full')
```

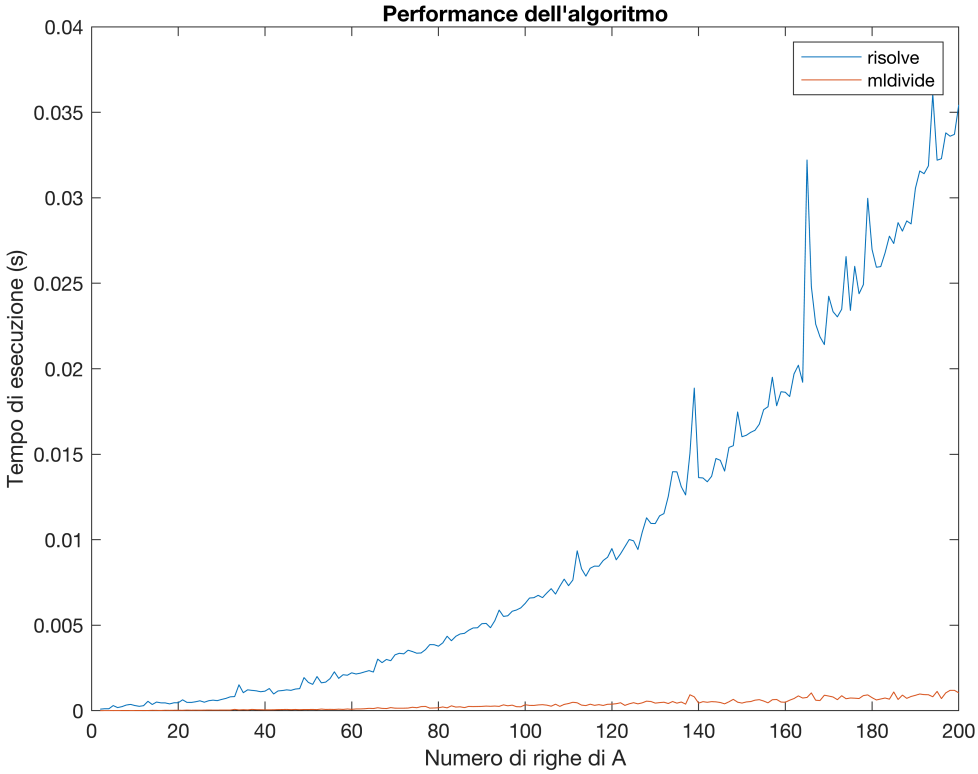
```
indice_cond = 1.6025e+13
err = 1.6307e-04
residuo = 2.0685e-16
```

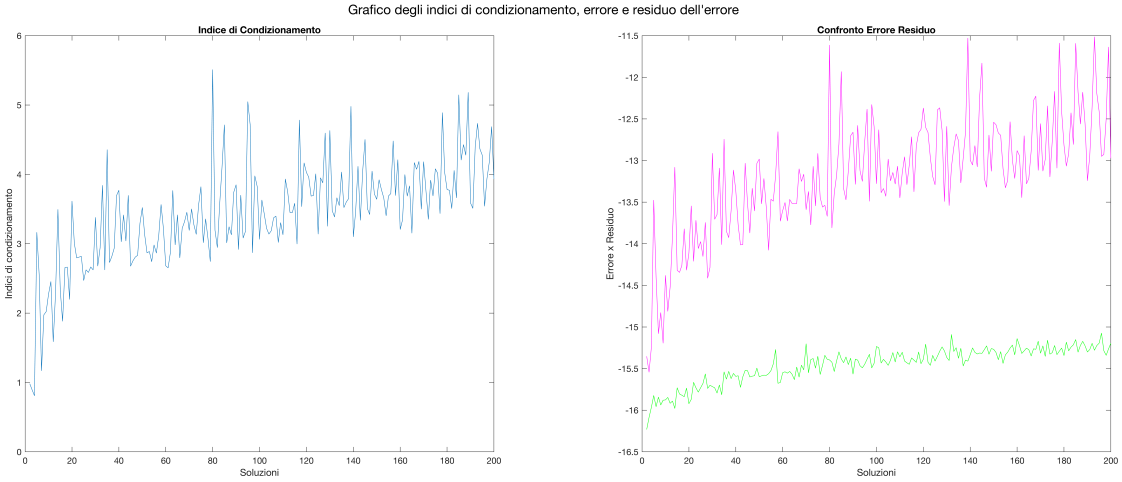
Nel caso delle matrici piene Calcolo_Accuratezza restituisce anche il residuo che notiamo essere tanto più vicino all'errore relativo quanto più l'indice di condizionamento è piccolo. Infatti l'errore per tali matrici(quando ben condizionate) può essere espresso come il prodotto tra l'indice di condizionamento ed il residuo stesso.

Valutazione Performance

Attraverso la funzione `valuta_performance` è stato valutato il tempo di esecuzione dell'algoritmo implementato. A partire da una matrice random di 200 elementi sono state effettuate tre misurazioni a seconda del tipo di matrici considerate.

```
Valuta_Performance(2,200,1,'full')
```





Nel caso della matrice piena è stato effettuato un confronto con i tempi di esecuzione della funzione mldivide(/). Quest'ultima è la migliore implementazione esistente dell'algoritmo di Gauss. Si è scelto per maggior chiarezza di graficare l'andamento dell'indice di condizionamento,errore e residuo per le matrici generate.

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: 16 ,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 il campo opt è una struttura.

Input

$A = \text{rand}(3, 3)$
$b = \text{rand}(3, 1)$
$\text{opt} = \text{'a'}$

Test case 2

Verifica l'errore nel caso in cui il campo di opt non sia uno tra: inf, sup, full.

Input

$A = \text{rand}(3, 3)$

$b = \text{rand}(3, 1)$

$\text{opt. sparsa} = \text{true}$

Test case 4

Verifica l'errore nel caso in cui più campi di opt assumono valore true

Input

$A = \text{rand}(3, 3)$
$b = \text{rand}(3, 1)$
$\text{opt. sup} = \text{true}$
$\text{opt. inf} = \text{true}$
$\text{opt. full} = \text{true}$

Test case 5

Verifica l'errore nel caso in cui la matrice A non è coerente con il campo opt specificato

Input

$A = \text{rand}(3, 3)$
$b = \text{rand}(3, 1)$
$\text{opt. inf} = \text{true}$
$\text{opt. full} = \text{false}$
$\text{opt. sup} = \text{false}$

Test case 7

Verifica l'errore nel caso in cui b non è un vettore reale.

Input

$A = \text{rand}(3, 3)$
$b = [3; 4; \text{'a'}]$
$\text{opt. full} = \text{true}$
$\text{opt. inf} = \text{false}$
$\text{opt. sup} = \text{false}$

Test case 9

Verifica l'errore nel caso in cui il vettore b ha un numero di righe diverso dal numero di righe di A.

Input

$A = \text{rand}(4)$
$b = [1; 2; 3]$
$\text{opt. full} = \text{true}$
$\text{opt. inf} = \text{false}$
$\text{opt. sup} = \text{false}$

Test case 11

Verifica l'errore nel caso in cui la matrice A è vuota

Input

$A = [\quad]$
$b = \text{rand}(3, 1)$
$\text{opt. full} = \text{true}$
$\text{opt. inf} = \text{false}$
$\text{opt. sup} = \text{false}$

Test case 13

Verifica l'errore nel caso in cui la matrice A non è numerica

Input

$A = [3, 2; , \text{'a'}, 4]$
$b = \text{rand}(3, 1)$
$\text{opt. full} = \text{true}$
$\text{opt. inf} = \text{false}$
$\text{opt. sup} = \text{false}$

Caso di test 14

Verifica l'errore nel caso in cui la matrice A è sparsa

Input

$A = \text{sparse}(A)$
$b = \text{rand}(3, 1)$
$\text{opt. full} = \text{true}$
$\text{opt. inf} = \text{false}$
$\text{opt. sup} = \text{false}$

Caso di test 16

Verifica l'errore nel caso in cui la matrice A non è quadratica.

Input

$A = [2\ 3\ 4; 4\ 5\ 3]$
$\text{b} = \text{rand}(3, 1)$
$\text{opt. full} = \text{true}$
$\text{opt. inf} = \text{false}$
$\text{opt. sup} = \text{false}$

Caso di test 18

Verifica l'errore nel caso in cui la matrice A è triangolare superiore e singolare.

Input

$A = \text{triu}(\text{rand}(3))$
$A(1, 1) = 0$
$b = \text{rand}(3, 1)$
$\text{opt. full} = \text{false}$
$\text{opt. inf} = \text{false}$
$\text{opt. sup} = \text{true}$

Esecuzione Test suite

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

```
result1 = runtests('TEST_SUITE.m')
```

```
Running TEST_SUITE
.......... ......
Done TEST_SUITE
__________
result1 =
  1×16 TestResult array with properties:

    Name
    Passed
    Failed
    Incomplete
    Duration
    Details

Totals:
```

```
16 Passed, 0 Failed, 0 Incomplete.
0.11257 seconds testing time.
```

```
table(result1)
```

ans = 16×6 table

	Name	Passed	Failed	Incomplete	Duration	Details
1	'TEST_SUI...	1	0	0	0.0064	1×1 struct
2	'TEST_SUI...	1	0	0	0.0095	1×1 struct
3	'TEST_SUI...	1	0	0	0.0093	1×1 struct
4	'TEST_SUI...	1	0	0	0.0067	1×1 struct
5	'TEST_SUI...	1	0	0	0.0052	1×1 struct
6	'TEST_SUI...	1	0	0	0.0064	1×1 struct
7	'TEST_SUI...	1	0	0	0.0081	1×1 struct
8	'TEST_SUI...	1	0	0	0.0110	1×1 struct
9	'TEST_SUI...	1	0	0	0.0090	1×1 struct
10	'TEST_SUI...	1	0	0	0.0057	1×1 struct
11	'TEST_SUI...	1	0	0	0.0049	1×1 struct
12	'TEST_SUI...	1	0	0	0.0053	1×1 struct
13	'TEST_SUI...	1	0	0	0.0060	1×1 struct
14	'TEST_SUI...	1	0	0	0.0058	1×1 struct
15	'TEST_SUI...	1	0	0	0.0070	1×1 struct
16	'TEST_SUI...	1	0	0	0.0064	1×1 struct

Riferimenti

- Testing in Matlab: https://it.mathworks.com/help/matlab/matlab-unit-test-framework.html

Autori

Giuseppe Napolano M63000856 Raffaele Formisano M63000912 Giuseppe Romito M63000936