Matlab e le equazioni non lineari

MATLAB mette a disposizione, per la ricerca degli zeri di una funzione non lineare la routine *fzero .*

Algorithms

The `fzero` command is a function file. The algorithm, created by T. Dekker, uses a combination of bisection, secant, and inverse quadratic interpolation methods. An Algol 60 version, with some improvements, is given in [1]. A Fortran version, upon which `fzero` is based, is in [2].

Dal prompt dei comandi è possibile richiamare a funzione con le relative opzioni:

```
>> x=fzero(fun,x0)
>>x=fzero(fun,x0,options)
>>[x,fval]=fzero(…)
>>[x,fval,exitflag]=fzero(…)
 >>[x,fval,exitflag,output]=fzero(…)
```

Input Arguments

› `fun` — Function to solve
function handle

› `x0` — Initial value
scalar | 2-element vector

› `options` — Options for solution process
structure, typically created using `optimset`

Output Arguments

› `x` — Location of root or sign change
real scalar

› `fval` — Function value at `x`
real scalar

› `exitflag` — Integer encoding the exit condition
integer

› `output` — Information about root-finding process
structure

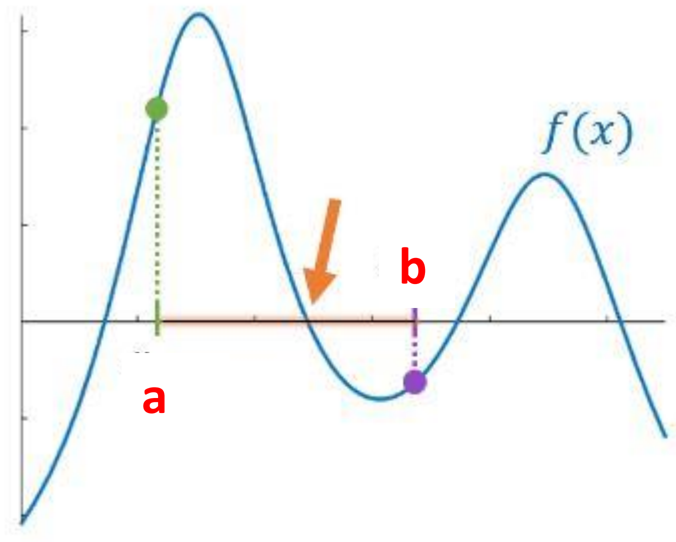
Calcolo dello zero di una funzione continua in [a,b]

f∈ [a,b], continua e tale che

$f(a)f(b)<0$

esiste almeno un punto z :

$f(z)=0$



- `options` — Options for solution process
  structure, typically created using `optimset`

  Options for solution process, specified as a structure. Create or modify the `options` structure using `optimset`. `fzero` uses these `options` structure fields.

```
>> X=fzero(@sin,3,optimset('disp','iter'))

Search for an interval around 3 containing a sign change:
 Func-count    a          f(a)             b          f(b)        Procedure
    1               3       0.14112             3       0.14112   initial interval
    3         2.91515      0.224515       3.08485     0.0567094   search
    5            2.88      0.258619          3.12      0.021591   search
    7         2.83029      0.306295       3.16971    -0.0281093   search
```

Fase1: ricerca dell’intervallo

· `options` — Options for solution process
structure, typically created using `optimset`

Options for solution process, specified as a structure. Create or modify the `options` structure using `optimset`. `fzero` uses these `options` structure fields.

```
>> X=fzero(@sin,3,optimset('disp','iter'))

Search for an interval around 3 containing a sign change:
 Func-count    a          f(a)             b          f(b)        Procedure
    1               3       0.14112             3       0.14112   initial interval
    3         2.91515      0.224515       3.08485     0.0567094   search
    5            2.88      0.258619          3.12      0.021591   search
    7         2.83029      0.306295       3.16971    -0.0281093   search
```

Fase1: ricerca dell'intervallo

intervallo

```
Search for a zero in the interval [2.83029, 3.16971]:
```

options — Options for solution process
structure, typically created using `optimset`

Options for solution process, specified as a structure. Create or modify the `options` structure using `optimset`. `fzero` uses these `options` structure fields.

```
>> X=fzero(@sin,3,optimset('disp','iter'))

Search for an interval around 3 containing a sign change:
 Func-count    a          f(a)             b          f(b)        Procedure
    1               3       0.14112              3       0.14112   initial interval
    3         2.91515      0.224515        3.08485     0.0567094   search
    5            2.88      0.258619           3.12      0.021591   search
    7         2.83029      0.306295        3.16971    -0.0281093   search
```

Fase1: ricerca dell'intervallo

intervallo

```
Search for a zero in the interval [2.83029, 3.16971]:
 Func-count    x          f(x)             Procedure
    7         3.16971     -0.0281093        initial
    8         3.14118    0.000417192        interpolation
    9         3.14159   -5.41432e-08        interpolation
   10         3.14159    1.45473e-15        interpolation
   11         3.14159    1.22465e-16        interpolation
   12         3.14159    1.22465e-16        interpolation
```

Fase2: ricerca zero nell'intervallo individuato

`options` — Options for solution process
structure, typically created using `optimset`

Options for solution process, specified as a structure. Create or modify the `options` structure using `optimset`. `fzero` uses these `options` structure fields.

```
>> X=fzero(@sin,3,optimset('disp','iter'))

Search for an interval around 3 containing a sign change:
 Func-count    a          f(a)             b         f(b)        Procedure
    1            3       0.14112            3       0.14112      initial interval
    3      2.91515      0.224515      3.08485     0.0567094      search
    5         2.88      0.258619         3.12      0.021591      search
    7      2.83029      0.306295      3.16971    -0.0281093      search
```

Fase1: ricerca dell'intervallo

intervallo

```
Search for a zero in the interval [2.83029, 3.16971]:
 Func-count    x          f(x)             Procedure
    7       3.16971     -0.0281093        initial
    8       3.14118     0.000417192       interpolation
    9       3.14159    -5.41432e-08       interpolation
   10       3.14159     1.45473e-15       interpolation
   11       3.14159     1.22465e-16       interpolation
   12       3.14159     1.22465e-16       interpolation
```

Fase2: ricerca zero nell'intervallo individuato

Zero della funzione «prossimo» al punto inziale x0=3

Esempio 1: una approssimazione di pi si può realizzare attraverso la ricerca dello zero della funzione seno in prossimità di 3

Punto iniziale x0=3

```
>> format long
>> x=fzero(@sin,3)
x =
   3.141592653589793
```

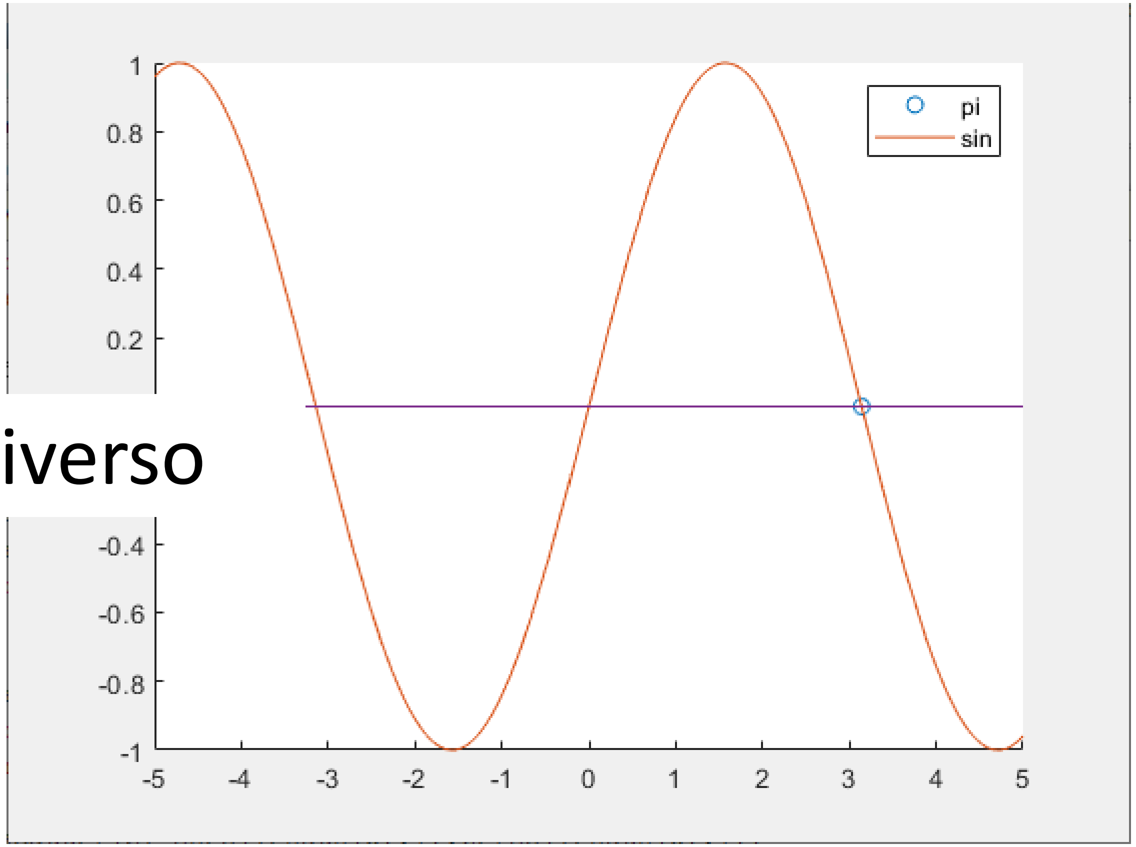
Approssimazione pi-greco

Esempio 1: una approssimazione di pi si può realizzare attraverso la ricerca dello zero della funzione seno in prossimità di 3

```
>> format long
>> x=fzero(@sin,3)
x =
   3.141592653589793
```

Punto iniziale x0=3

Approssimazione pi-greco



Scegliamo un x0 diverso

`options` — Options for solution process
structure, typically created using `optimset`

Options for solution process, specified as a structure. Create or modify the `options` structure using `optimset`. `fzero` uses these `options` structure fields.

```
>> X=fzero(@sin,-3,optimset('disp','iter'));
```

```
Search for an interval around -3 containing a sign change:
 Func-count    a          f(a)             b             f(b)        Procedure
    1              -3       -0.14112             -3       -0.14112   initial interval
    3        -2.91515      -0.224515       -3.08485     -0.0567094   search
    5           -2.88      -0.258619          -3.12      -0.021591   search
    7        -2.83029      -0.306295       -3.16971      0.0281093   search
```

Fase1: ricerca dell'intervallo

intervallo

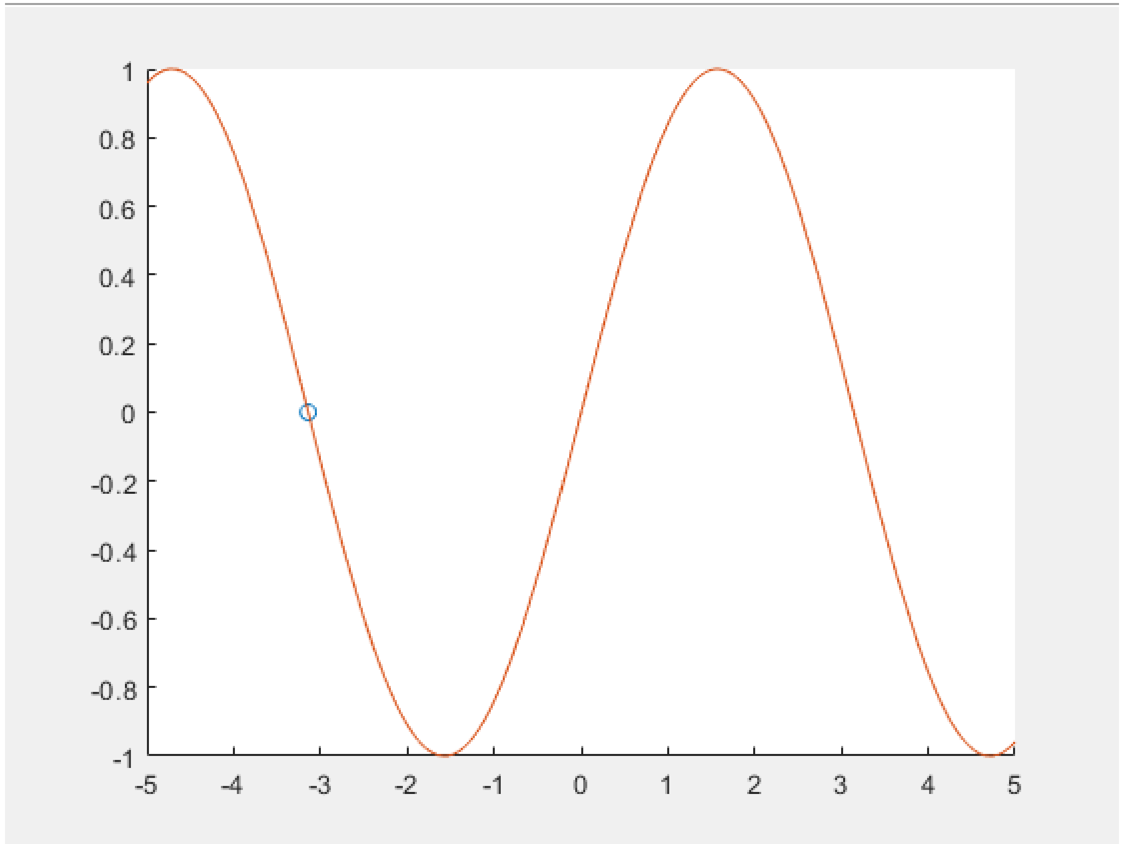
```
Search for a zero in the interval [-2.83029, -3.16971]:
 Func-count    x          f(x)             Procedure
    7        -3.16971      0.0281093        initial
    8        -3.14118   -0.000417192        interpolation
    9        -3.14159    5.41432e-08        interpolation
   10        -3.14159   -1.45473e-15        interpolation
   11        -3.14159   -1.22465e-16        interpolation
   12        -3.14159   -1.22465e-16        interpolation

Zero found in the interval [-2.83029, -3.16971]
```

Fase2: ricerca zero nell'intervallo individuato

Esempio 1: una approssimazione di pi si può realizzare attraverso la ricerca dello zero della funzione seno in prossimità di 3

Punto iniziale x0=-3

```
>> format long
>> x=fzero(@sin,-3)
x =
  - 3.141592653589793
```



Esempio: funzione y=f(x) con x0=2

```
function y = f( x )
%function per test
y=x.^2-0.25;
end
```

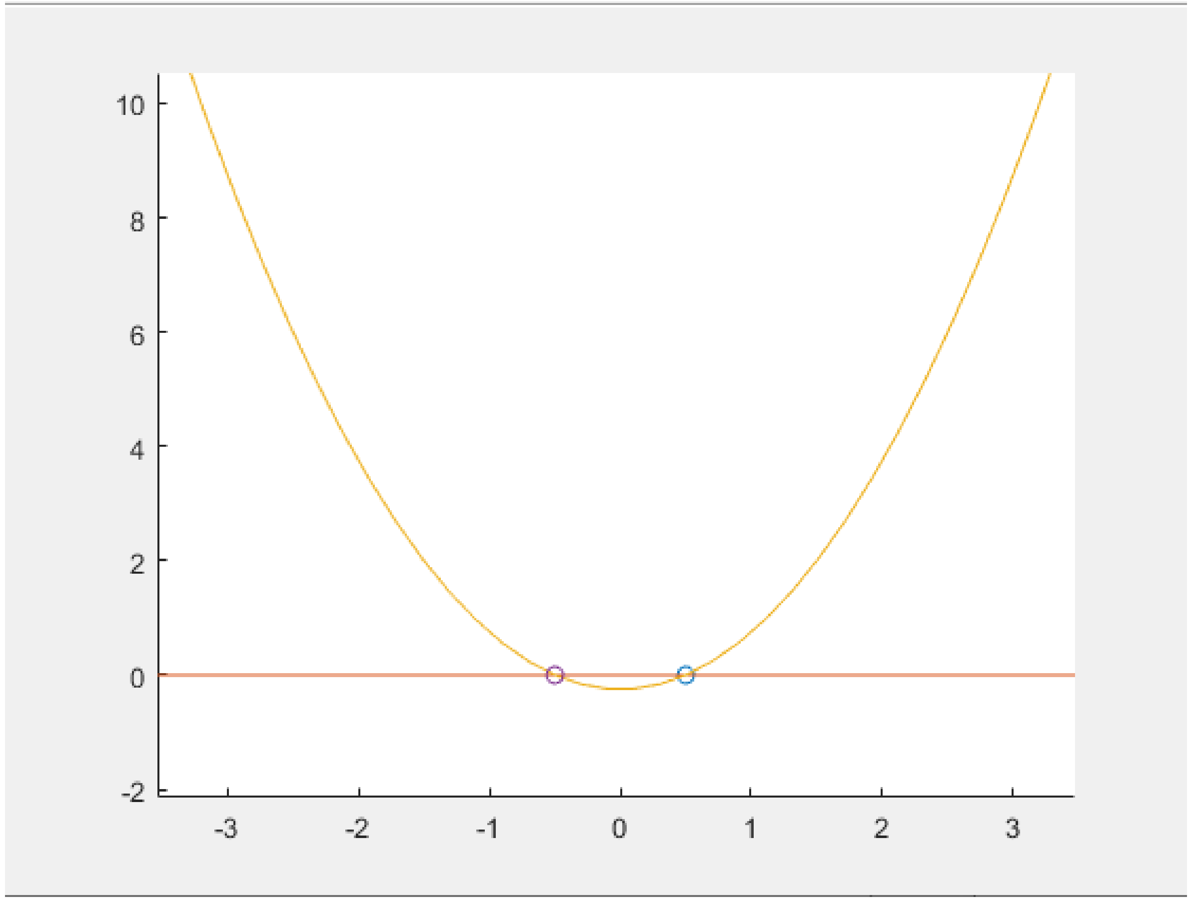
```
>> [x,fval,exitflag,output]=fzero(@f,2)
x =
    5.000000000000000e-01
fval =
    0
exitflag =
    1
output =
  struct with fields:

    intervaliterations: 11
            iterations: 8
             funcCount: 30
             algorithm: 'bisection, interpolation'
               message: 'Zero found in the interval [0.189807, 3.28]'
```

Esempio: funzione y=f(x) con x0=-2

```
function y = f( x )
%function per test
y=x.^2-0.25;
end
```

```
>> [x,fval,exitflag,output]=fzero(@f,-2)
x =
   -5.000000000000000e-01
fval =
     0
exitflag =
     1

output =
  struct with fields:
    intervaliterations: 11
            iterations: 8
             funcCount: 30
             algorithm: 'bisection, interpolation'
               message: 'Zero found in the interval [-0.189807, -3.28]'
```

Esercizio: grafico funzione f(x)=x^2-0.25;



Esempio consideriamo la seguente funzione

```
function y = f( x )
%function per test
y=x.^2-0.25;
end
```

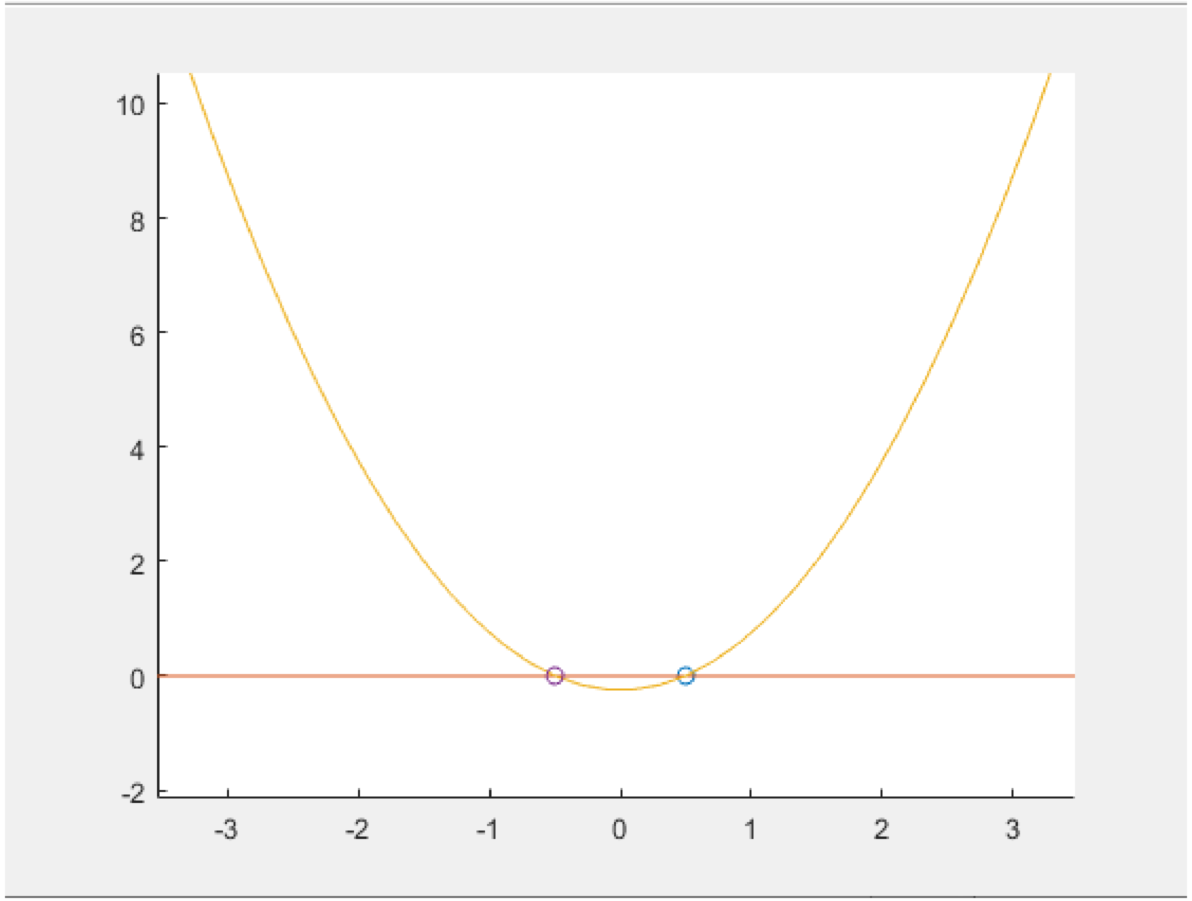
Possibilità di dare in input un intervallo

```
>> [x,fval,exitflag,output]=fzero(@f,[1 2])
```

Esempio consideriamo la seguente funzione

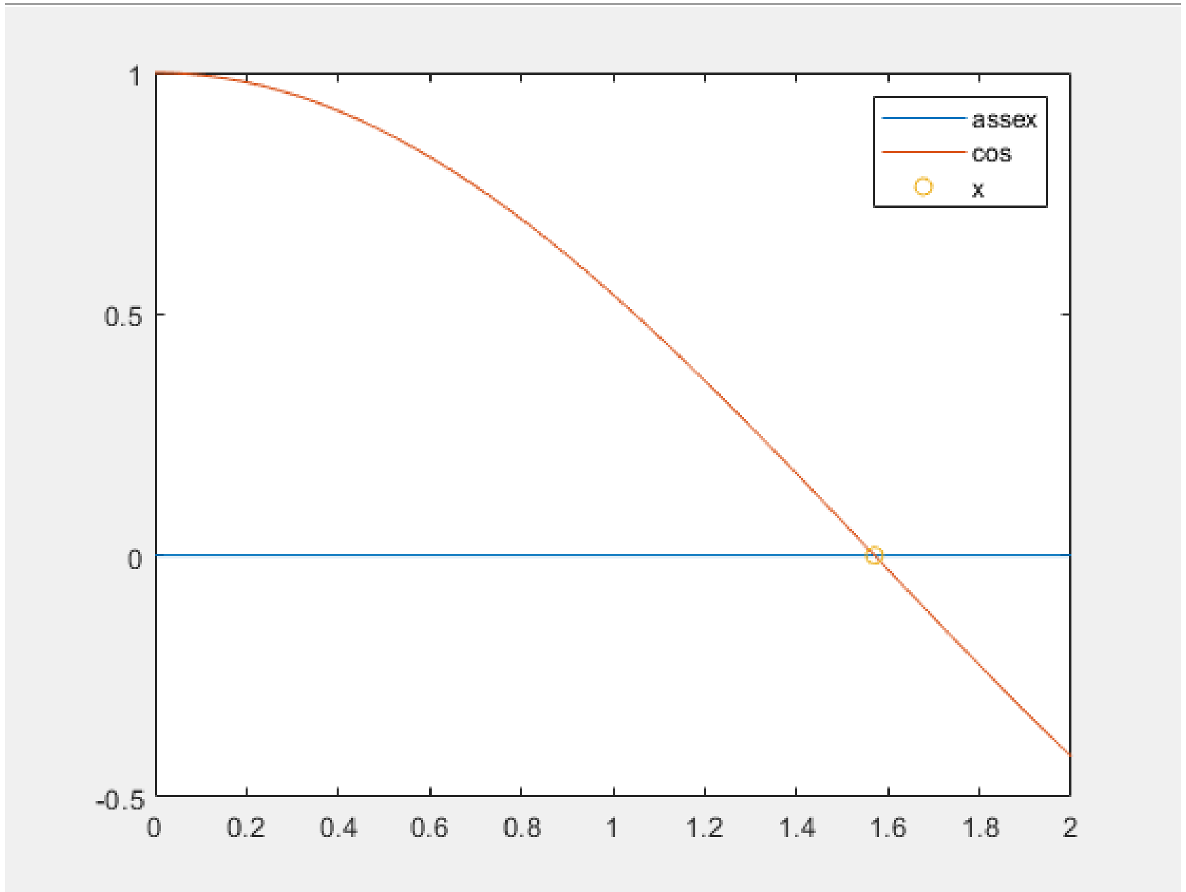
```
function y = f( x )
%function per test
y=x.^2-0.25;
end
```

```
>> [x,fval,exitflag,output]=fzero(@f,[1 2])
??? Error using ==> fzero at 290
The function values at the interval endpoints
must differ in sign.
```



Esempio 2:trovare lo zero della funzione coseno compreso tra 1 e 2:

```
>> x=fzero(@cos,2)

x =

   1.570796326794897
```



```
>> [x,fval,exitflag,output]=fzero(@cos,2)
```

exitflag — Integer encoding the exit condition

Integer

Integer encoding the exit condition, meaning the reason `fsolve` stopped its iterations.

`1`	Function converged to a solution `x`.
`-1`	Algorithm was terminated by the output function or plot function.
`-3`	`NaN` or `Inf` function value was encountered while searching for an interval containing a sign change
`-4`	Complex function value was encountered while searching for an interval containing a sign change.
`-5`	Algorithm might have converged to a singular point.
`-6`	`fzero` did not detect a sign change.

```
>> [x,fval,exitflag,output]=fzero(@cos,2)
```

`output` — Information about root-finding process

Structure

Information about root-finding process, returned as a structure. The fields of the structure are:

`intervaliterations`	Number of iterations taken to find an interval containing a root
`iterations`	Number of zero-finding iterations
`funcCount`	Number of function evaluations
`algorithm`	`'bisection, interpolation'`
`message`	Exit message

```
>>  [x,fval,exitflag,output]=fzero(@cos,2)
x =
 1.570796326794897
fval =
    6.123233995736766e-17
exitflag =
    1
```

converge

exitflag — Integer encoding the exit condition

Integer

1	Function converged to a solution x.

```
output =
  struct with fields:
    intervaliterations: 7
            iterations: 5
             funcCount: 19
             algorithm: 'bisection, interpolation'
               message: 'Zero found in the interval [1.54745, 2.32]'
```

intervaliterations: Iterazioni per la ricerca dell'intervallo

iterations: Iterazioni per la ricerca dello zero

options — Options for solution process
structure, typically created using `optimset`

Options for solution process, specified as a structure. Create or modify the `options` structure using `optimset`. `fzero` uses these `options` structure fields.

`TolX`	Termination tolerance on `x`, a positive scalar. The default is `eps`, 2.2204e–16.

```
[x,fval]=fzero(@cos,2)
x =
   1.570796326794897e+000
fval =
   6.123233995736766e-017

options = optimset('TolX',1e-4);
[x,fval]=fzero(@cos,2,options)
x =
   1.570664378820111e+000
fval =
   1.319479744028485e-004
```

Un’altra opzione: l’utente può scegliere la Tolleranza.

Limitazioni fzero matlab (funzione con punti di discontinuità)

Esempio: funzione tan considerando un punto iniziale x0

```
>> x=fzero(@tan,0.5,optimset('disp','iter'));

Search for an interval around 0.5 containing a sign change:
 Func-count    a          f(a)             b             f(b)        Procedure
    1              0.5       0.546302           0.5       0.546302   initial interval
    3         0.485858       0.528079      0.514142        0.56481   search
    5             0.48       0.520611          0.52       0.572562   search
    7         0.471716       0.510126      0.528284       0.583615   search
    9             0.46       0.495449          0.54        0.59943   search
   11         0.443431       0.474979      0.556569        0.62218   search
   13             0.42       0.446573          0.58       0.655168   search
   15         0.386863       0.407392      0.613137       0.703599   search
   17             0.34       0.353737          0.66       0.776105   search
   19         0.273726       0.280774      0.726274        0.88823   search
   21             0.18        0.18197          0.82        1.07171   search
   23        0.0474517      0.0474873      0.952548        1.40594   search
   24            -0.14      -0.140922      0.952548        1.40594   search

Zero found in the interval [-0.14, 0.952548]
>> x

x =

   -1.633179678712147e-23
```

Limitazioni fzero matlab (funzione con punti di discontinuità)

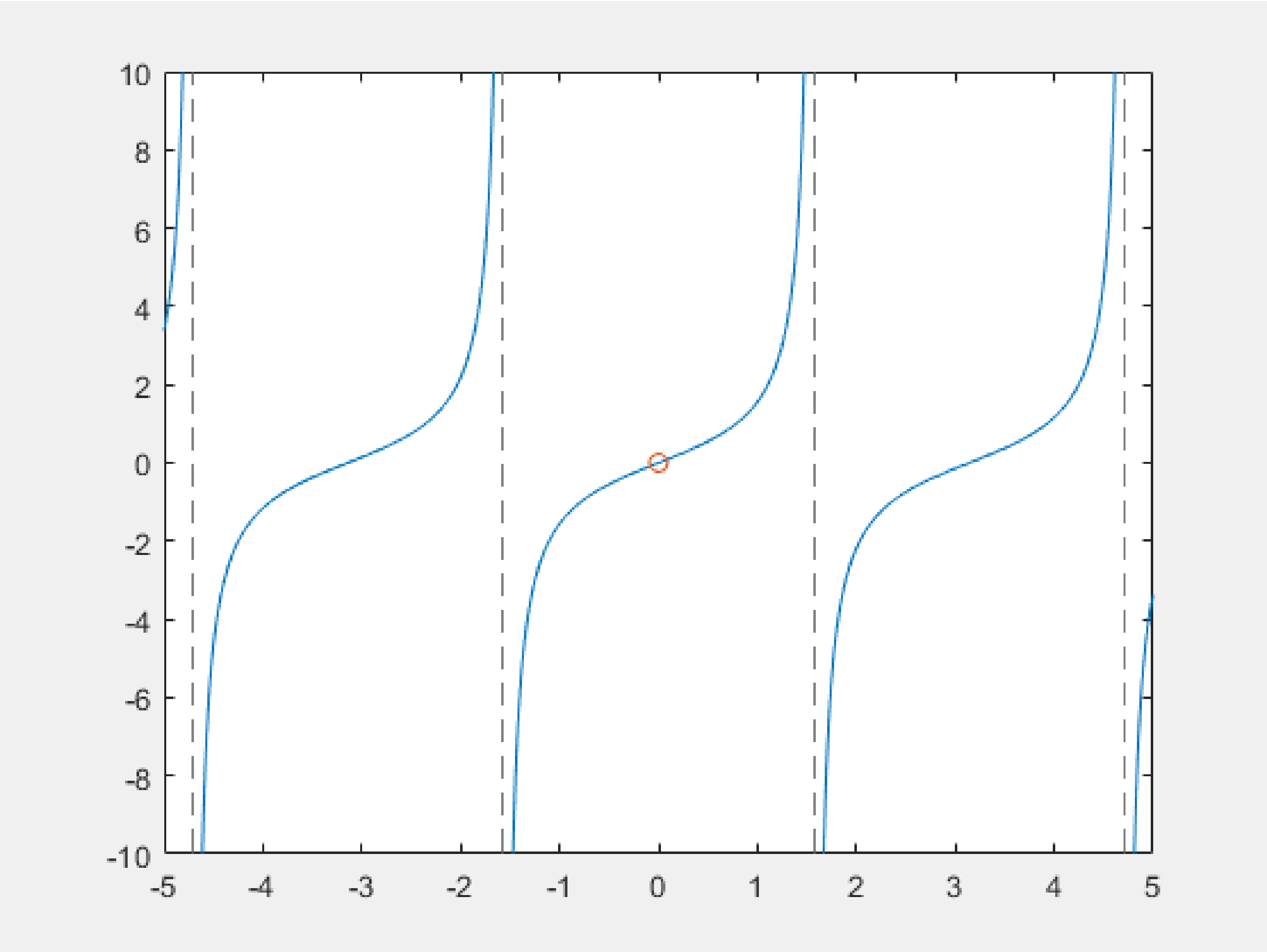
Esempio: funzione tan considerando un punto iniziale x0

```
>> x=fzero(@tan,0.5,optimset('disp','iter'));

Search for an interval around 0.5 containing a sign change:
 Func-count    a          f(a)             b             f(b)        Procedure
    1             0.5      0.546302           0.5      0.546302    initial interval
    3        0.485858      0.528079      0.514142       0.56481    search
    5            0.48      0.520611          0.52      0.572562    search
    7        0.471716      0.510126      0.528284      0.583615    search
    9            0.46      0.495449          0.54       0.59943    search
   11        0.443431      0.474979      0.556569       0.62218    search
   13            0.42      0.446573          0.58      0.655168    search
   15        0.386863      0.407392      0.613137      0.703599    search
   17            0.34      0.353737          0.66      0.776105    search
   19        0.273726      0.280774      0.726274       0.88823    search
   21            0.18       0.18197          0.82       1.07171    search
   23       0.0474517     0.0474873      0.952548       1.40594    search
   24           -0.14     -0.140922      0.952548       1.40594    search

Zero found in the interval [-0.14, 0.952548]
>> x

x =

   -1.633179678712147e-23
```

Intervallo in cui la funzione è continua

Limitazioni fzero matlab

Esempio: funzione tan considerando un punto iniziale x0 e un intervallo



Esempio: consideriamo la funzione tan e come punto iniziale 1 «vicino» ad un punto di discontinuità

```
>> x=fzero(@tan,1,optimset('disp','iter'));

Search for an interval around 1 containing a sign change:
 Func-count    a          f(a)             b          f(b)        Procedure
    1              1       1.55741              1       1.55741   initial interval
    3       0.971716       1.46458        1.02828       1.65879   search
    5           0.96       1.42836           1.04       1.70361   search
    7       0.943431       1.37915        1.05657       1.77015   search
    9           0.92       1.31326           1.08       1.87122   search
   11       0.886863       1.22671        1.11314       2.03031   search
   13           0.84       1.11563           1.16        2.2958   search
   15       0.773726      0.976924        1.22627       2.78681   search
   17           0.68      0.808661           1.32       3.90335   search
   19       0.547452      0.609604        1.45255       8.41735   search
   21           0.36      0.376403           1.64       -14.427   search
```

```
Search for a zero in the interval [0.36, 1.64]:

 Func-count    x          f(x)             Procedure
   62          1.5708  -3.16043e+09         interpolation
   63          1.5708  -7.03991e+09         bisection
   64          1.5708  -1.82267e+10         bisection
   65          1.5708   3.09423e+10         bisection
   66          1.5708   4.75172e+10         interpolation
   67          1.5708  -8.87064e+10         bisection
   68          1.5708  -1.56561e+11         interpolation
   69          1.5708   2.04669e+11         bisection
   70          1.5708   2.41826e+11         interpolation
   71          1.5708   5.90925e+11         bisection
   72          1.5708  -1.33214e+12         bisection
   73          1.5708   2.12406e+12         interpolation
   74          1.5708  -3.57223e+12         interpolation
   75          1.5708   5.24116e+12         interpolation
   76          1.5708   9.84876e+12         interpolation
   77          1.5708  -1.12107e+13         bisection
   78          1.5708  -1.20506e+13         interpolation
   79          1.5708  -2.60739e+13         bisection
   80          1.5708  -6.27905e+13         bisection
   81          1.5708   1.59274e+14         bisection
   82          1.5708  -2.07308e+14         interpolation
   83          1.5708  -2.86411e+14         interpolation
   84          1.5708   7.17618e+14         bisection
```

Intervallo individuato

Circa pi/2

```
Search for a zero in the interval [0.36, 1.64]:

Func-count    x          f(x)             Procedure
    62        1.5708   -3.16043e+09        interpolation
    63        1.5708   -7.03991e+09        bisection
    64        1.5708   -1.82267e+10        bisection
    65        1.5708    3.09423e+10        bisection
    66        1.5708    4.75172e+10        interpolation
    67        1.5708   -8.87064e+10        bisection
    68        1.5708   -1.56561e+11        interpolation
    69        1.5708    2.04669e+11        bisection
    70        1.5708    2.41826e+11        interpolation
    71        1.5708    5.90925e+11        bisection
    72        1.5708   -1.33214e+12        bisection
    73        1.5708    2.12406e+12        interpolation
    74        1.5708   -3.57223e+12        interpolation
    75        1.5708    5.24116e+12        interpolation
    76        1.5708    9.84876e+12        interpolation
    77        1.5708   -1.12107e+13        bisection
    78        1.5708   -1.20506e+13        interpolation
    79        1.5708   -2.60739e+13        bisection
    80        1.5708   -6.27905e+13        bisection
    81        1.5708    1.59274e+14        bisection
    82        1.5708   -2.07308e+14        interpolation
    83        1.5708   -2.86411e+14        interpolation
    84        1.5708    7.17618e+14        bisection
```

Intervallo individuato

Ad ogni iterazione effettua valutazioni della funzione

```
Search for a zero in the interval [0.36, 1.64]:

Func-count    x          f(x)             Procedure
   62       1.5708   -3.16043e+09        interpolation
   63       1.5708   -7.03991e+09        bisection
   64       1.5708   -1.82267e+10        bisection
   65       1.5708    3.09423e+10        bisection
   66       1.5708    4.75172e+10        interpolation
   67       1.5708   -8.87064e+10        bisection
   68       1.5708   -1.56561e+11        interpolation
   69       1.5708    2.04669e+11        bisection
   70       1.5708    2.41826e+11        interpolation
   71       1.5708    5.90925e+11        bisection
   72
   73
   74
   75
   76
   77       1.5708   -1.12107e+13        bisection
   78       1.5708   -1.20506e+13        interpolation
   79       1.5708   -2.60739e+13        bisection
   80       1.5708   -6.27905e+13        bisection
   81       1.5708    1.59274e+14        bisection
   82       1.5708   -2.07308e+14        interpolation
   83       1.5708   -2.86411e+14        interpolation
   84       1.5708    7.17618e+14        bisection
```

Intervallo individuato

Ad ogni iterazione effettua valutazioni della funzione e si accorge della presenza di un punto di Discontinuità

```
Current point x may be near a singular point. The interval [0.36, 1.64]
reduced to the requested tolerance and the function changes sign in the interval,
but f(x) increased in magnitude as the interval reduced.
```

Oppure

Limitazioni fzero matlab

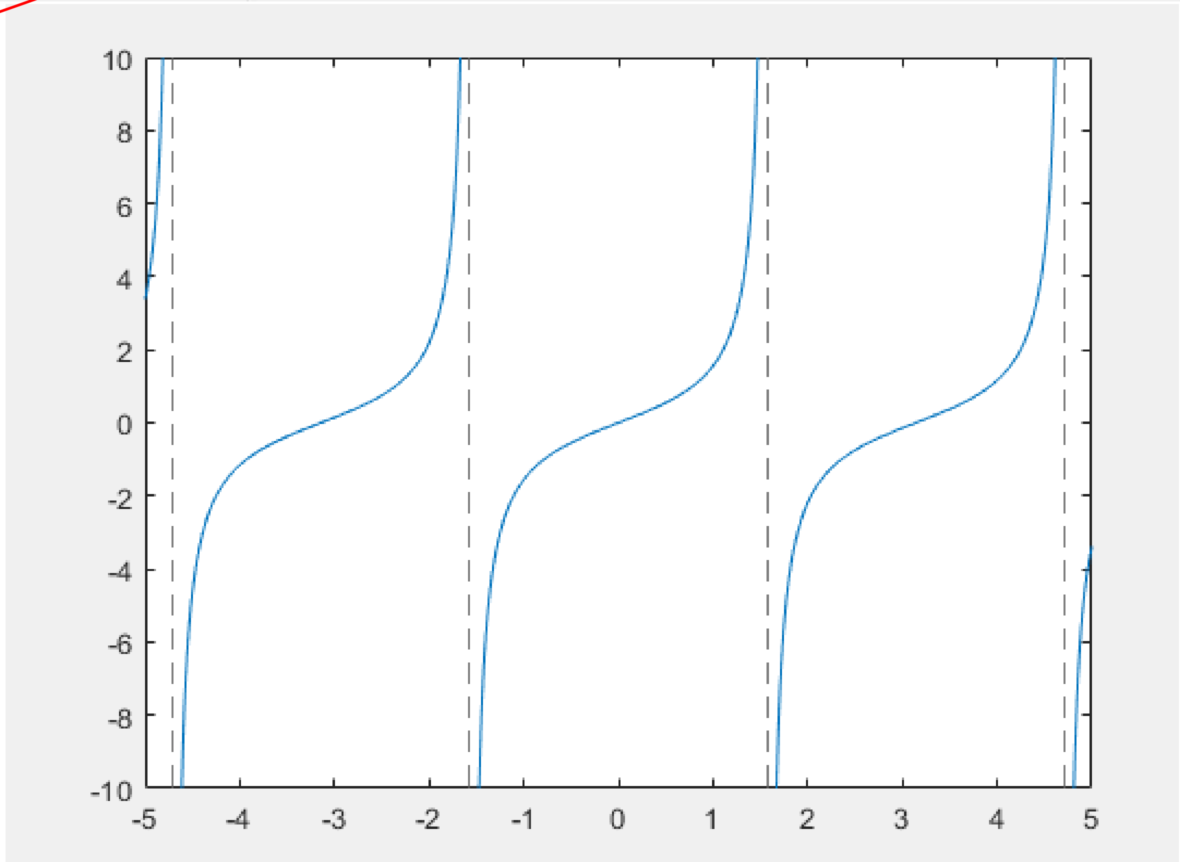
Esempio: funzione tan considerando un punto iniziale x0 e un intervallo

```
>> [x,~,exitflag]=fzero(@tan,1)

x =

   1.570796326794898


exitflag =

   -5

>>
```

exitflag — Integer encoding the exit condition

Integer

-5	Algorithm might have converged to a singular point.



Limitazioni fzero matlab

Esempio: funzione tan considerando un punto iniziale x0 e un intervallo

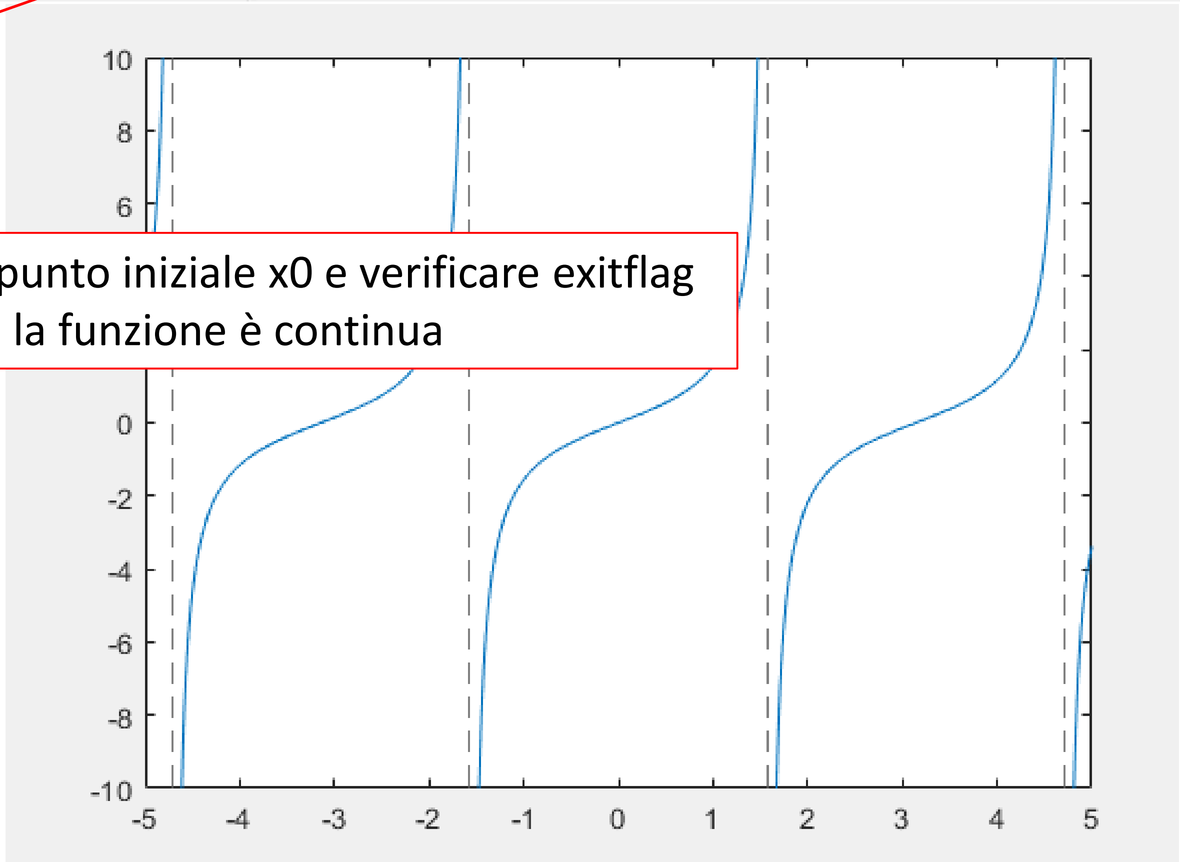
```
>> [x,~,exitflag]=fzero(@tan,1)

x =

   1.570796326794898


exitflag =

  -5

>>
```

exitflag — Integer encoding the exit condition

Integer

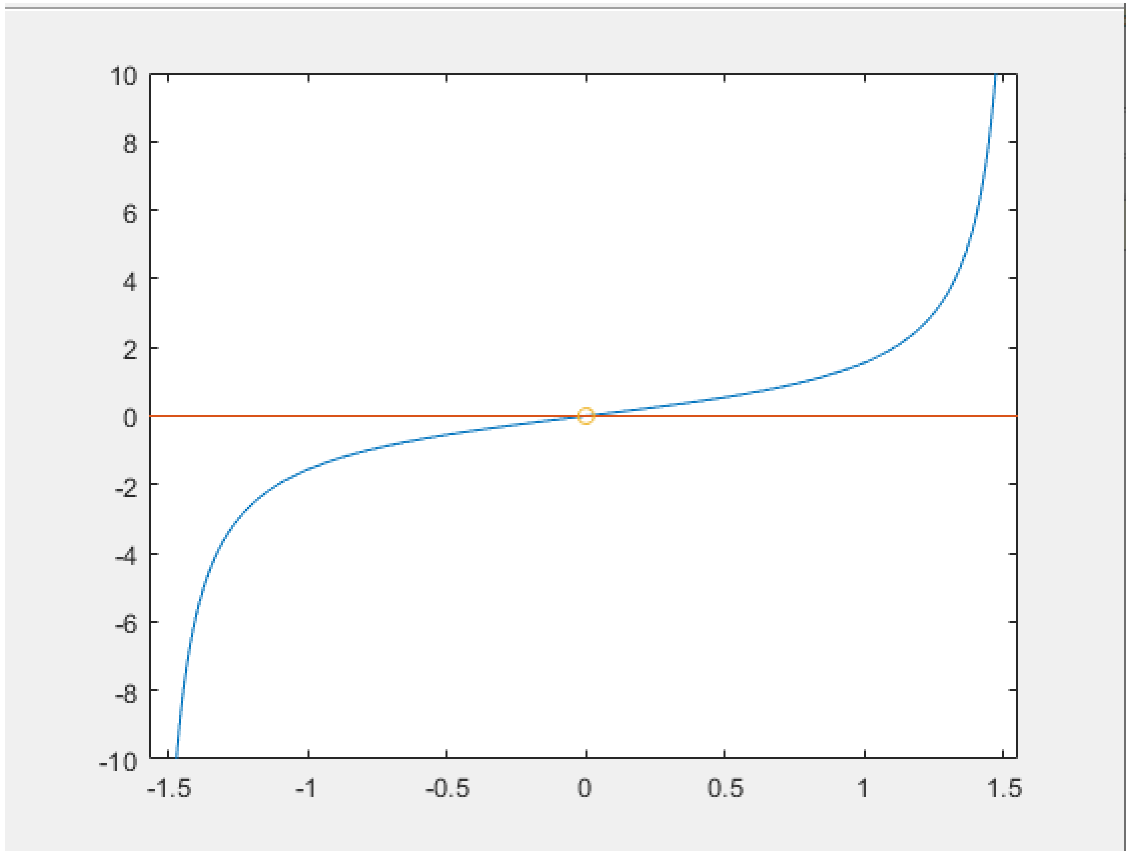
-5	Algorithm might have converged to a singular point.

Come procedere? Cambiare il punto iniziale x0 e verificare exitflag o dare in input intervallo in cui la funzione è continua



Esempio: funzione tan considerando un intervallo

```
>> [x,~,exitflag]=fzero(@tan,[0,1])

x =

     0


exitflag =

     1
```



Esempio: funzione f(x)=1/x

```
>> x=fzero(@(x) 1/x,5,optimset('disp','iter'));

Search for an interval around 5 containing a sign change:
 Func-count    a          f(a)             b             f(b)        Procedure
    1               5          0.2             5             0.2        initial interval
    3         4.85858     0.205822       5.14142       0.194499        search
    5             4.8     0.208333           5.2       0.192308        search
    7         4.71716     0.211992       5.28284       0.189292        search
    9             4.6     0.217391           5.4       0.185185        search
   11         4.43431     0.225514       5.56569       0.179672        search
   13             4.2     0.238095           5.8       0.172414        search
   15         3.86863     0.258489       6.13137       0.163096        search
   17             3.4     0.294118           6.6       0.151515        search
   19         2.73726     0.365329       7.26274       0.137689        search
   21             1.8     0.555556           8.2       0.121951        search
   23        0.474517      2.10741       9.52548       0.104982        search
   24            -1.4    -0.714286       9.52548       0.104982        search

Search for a zero in the interval [-1.4, 9.52548]:
```

Esempio: funzione f(x)=1/x

Func-count	x	f(x)	Procedure
72	5.14563e-11	1.9434e+10	interpolation
73	-5.13592e-11	-1.94707e+10	interpolation
74	-5.12621e-11	-1.95076e+10	interpolation
75	-2.55825e-11	-3.90892e+10	bisection
76	-1.27427e-11	-7.84763e+10	bisection
77	-6.3228e-12	-1.58158e+11	bisection
78	-3.11285e-12	-3.21249e+11	bisection
79	-1.50788e-12	-6.63183e+11	bisection
80	-7.05392e-13	-1.41765e+12	bisection
81	-3.04149e-13	-3.28787e+12	bisection
82	-1.03527e-13	-9.65934e+12	bisection
83	9.7095e-14	1.02992e+13	bisection
84	9.38791e-14	1.0652e+13	interpolation
85	4.53316e-14	2.20597e+13	bisection
86	2.10579e-14	4.74882e+13	bisection
87	8.92099e-15	1.12095e+14	bisection
88	-3.21588e-15	-3.10957e+14	bisection
89	2.85255e-15	3.50563e+14	interpolation
90	2.40847e-15	4.15202e+14	interpolation
91	1.02257e-15	9.7793e+14	bisection
92	-3.63329e-16	-2.75233e+15	bisection

Ad ogni iterazione effettua valutazioni della funzione

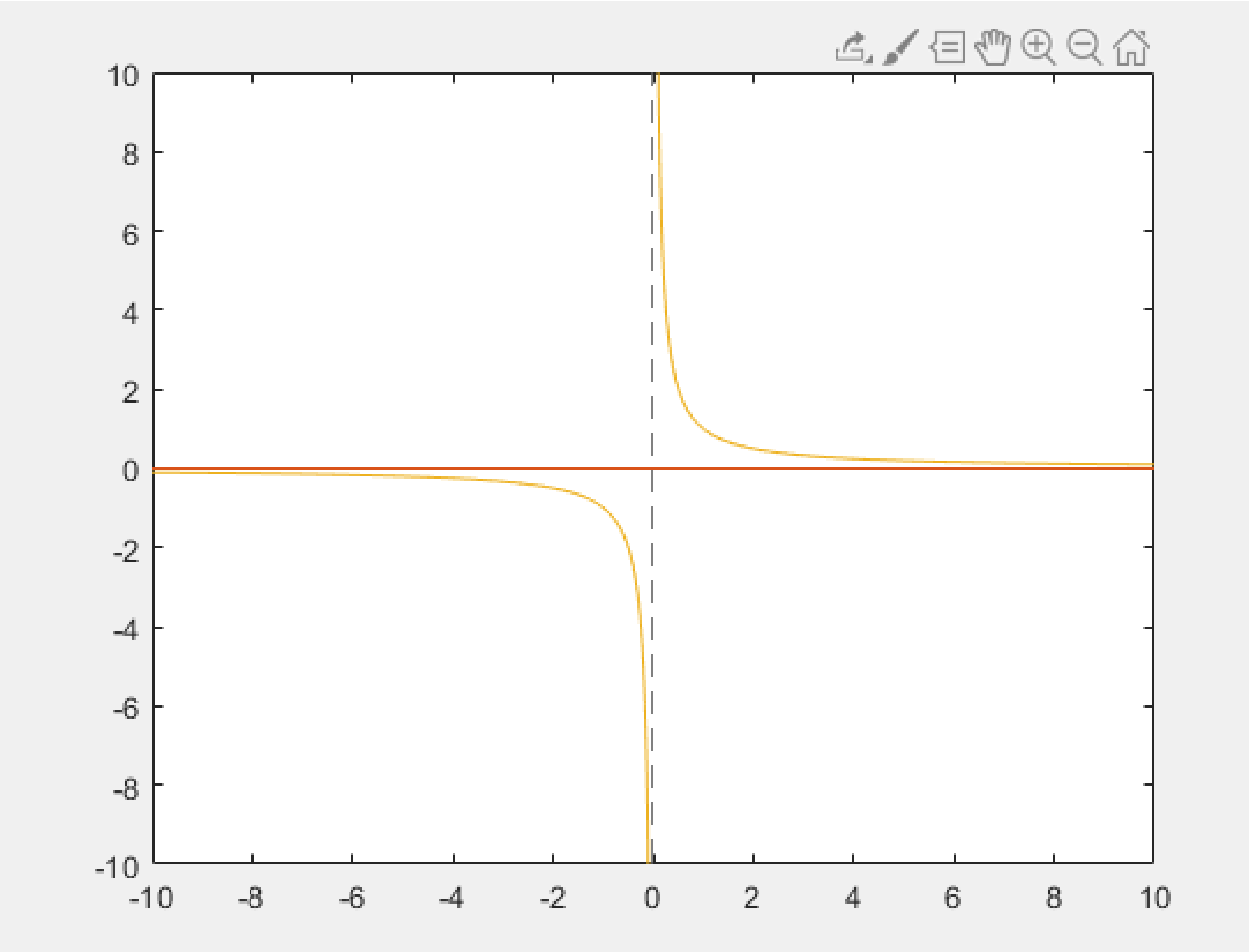
Esempio: funzione f(x)=1/x

Func-count	x	f(x)	Procedure
72	5.14563e-11	1.9434e+10	interpolation
73	-5.13592e-11	-1.94707e+10	interpolation
74	-5.12621e-11	-1.95076e+10	interpolation
75	-2.55825e-11	-3.90892e+10	bisection
76	-1.27427e-11	-7.84763e+10	bisection
77	-6.3228e-12	-1.58158e+11	bisection
78	-3.11285e-12	-3.21249e+11	bisection
79	-1.50788e-12	-6.63183e+11	bisection
80	-7.05392e-13	-1.41765e+12	bisection
81	-3.0[illegible]	[illegible]	[illegible]
82	-1.0[illegible]	[illegible]	[illegible]
83	9.7[illegible]	[illegible]	[illegible]
84	9.3[illegible]	[illegible]	[illegible]
85	4.5[illegible]	[illegible]	[illegible]
86	2.10579e-14	4.74882e+13	bisection
87	8.92099e-15	1.12095e+14	bisection
88	-3.21588e-15	-3.10957e+14	bisection
89	2.85255e-15	3.50563e+14	interpolation
90	2.40847e-15	4.15202e+14	interpolation
91	1.02257e-15	9.7793e+14	bisection
92	-3.63329e-16	-2.75233e+15	bisection

Ad ogni iterazione effettua valutazioni
della funzione e si accorge della presenza di un punto di
Discontinuità

```
Current point x may be near a singular point. The interval [0.36, 1.64]
reduced to the requested tolerance and the function changes sign in the interval,
but f(x) increased in magnitude as the interval reduced.
```

Esempio: grafico funzione 1/x



Esempio: consideriamo la seguente funzione

f(x)=x^8-10^-8

Ricerca degli zeri utilizzando fzero:

```
>> [x,fval,exitflag,output]=fzero(@(x) x.^8-10^-8,1)
x =
   0.100000000000000
fval =
     3.308722450212111e-24

exitflag =


     1
output =

  struct with fields:

    intervaliterations: 11
            iterations: 8
             funcCount: 30
             algorithm: 'bisection, interpolation'
               message: 'Zero found in the interval [0.0949033, 1.64]'
```

Iterazioni per individuare l'intervallo

Iterazioni per individuare lo zero della fun

Intervallo

Esempio: consideriamo la seguente funzione

f(x)=x^8-10^-8

Ricerca degli zeri utilizzando fzero:

```
>> [x,fval,exitflag,output]=fzero(@(x) x.^8-10^-8,1)
x =
   0.100000000000000
fval =
     3.308722450212111e-24

exitflag =

     1
output =

  struct with fields:

    intervaliterations: 11
            iterations: 8
             funcCount: 30
             algorithm: 'bisection, interpolation'
               message: 'Zero found in the interval [0.0949033, 1.64]'
```

Calcoliamo la derivata della funzione

```
>> Df=8*x^7

Df =

     8.000000000000003e-07
```

Iterazioni per individuare l’intervallo

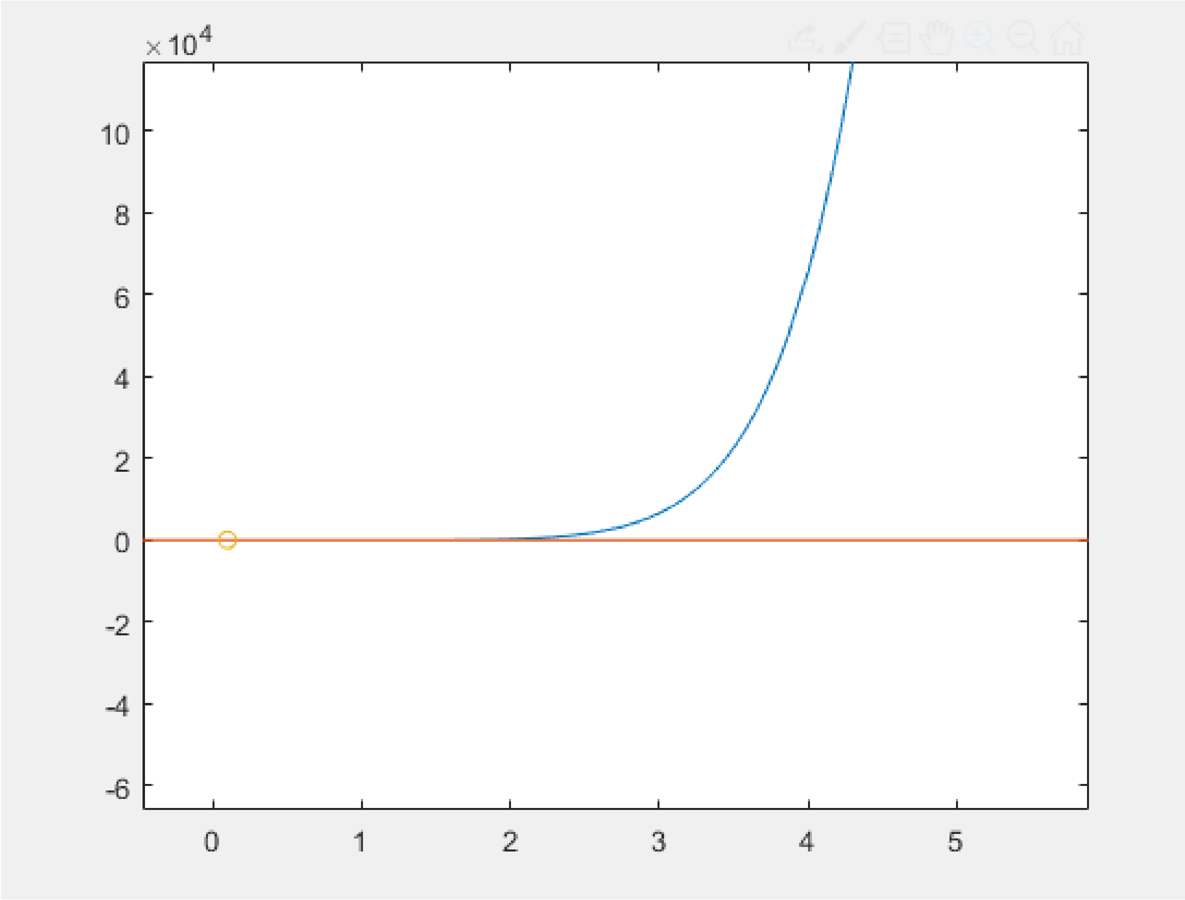
Iterazioni per individuare lo zero della fun

Intervallo

Esempio: consideriamo la seguente funzione

$$f(x)=x^8-10^{-8}$$

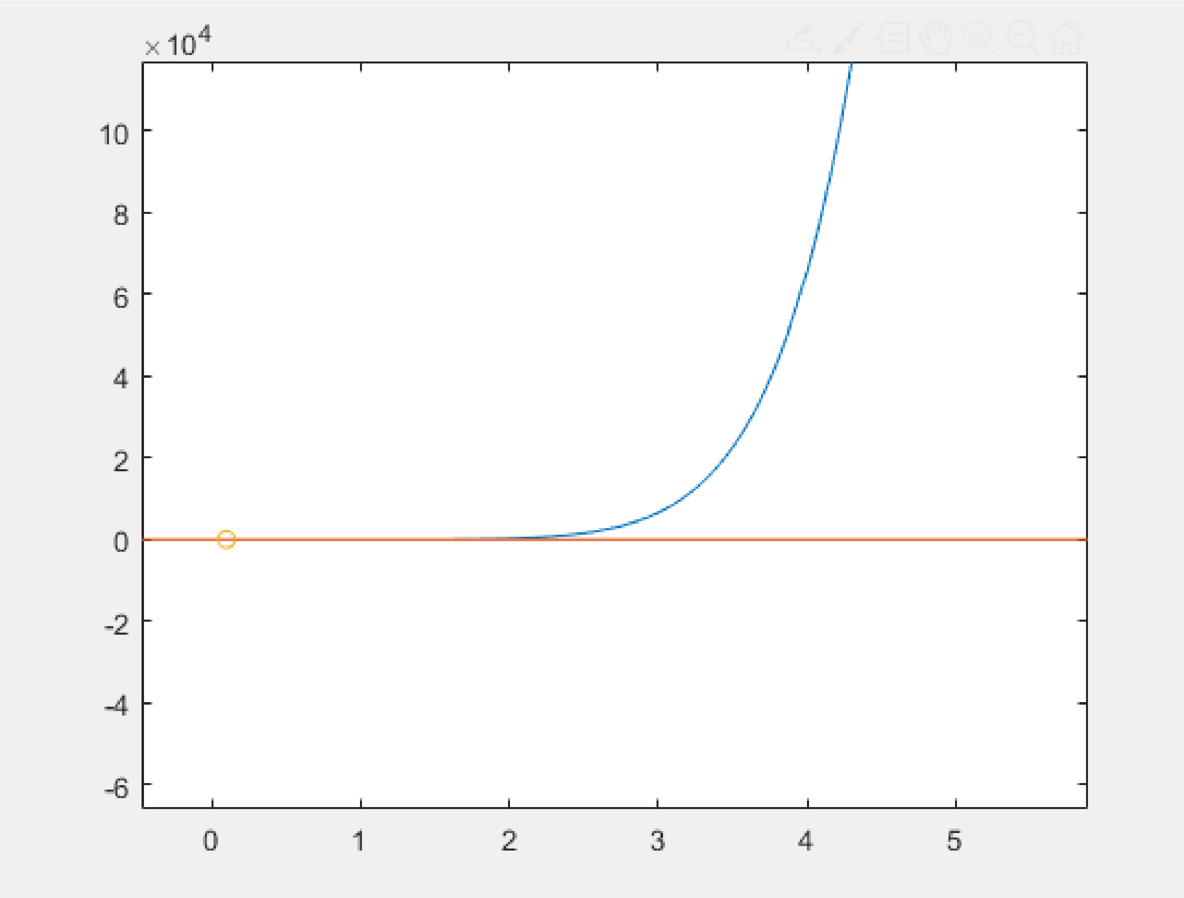
Ricerca degli zeri utilizzando fzero:



Esempio: consideriamo la seguente funzione

f(x)=x^8-10^-8

Ricerca degli zeri utilizzando fzero:



In particolare

σ<=μδ dove

μ=1/|f'(x*)|

Nel nostro caso
μ=1.250000000000000e+06

Infatti perturbando di «poco»
I dati iniziali →

Esempio: consideriamo la seguente funzione

$$f(x)=x^{8}-10^{-8} \longrightarrow f_t(x)=x^{8}-7*10^{-8}$$

Ricerca degli zeri utilizzando fzero:

```
>> [x_t,~,exitflag]=fzero(@(x) x.^8-7*10^-8,1)

x_t =
    1.275373106858454e-01


exitflag =

    1


>> abs(x-x_t)

ans =

    2.753731068584539e-02
```

In particolare

$\sigma<=\mu\delta$ dove

$\mu=1/|f'(x^*)|$

Nel nostro caso
$\mu=1.250000000000000e+06$

Esempio: consideriamo la seguente funzione

f(x)=x^8-10^-8 → f_t(x)=x^8-7*10^-8

Ricerca degli zeri utilizzando fzero:

```
>> [x_t,~,exitflag]=fzero(@(x) x.^8-7*10^-8,1)

x_t =
    1.275373106858454e-01


exitflag =

    1


>> abs(x-x_t)

ans =

    2.753731068584539e-02
```

In particolare

$\sigma <= \mu\delta$ dove

$\mu=1/|f'(x^*)|$

Nel nostro caso
μ=1.250000000000000e+06

Un errore sui dati dell'ordine di 10^-8
ha portato ad un errore sui dati dell'ordine di 10^-2

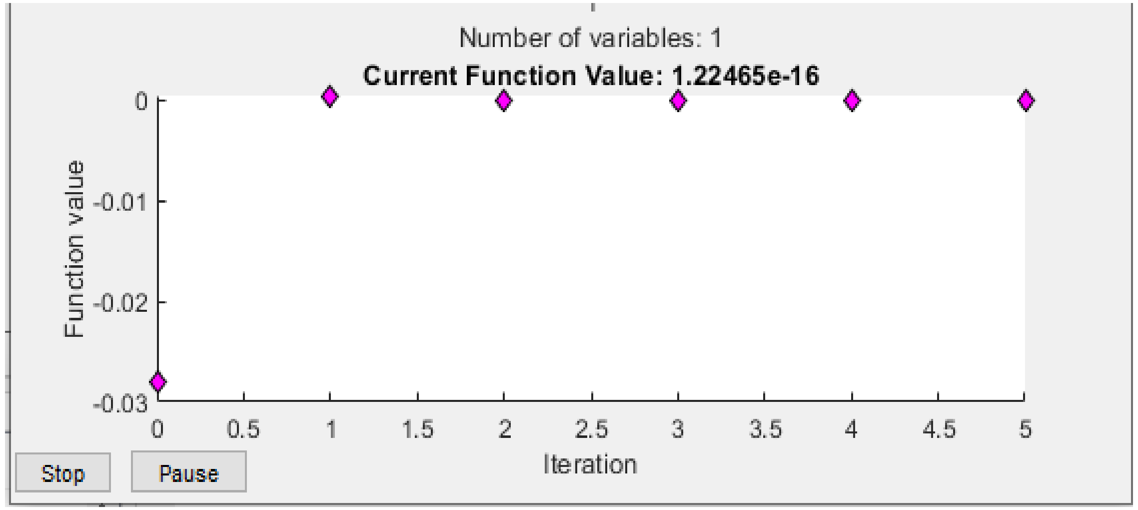
PlotFcns	Plot various measures of progress while the algorithm executes. Select from predefined plots or write your own. Pass a function handle or a cell array of function handles. The default is none ([]). • `@optimplotx` plots the current point. • `@optimplotfval` plots the function value. For information on writing a custom plot function, see Plot Functions.

```
>>options=optimset('PlotFcns',{@optimplotfval});
>> X=fzero(@sin,3,options)

X =

   3.141592653589793
```

PlotFcns	Plot various measures of progress while the algorithm executes. Select from predefined plots or write your own. Pass a function handle or a cell array of function handles. The default is none ([ ]). • `@optimplotx` plots the current point. • `@optimplotfval` plots the function value. For information on writing a custom plot function, see Plot Functions.

```
>>options=optimset('PlotFcns',{@optimplotfval});
>> X=fzero(@sin,3,options)

X =

   3.141592653589793
```

Valori della funzione ad ogni iterazione



Esempio: individuare uno zero della funzione

f(x)=x^3-2x-5

```
>> f=@(x) x.^3-2*x-5

f =

  function_handle with value:

    @(x)x.^3-2*x-5

>> z=fzero(f,2)

z =

    2.0946
```

Esempio: individuare uno zero della funzione

f(x)=x^3-2x-5

```
>> f=@(x) x.^3-2*x-5

f =

  function_handle with value:

    @(x)x.^3-2*x-5

>> z=fzero(f,2)

z =

    2.0946
```

Poiché la funzione f è un polinomio, si può, utilizzare, in alternativa, la funzione

```
>> roots([ 1 0 -2 -5])

ans =

   2.0946 + 0.0000i
  -1.0473 + 1.1359i
  -1.0473 - 1.1359i

>>
```

Nota: la funzione roots implementa un algoritmo basato sul
Calcolo degli autovalori di una matrice associata la polinomio
(companion matrix), costruita a partire dal vettore dei suoi coefficienti.

Limitazioni fzero matlab

Consideriamo la seguente funzione:

$$f(x)=x^2$$

```
>> x=fzero(@(x) x^2, 1)
Exiting fzero: aborting search for an interval containing a sign change
    because NaN or Inf function value encountered during search.
(Function value at -1.7162e+154 is Inf.)
Check function or try again with a different starting value.

x =

   NaN
```

Perché la funzione non «attraversa» l'asse delle x

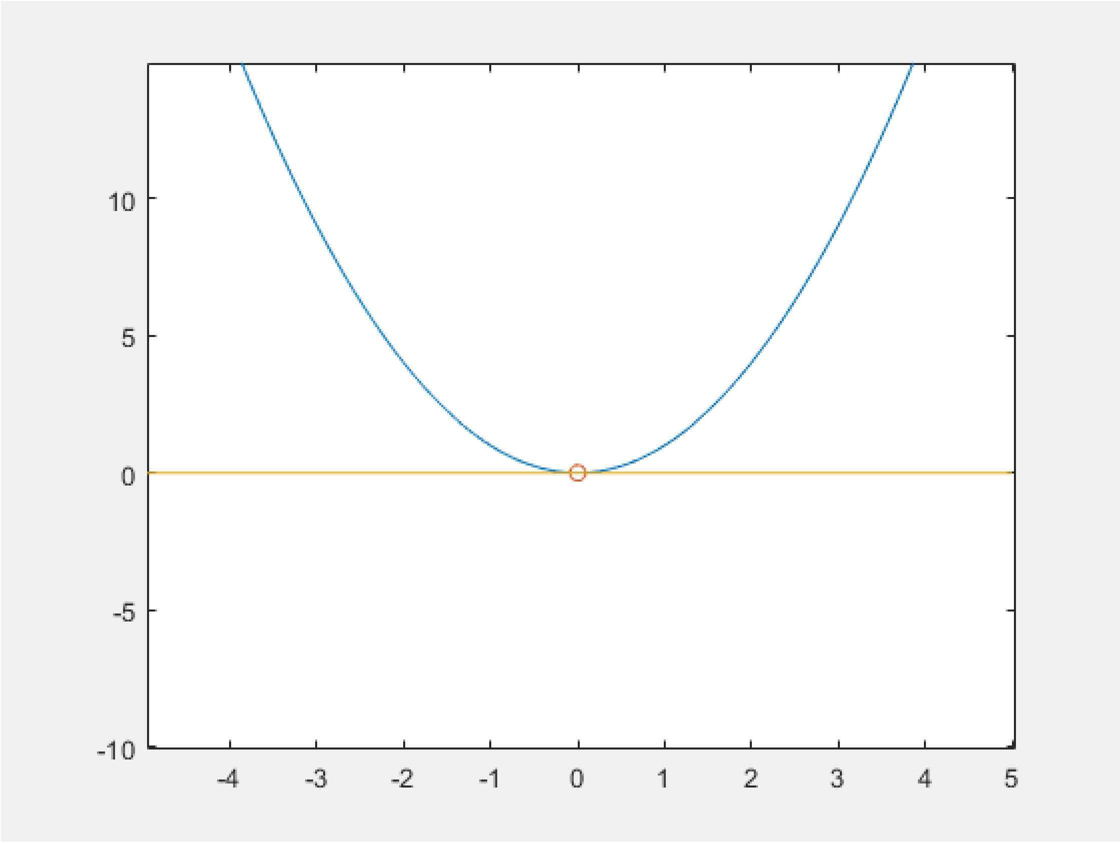
Limitazioni fzero matlab

Consideriamo la seguente funzione:

f(x)=x^2

Alternativa utilizzare 'roots'

```
>> roots([1 0])

ans =

     0
```



Esempio: Trovare lo zero della funzione |x|+1 vicino ad 1

```
>> f=@(x) abs(x)+1

f =

  function_handle with value:

    @(x)abs(x)+1

>> X=fzero(f,1)
Exiting fzero: aborting search for an interval containing a sign change
    because NaN or Inf function value encountered during search.
(Function value at -Inf is Inf.)
Check function or try again with a different starting value.

X =

   NaN

>>
```

fzero restituisce NAN se, la funzione
non ammette alcuno zero appartenente all’asse reale

Confronto metodo di bisezione

```
function [alfa,k]=bisezione(f,a,b,tol)
% La funzione approssima la radice con il metodo di bis
%
% Parametri di input
% f = funzione della quale calcolare la radice
% a = estremo sinistro dell'intervallo
% b = estremo destro dell'intervallo
% tol = precisione fissata
%
% Parametri di output
% alfa = approssimazione della radice
% k = numero di iterazioni
kmax=100;
if nargin==3
tol = 1e-8; % Tolleranza di default
end
fa = f(a);
fb = f(b);
if fa*fb>0
error('Il metodo non è applicabile')
end
c =a+ (b-a)/2;
fc =f(c);
k = 0;
while (((b-a)>tol || abs(fc)>tol) && k<kmax)
if fa*fc<0
b = c;
fb = fc;
else
a = c;
fa = fc;
end
c = a+(b-a)/2;
fc = f(c);
k = k+1;
end
alfa = c;
end
```

Confronto metodo di bisezione

Esempio: consideriamo la seguente funzione:

$$f(x)=2-e^{-x}-x^{1/2} \quad ,[a,b]=[0\ 4]$$

$$TOL=10^{-10}$$

Bisezione

```
>> [alfa,k]=bisezione(@fun,0,4,10^-10)

alfa =

    3.9211


k =

    34
```

Iterazioni

Confronto metodo di bisezione
Esempio: consideriamo la seguente funzione:

$$f(x)=2-e^{-x}-x^{1/2} \quad ,[a,b]=[0\ 4]$$

$$TOL=10^{-10}$$

Bisezione

```
>> [alfa,k]=bisezione(@fun,0,4,10^-10)

alfa =

    3.9211


k =

    34
```

Iterazioni

fzero

```
>> options = optimset('TolX',1e-10);
>> [x,fval,~,output]=fzero(@fun,[0 4],options)

x =

    3.9211


fval =

   2.6237e-11


output =

  struct with fields:

    intervaliterations: 0
            iterations: 4
             funcCount: 6
             algorithm: 'bisection, interpolation'
               message: 'Zero found in the interval [0, 4]'
```

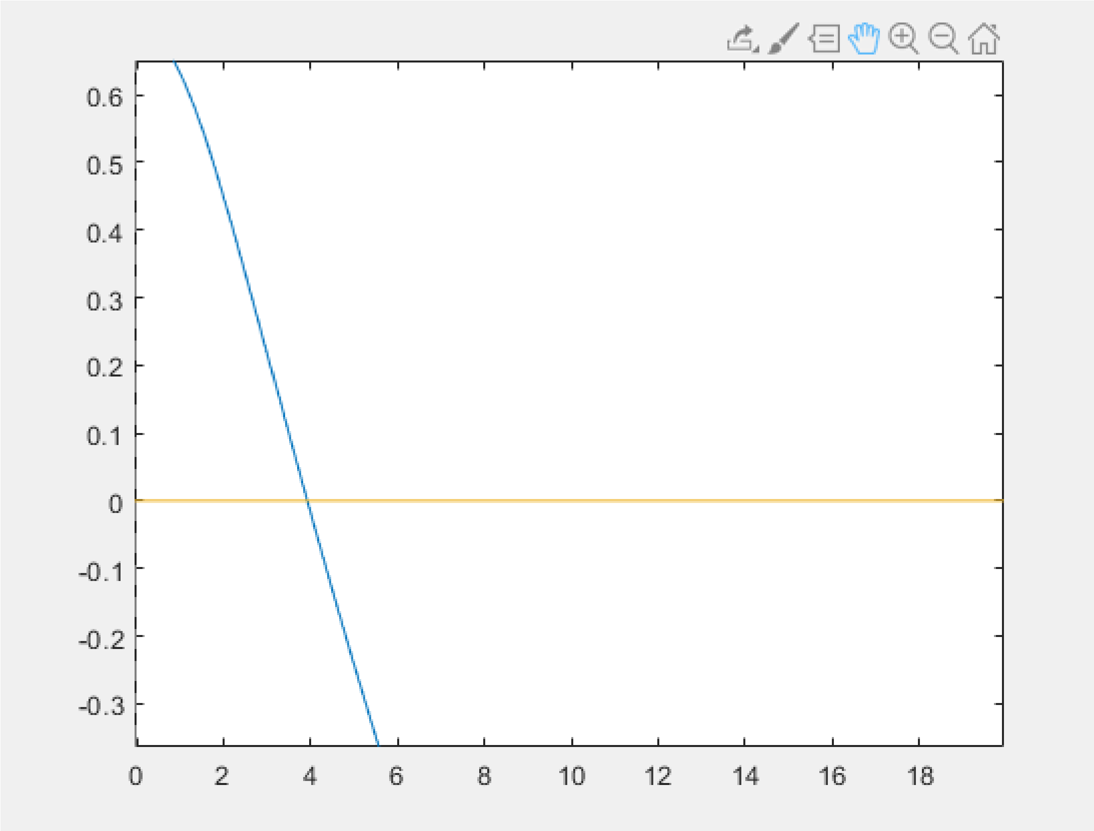
Iterazioni

Confronto metodo di bisezione

Esempio: consideriamo la seguente funzione:

$$f(x)=2-e^{-x}-x^{1/2} \quad ,[a,b]=[0\ 4]$$

$$TOL=10^{-10}$$



Confronto metodo Newton

```matlab
function [alfa,k]=newton(f,f1,x0,tol,Nmax)
%
% La funzione approssima la radice con il metodo di Newton
%
% Parametri di input
% f = funzione della quale calcolare la radice
% f1 = derivata prima della funzione f
% x0 = approssimazione iniziale della radice
% tol = precisione fissata
% Nmax = numero massimo di iterazioni fissate
%
% Parametri di output
% alfa = approssimazione della radice
% k = numero di iterazioni
%
if nargin==3
tol=1e-8;
Nmax=1000;
end
k=0;
if (f1(x0)<eps)
    error(' errore: derivata nulla')
end
x1=x0-f(x0)/f1(x0);
fx1 = f(x1);
while abs(x1-x0)>tol || abs(fx1)>tol
x0 = x1;
if (f1(x0)<eps)
    error(' errore: derivata nulla')
end
x1 = x0-f(x0)/f1(x0);
fx1 = f(x1);
k=k+1;
if k>Nmax
disp('Il metodo non converge');
alfa = inf;
break
end
end
alfa=x1;
end
```

Confronto metodo Newton

```
>> [x,fval,~,output]=fzero(@sin,3,options)

x =

    3.1416


fval =

   1.4547e-15


output =

  struct with fields:

    intervaliterations: 3
            iterations: 4
             funcCount: 11
             algorithm: 'bisection, interpolation'
               message: 'Zero found in the interval [2.83029, 3.16971]'
```

Confronto metodo Newton

In input 2 funzioni, f e f'

```
>> [x,fval,~,output]=fzero(@sin,3,options)

x =

    3.1416


fval =

   1.4547e-15


output =

  struct with fields:

    intervaliterations: 3
            iterations: 4
             funcCount: 11
             algorithm: 'bisection, interpolation'
               message: 'Zero found in the interval [2.83029, 3.16971]'
```

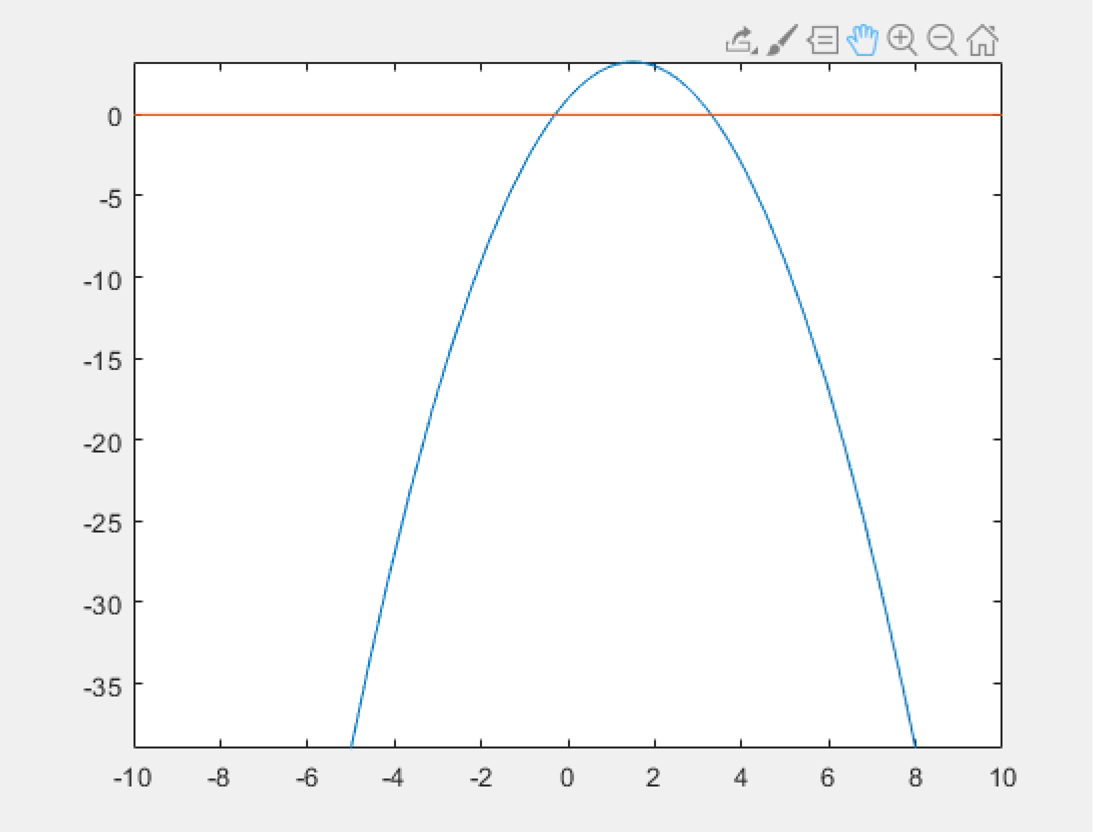
```
>> [alfa,k]=newton(@sin,@cos,3,10^-10,100)

alfa =

    3.1416


k =

     2
```

meno iterazioni

Confronto metodo Newton

```
>> f
f =

  function_handle with value:

    @(x)-x^2+3*x+1
```

```
>> [x]=fzero(f,[-4,2]);
>> x

x =

    -3.027756377319947e-01

>> [x]=fzero(f,[2,8])

x =

     3.302775637731995e+00
```



Confronto metodo Newton

```
>> [x,~,exitflag,output]=fzero(f,7)

x =

     3.302775637731995e+00


exitflag =

     1


output =

  struct with fields:

    intervaliterations: 10
            iterations: 7
             funcCount: 27
             algorithm: 'bisection, interpolation'
               message: 'Zero found in the interval [2.52, 10.1678]'

>>
```

```
>> [alfa,k]=newton(f,@(x) -2*x+3,7,eps,1000)
Error using newton (line 22)
 errore: derivata nulla
```

Confronto metodo Newton

```
>> [x,fval,exitflag,output]=fzero(f,1)

x =

   -3.027756377319947e-01


fval =

     0


exitflag =

     1


output =

  struct with fields:

    intervaliterations: 13
            iterations: 7
             funcCount: 33
             algorithm: 'bisection, interpolation'
               message: 'Zero found in the interval [-0.810193, 2.28]'

>>
```

Confronto metodo Newton

```
>> [x,fval,exitflag,output]=fzero(f,1)

x =

   -3.027756377319947e-01


fval =

     0


exitflag =

     1


output =

  struct with fields:

    intervaliterations: 13
            iterations: 7
             funcCount: 33
             algorithm: 'bisection, interpolation'
               message: 'Zero found in the interval [-0.810193, 2.28]'

>>
```

```
>> [alfa,k]=newton(f,@(x) -2*x+3,1,eps,1000)

alfa =

   -3.027756377319947e-01


k =

     7
```

Confronto metodo secanti

```
function [x,i]=secanti(x0,x1,f,tol,nmax)
%   Input
%       x0 -> il punto iniziale e prima approssimazione di x
%       x1 -> la seconda approssimazione della soluzione x
%       f -> funzione di cui valutare uno zero
%       tol -> tolleranza
%       nmax -> limite superiore al numero di iterazioni
%    Output
%       x -> la soluzione trovata
%       i -> il numero di iterazioni impiegate per ottenere la soluzione
  i=0;
  fx0=f(x0);
  err=abs(x1-x0);
  while (i<nmax && err>tol)
      fx1=f(x1);
      dfx1=(fx1-fx0)/(x1-x0);
      if abs(fx1)<=tol
         break
      end
      x2=x1-(fx1/dfx1);
      err=abs(x2-x1);
      x0=x1;
      x1=x2;
      fx0=fx1;
      i=i+1;
  end
  x=x1;
end
```

Confronto metodo Secanti

```
>> [x,fval,~,output]=fzero(@sin,3,options)

x =

    3.1416


fval =

   1.4547e-15


output =

  struct with fields:

    intervaliterations: 3
            iterations: 4
             funcCount: 11
             algorithm: 'bisection, interpolation'
               message: 'Zero found in the interval [2.83029, 3.16971]'
```

Confronto metodo Secanti

```
>> [x,fval,~,output]=fzero(@sin,3,options)

x =

    3.1416


fval =

   1.4547e-15


output =

  struct with fields:

    intervaliterations: 3
            iterations: 4
             funcCount: 11
             algorithm: 'bisection, interpolation'
               message: 'Zero found in the interval [2.83029, 3.16971]'
```

```
>> [x,i]=secanti(3,1,@sin,10^-10,1000)

x =

    3.1416


i =

     5
```

Iterazioni