

Analisi Algoritmo di Bisezione

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

Test di accuratezza

Il test di accuratezza determina quanto la soluzione approssimata trovata attraverso il metodo di bisezione si avvicina alla soluzione dell'algoritmo *fzero()*. Quest'ultimo è il migliore algoritmo utilizzato per trovare lo zero di una funzione in Matlab. Ai fini del calcolo dell' accuratezza è stata implementata la funzione *CalcoloAccuratezza()* . Essa calcola l'errore relativo tra le soluzioni restituite dai due algoritmi al medesimo problema.

Esempio

Calcolare lo zero di $2 - e^{-x} - \sqrt{x}$ nell'intervallo $[0, 4]$ al variare della tolleranza TOL inserita dall'utente supponendo NMAX di default.

Command line Matlab

```
f = @(x)(2-exp(-x)-sqrt(x));  
x0 = [0 4];
```

```
CalcoloAccuratezza(f,x0,1e-10)
```

```
ans = 4.5544e-11
```

```
CalcoloAccuratezza(f,x0,1e-11)
```

```
ans = 1.7684e-12
```

```
CalcoloAccuratezza(f,x0,1e-12)
```

```
ans = 3.7669e-13
```

```
CalcoloAccuratezza(f,x0,1e-13)
```

```
ans = 2.2651e-16
```

```
CalcoloAccuratezza(f,x0,1e-14)
```

```
ans = 2.2651e-16
```

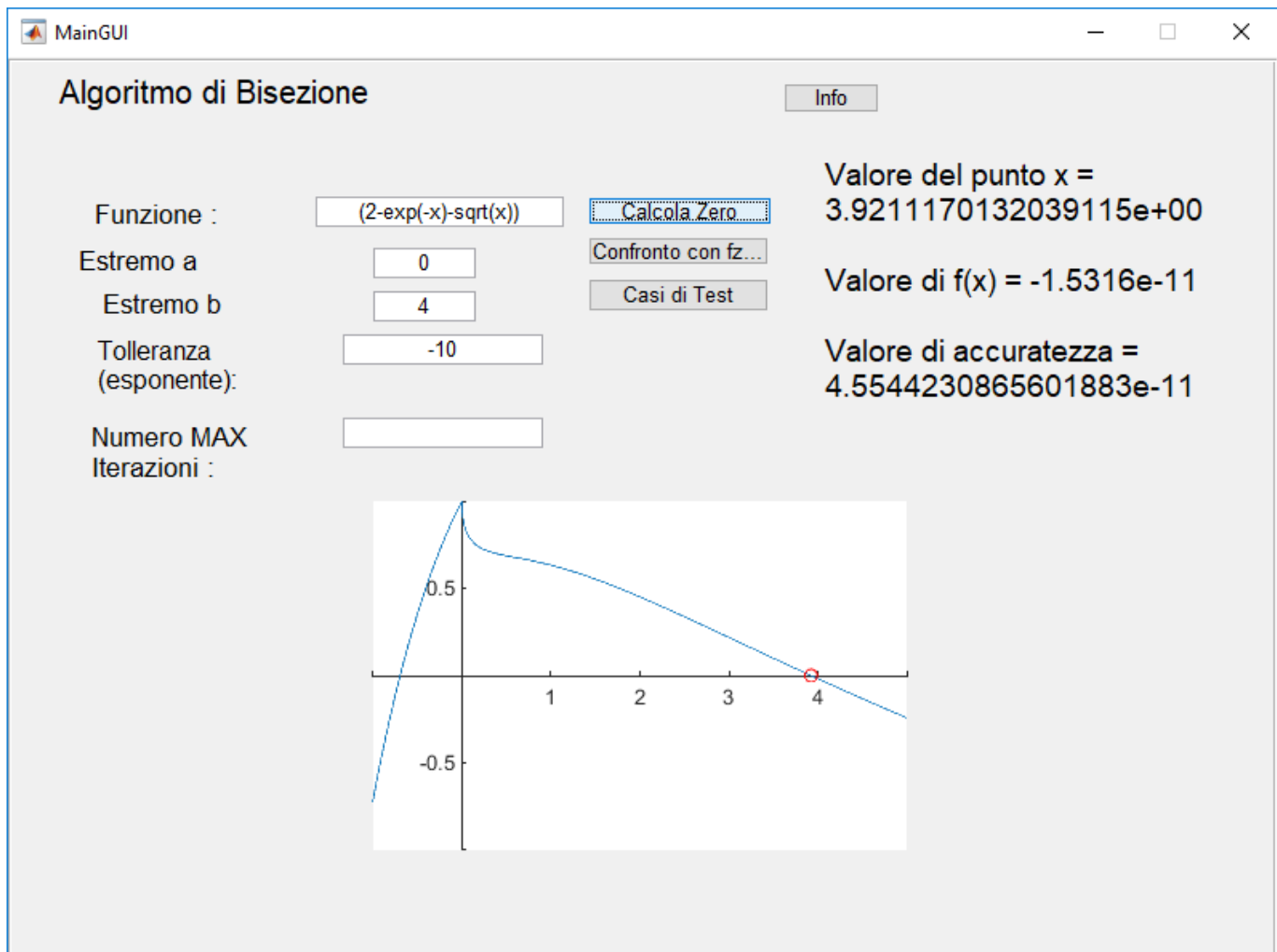
```
CalcoloAccuratezza(f,x0,1e-15)
```

```
ans = 2.2651e-16
```

```
CalcoloAccuratezza(f,x0,eps)
```

```
ans = 2.2651e-16
```

Esecuzione da interfaccia grafica



Valutazione Performance

Si è implementata una funzione che crea un grafico di confronto tra i due algoritmi sopra citati. Si è calcolata la performance confrontando il numero di iterazioni necessarie all'algoritmo di Bisezione con quelle necessarie alla funzione `fzero()` considerata la tolleranza richiesta.

Esempio

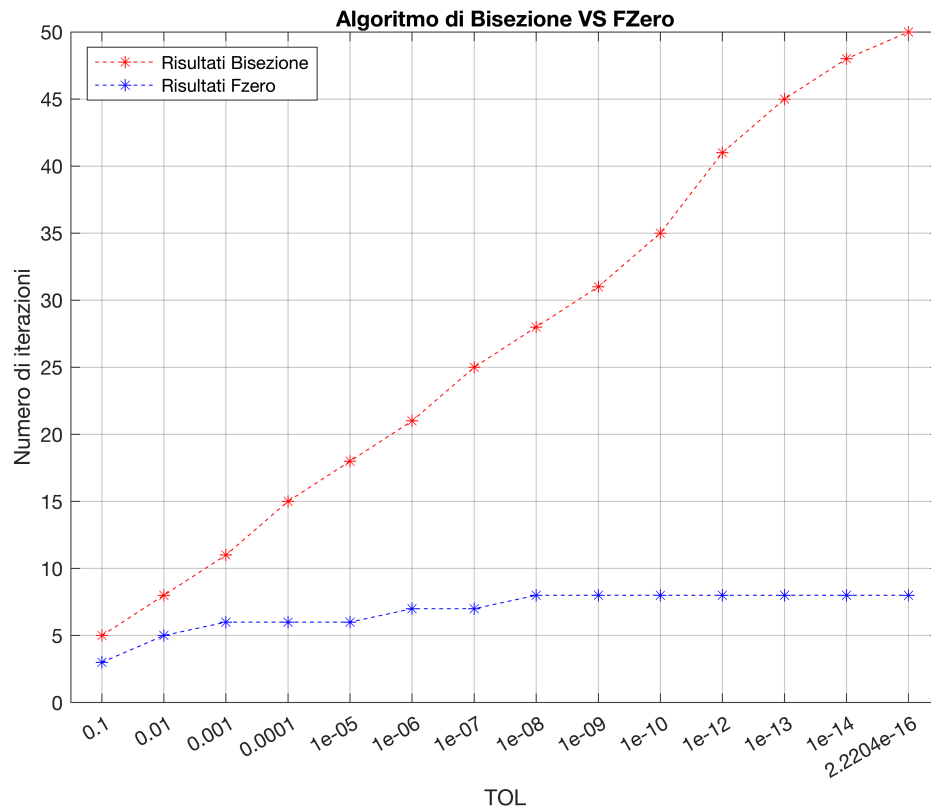
Calcola la performance dell'algoritmo di bisezione quando è in input la funzione $\sin(x) + \cos(x) - x^2 + 4$ nell'intervallo $[0, 2\pi]$. Ciò viene calcolato al variare della tolleranza TOL inserita dall'utente e supponendo NMAX di default. La funzione restituirà un grafico di confronto tra i due approcci.

Command line Matlab

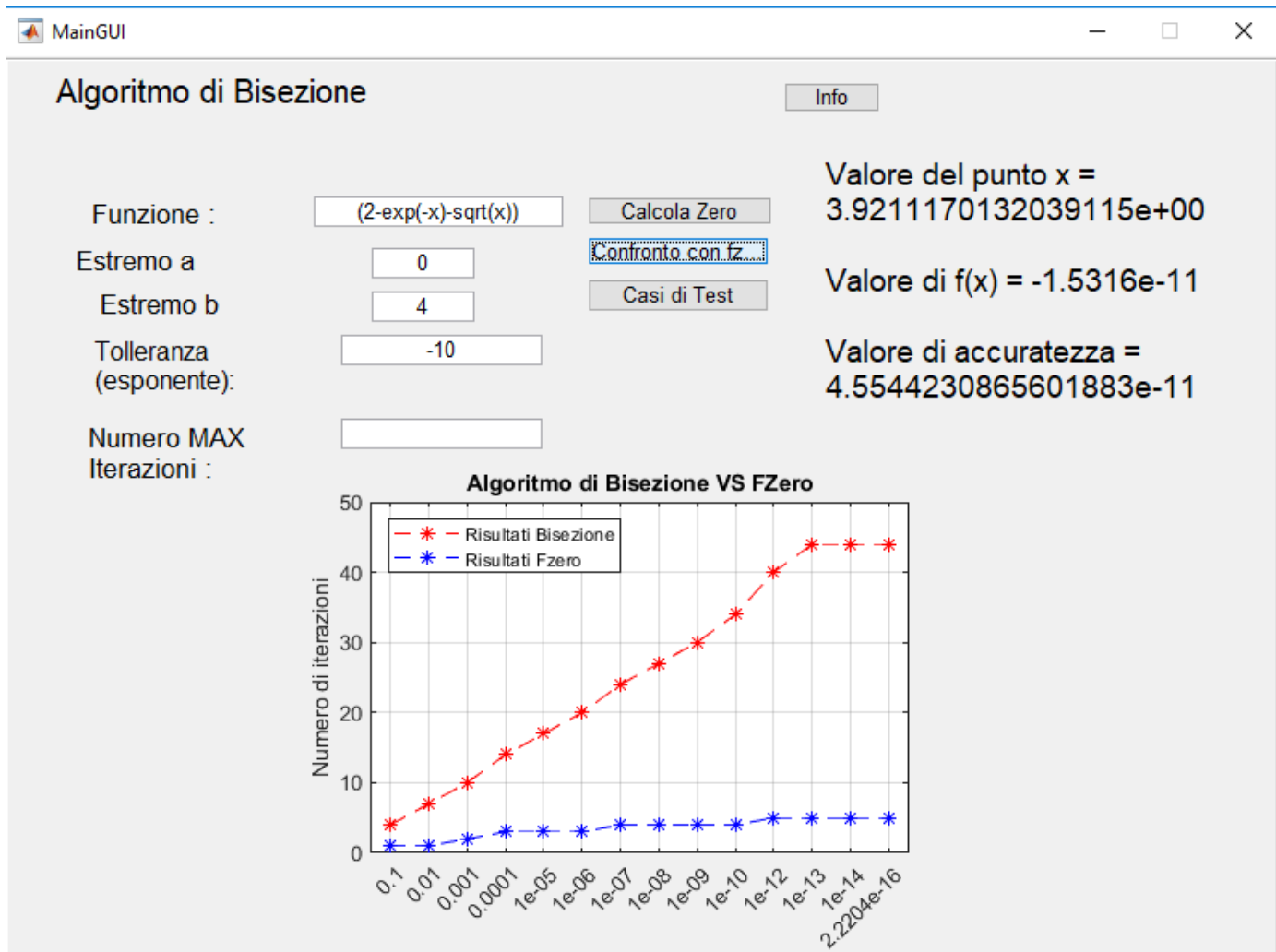
```
warning('off');  
warning;
```

All warnings have the state 'off'.

```
f = @(x) (sin(x)+cos(x)-x.^2+4);%funzione handle  
xo = [0 2*pi];  
%Vettore di tolleranze da 10^-1 sino a eps  
TOL=[10.^-1 10.^-2 10.^-3 10.^-4 10.^-5 10.^-6 10.^-7 10.^-8 10.^-9 10.^-10 10.^-12  
Valuta_Performance(f,x0,TOL);
```



Esecuzione da interfaccia grafica



Come si nota dal grafico il numero di iterazioni richieste dalla funzione `fzero()` è minore rispetto a quelle della funzione `algoritmo_di_bisezione()` per ogni valore di TOL. Si può osservare che la differenza tra i due approcci è tanto più marcata quanto più si cerca una risoluzione migliore diminuendo la tolleranza.

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 2

Verifica l'errore nel caso in cui il valore della funzione non sia handle.

Input

$f = 'a'$
 $x0 = [0\ 4]$
 $TOL = 10^{-15}$
 $NMAX = 600$

Test case 4

Verifica l'errore nel caso in cui il primo estremo o il secondo estremo dell'intervallo inserito non sia numero.

Input

$f = x^2 - 4$
 $x0 = ['a'\ 4]$
 $TOL = 10^{-15}$
 $NMAX = 600$

Test case 6

Verifica l'errore nel caso in cui entrambi i valori dell'intervallo $x0$ siano uguali.

Input

$f = x^2 - 4$
 $x0 = [3\ 3]$
 $TOL = 10^{-15}$
 $NMAX = 600$

Test case 7

Verifica l'errore nel caso in cui non è soddisfatto il teorema degli zeri.

Input

$f = x^2 - 4$
 $x0 = [-3\ 3]$
 $TOL = 10^{-15}$
 $NMAX = 600$

Test case 8

Verifica che la tolleranza sia stata inserita o meno dall'utente.

Input

$$f = x^2 - 4$$

$$x0 = [0 \ 4]$$

TOL = non inserito

NMAX = non inserito

Test case 9

Si verifica se la tolleranza inserita è minore di eps.

Input

$$f = x^2 - 4$$

$$x0 = [0 \ 4]$$

TOL = -1

NMAX = 600

Test case 16

Verifica se l'accuratezza è adeguata a partire da una determinata tolleranza TOL.

Input

$$f = x^2 - 4$$

$$x0 = [0 \ 4]$$

TOL = 10^{-15}

NMAX = 600

Test case 11

Verifica se il valore di NMAX non è numerico, scalare, infinito oppure NaN.

Input

$$f = x^2 - 4$$

$$x0 = [0 \ 4]$$

TOL = 10^{-15}

NMAX = 'a'

Caso di test 13

Verifica se il valore di NMAX è minore 2.

Input

$$f = x^2 - 4$$

$$x0 = [0 \ 4]$$

TOL = 10^{-15}

NMAX = 1

Caso di test 15

Verifica se si è superato il numero massimo di iterazioni per determinare l'uscita. Non sarà trovato lo zero.

Input

$$f = 2 - e^{-x} - \sqrt{x}$$

$x0 = [0\ 4]$

$TOL = 10^{-15}$

$NMAX = 5$

Esecuzione Test suite

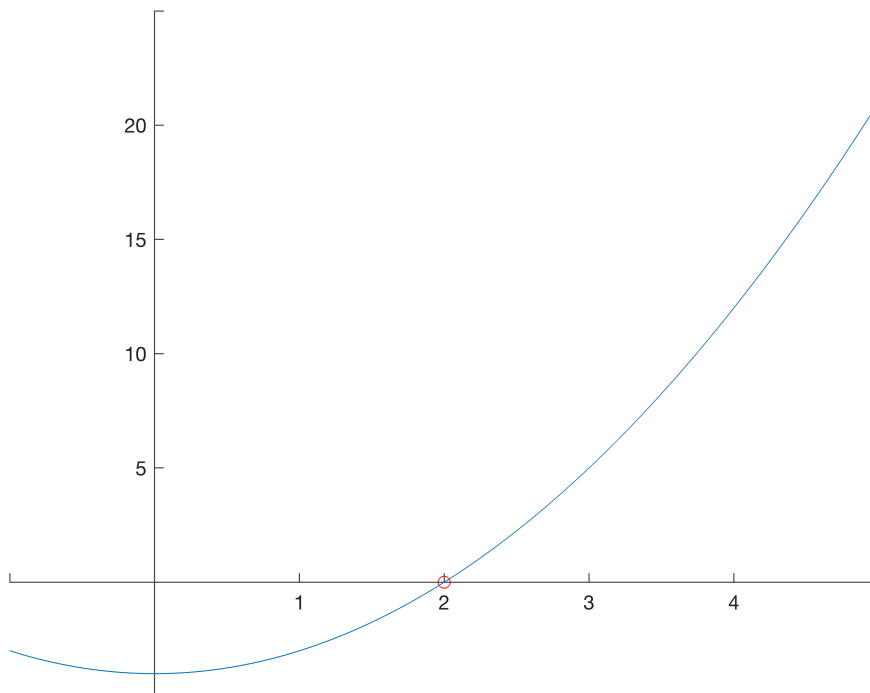
Vengono effettuati i test a partire dagli input definiti in precedenza.

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

Running TEST_RICHIAMA_PARAMETRI

.....

Warning: Numero iterazioni massimo non specificato, uso 500 come valore di default



Done TEST_RICHIAMA_PARAMETRI

result1 =
1x15 TestResult array with properties:

Name
Passed
Failed
Incomplete
Duration

Details

Totals:

15 Passed, 0 Failed, 0 Incomplete.
5.6743 seconds testing time.

```
table(result1)
```

```
ans = 15x6 table
```

	Name	Passed	Failed	Incomplete	Duration	Details
1	'TEST_RIC...	1	0	0	0.4265	1×1 struct
2	'TEST_RIC...	1	0	0	0.0349	1×1 struct
3	'TEST_RIC...	1	0	0	0.0299	1×1 struct
4	'TEST_RIC...	1	0	0	0.0221	1×1 struct
5	'TEST_RIC...	1	0	0	0.0780	1×1 struct
6	'TEST_RIC...	1	0	0	0.0100	1×1 struct
7	'TEST_RIC...	1	0	0	0.4209	1×1 struct
8	'TEST_RIC...	1	0	0	0.0214	1×1 struct
9	'TEST_RIC...	1	0	0	0.0090	1×1 struct
10	'TEST_RIC...	1	0	0	0.0085	1×1 struct
11	'TEST_RIC...	1	0	0	0.0084	1×1 struct
12	'TEST_RIC...	1	0	0	0.0052	1×1 struct
13	'TEST_RIC...	1	0	0	4.4014	1×1 struct
14	'TEST_RIC...	1	0	0	0.0393	1×1 struct
15	'TEST_RIC...	1	0	0	0.1590	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