

# 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 strettamente legato all'errore che possiamo avere, infatti, ci dà 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 `pivoting` 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 = 367.3520
err = 1.7100e-14
```

```
% 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 = 3.2998e+18
err = 2.2819e+10
```

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, err] = Calcolo_Accuratezza(A,x,b,'inf')
```

```
indice_cond = 515.1200
err = 1.3466e-14
```

```
% 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 = 8.5362e+19
err = 3.8362e+14
```

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 = 7.1313e+03
err = 1.8287e-13
residuo = 6.8872e-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
```

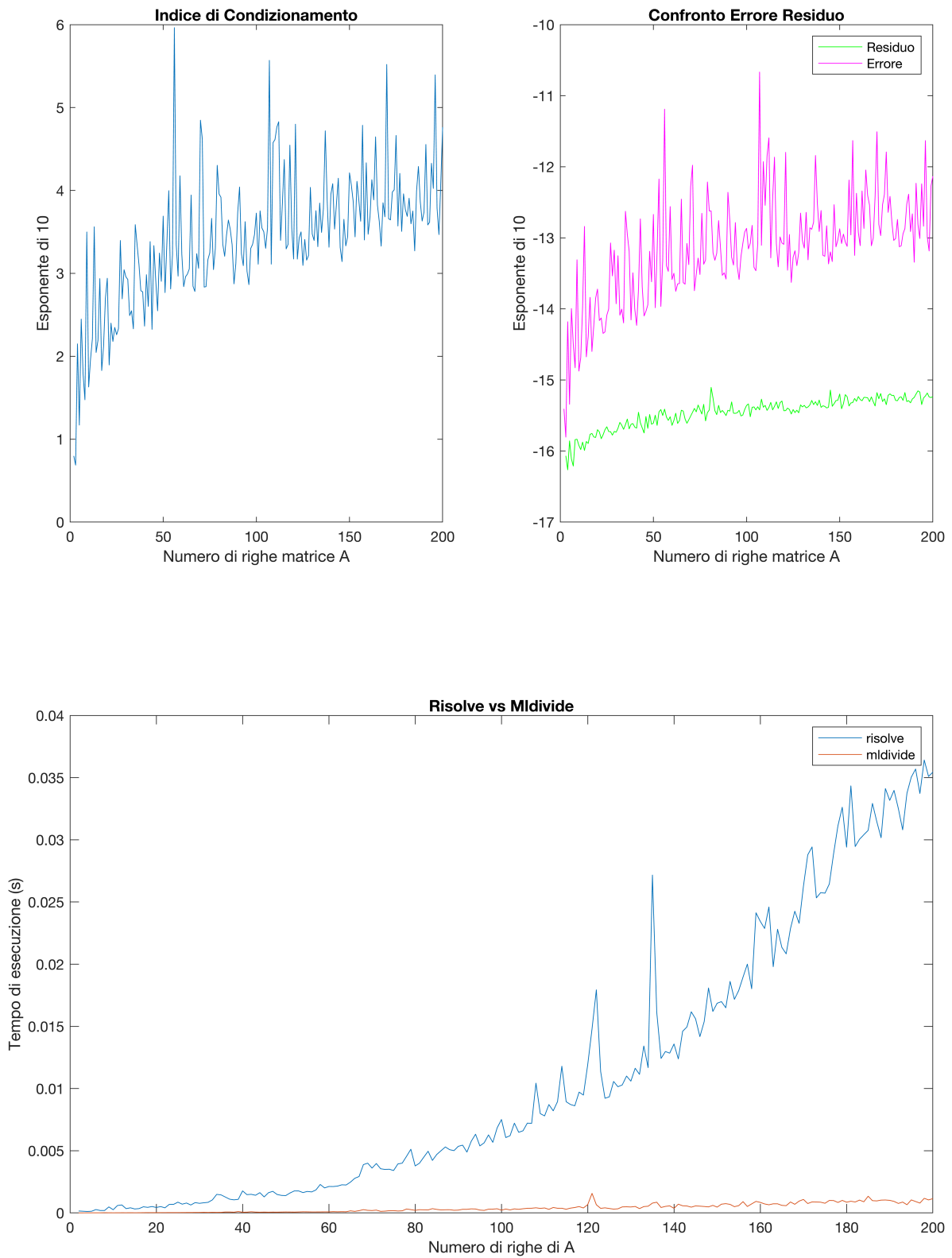
Nel caso delle matrici piene Calcolo\_Accuratezza restituisce anche il **residuo** che notiamo essere tanto più vicino all'errore relativo quanto più l'indice di condizionamento è piccolo. Infatti l'errore per tali matrici(quando ben condizionate) può essere espresso come il prodotto tra l'indice di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

## Valutazione Performance

Attraverso la funzione `valuta_performance` è stato valutato il tempo di esecuzione dell'algoritmo implementato. A partire dalla generazione di matrici random da 1 fino a 200 elementi sono state effettuate tre misurazioni a seconda del tipo di matrici considerate.

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

Grafico degli indici di condizionamento, errore e residuo



Nel caso della matrice piena è stato effettuato un confronto con i tempi di esecuzione della funzione `mldivide(/)`. Quest'ultima è il miglior algoritmo presente in Matlab per la risoluzione di sistemi lineari. 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 sono mostrati i casi di test più rilevanti che sono stati effettuati ,rispettando la precedenza tra essi.

### Test case 1

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

#### Input

```
A = [ ]  
b = rand(3,1)  
opt.full = true  
opt.inf = false  
opt.sup = false
```

### Test case 2

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

#### Input

```
A = [3,2;,'a',4]  
b = rand(3,1)  
opt.full = true  
opt.inf = false  
opt.sup = false
```

### Caso di test 3

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

#### Input

```
A = sparse(A)
b = rand(3, 1)
opt. full = true
opt. inf = false
opt. sup = false
```

## Caso di test 5

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

### Input

```
A = [2 3 4; 4 5 3]
b = rand(3, 1)
opt. full = true
opt. inf = false
opt. sup = false
```

## Test case 6

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

### Input

```
A = rand(3, 3)
b = [3; 4; 'a']
opt. full = true
opt. inf = false
opt. sup = false
```

## Test case 8

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

### Input

```
A = rand(4)
b = [1; 2; 3]
opt. full = true
opt. inf = false
opt. sup = false
```

## Test case 9

Verifica se il campo opt è una struttura.

### Input

```
A = rand(3, 3)
b = rand(3, 1)
opt = 'a'
```

## Test case 10

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

### Input

```
A = rand(3, 3)
b = rand(3, 1)

opt. sparsa = true
```

## Test case 12

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

### Input

```
A = rand(3, 3)
b = rand(3, 1)
opt. sup = true
opt. inf = true
opt. full = true
```

## Test case 13

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

### Input

```
A = rand(3, 3)
b = rand(3, 1)
opt. inf = true
opt. full = false
opt. sup = false
```

## Caso di test 16

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

### Input

```

A = triu(rand(3))
A(1,1) = 0
b = rand(3,1)
opt.full = false
opt.inf = false
opt.sup = true

```

**Osservazione :** Stesso discorso vale per il caso di test in cui la matrice  $A$  è triangolare inferiore e singolare. Inoltre questa condizione non viene verificata sulla matrice di input per l'algoritmo di Gauss implementato. Infatti, la versione implementata con pivot parziale trova **il pivot** come massimo della relativa sottocolonna. Nel caso in cui il massimo trovato sia zero, la matrice sarà singolare. Dunque per tale ragionamento si è scelto di restituire "errore" quando si verifica tale condizione nel corpo dell'algoritmo di Gauss implementato.

## 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 =
```

```
1x16 TestResult array with properties:
```

```

Name
Passed
Failed
Incomplete
Duration
Details

```

```
Totals:
```

```
16 Passed, 0 Failed, 0 Incomplete.
```

```
0.82409 seconds testing time.
```

```
table(result1)
```

```
ans = 16x6 table
```

	Name	Passed	Failed	Incomplete	Duration	Details
1	'TEST_SUI...	1	0	0	0.4908	1x1 struct
2	'TEST_SUI...	1	0	0	0.0420	1x1 struct
3	'TEST_SUI...	1	0	0	0.0449	1x1 struct
4	'TEST_SUI...	1	0	0	0.0266	1x1 struct
5	'TEST_SUI...	1	0	0	0.0998	1x1 struct
6	'TEST_SUI...	1	0	0	0.0228	1x1 struct
7	'TEST_SUI...	1	0	0	0.0114	1x1 struct
8	'TEST_SUI...	1	0	0	0.0096	1x1 struct

	Name	Passed	Failed	Incomplete	Duration	Details
9	'TEST_SUI...	1	0	0	0.0196	1×1 struct
10	'TEST_SUI...	1	0	0	0.0085	1×1 struct
11	'TEST_SUI...	1	0	0	0.0069	1×1 struct
12	'TEST_SUI...	1	0	0	0.0117	1×1 struct
13	'TEST_SUI...	1	0	0	0.0096	1×1 struct
14	'TEST_SUI...	1	0	0	0.0065	1×1 struct
15	'TEST_SUI...	1	0	0	0.0061	1×1 struct
16	'TEST_SUI...	1	0	0	0.0072	1×1 struct

## Riferimenti

- Testing in Matlab: <https://it.mathworks.com/help/matlab/matlab-unit-test-framework.html>
- Misura di Performance in Matlab <https://it.mathworks.com/help/matlab/performance-and-memory.html>

## Autori

**Giuseppe Napolano M63000856 Raffaele Formisano M63000912 Giuseppe Romito M63000936**