CODICE PER LA FUNZIONE DI PUNTO FISSO (BOZZA)

```
function [x] = puntofisso(g,x0,toll,nmax)
%PUNTOFISSO risolve x=g(x)
%
%   x(n+1) = g( x(n) ), n=0,1,2,...
%
%  Criterio di arresto:
%    | x(n+1)-x(n) |<toll
%
%

x(1) = x0;
x(2) = g(x0);

n = 1;
while( abs(x(n+1)-x(n))>=toll && n<nmax)
    n = n+1;
    x(n+1) = g(x(n));
end
```

```
% programma di prova del punto fisso

% definisco la funzione g
%g = inline('sqrt(x)');
g  = inline('x.^3');
xx = linspace(0,2,1000);
yy = g(xx);
figure(1)
plot(xx,yy,'b')
hold on
plot(xx,xx,'m')
hold off

% definisco il punto di partenza x0
x0 = 0.5;

% definisco i parametri di arresto
toll = 1e-3;
nmax = 100;

% richiamo la funzione puntofisso
[x] = puntofisso(g,x0,toll,nmax);

% grafico dell'errore
xvera = 0.0;
errore = abs(x-xvera);
figure(2)
semilogy(errore,'o-b')
grid on
xlabel('n','fontSize',18)
ylabel('log10(errore)','fontSize',18)
```

==========
==========

PARTE DI ALGEBRA LINEARE

```
% definiamo una matrice di 3 righe
% e 2 colonne
A=[ 1 2; 3 4; 5 6]
```

```
A =

     1     2
     3     4
     5     6

% per conoscere le dimensioni
[m,n]=size(A)

m =

     3


n =

     2

% per avere solo il numero di righe
m = size(A,1)

m =

     3

% per avere solo il numero di colonne
n = size(A,2)

n =

     2

% per riferirsi ad un elemento della
% matrice
A(3,2)=A(1,1)

A =

     1     2
     3     4
     5     1

% per effettuare la trasposta
AT = A'

AT =

     1     3     5
     2     4     1

% per calcolare somma e prodotto per
% un numero
A2 = A+A

A2 =

     2     4
     6     8
    10     2

B=A+AT
??? Error using ==> plus
Matrix dimensions must agree.

A3 = 3*A
```

```
A3 =

     3     6
     9    12
    15     3

% prodotto di due matrici
ATperA=AT*A

ATperA =

    35    19
    19    21

%per elevare ogni elemento di A
% allo stesso numero (operazione punto)

A4=A.^4

A4 =

     1    16
    81   256
   625     1

% posso applicare le operazioni punto
% anche a due matrici delle stesse
% dimensioni
E = [1 -1; 2 -2; 0 3]

E =

     1    -1
     2    -2
     0     3

A

A =

     1     2
     3     4
     5     1

A.^E

ans =

    1.0000    0.5000
    9.0000    0.0625
    1.0000    1.0000

A.*E

ans =

     1    -2
     6    -8
     0     3

% posso eliminare righe o colonne di A
A

A =
```

```
     1     2
     3     4
     5     1

A(2,:)=[]

A =

     1     2
     5     1

% sopprimiamo la prima colonna di A
A(:,1)=[]

A =

     2
     1

% vediamo questo comando...
A(3,3)=4

A =

     2     0     0
     1     0     0
     0     0     4

% vediamo cosa succede in questo caso
A(:)

ans =

     2
     1
     0
     0
     0
     0
     0
     0
     4

clear



% matrice di tutti zeri
O = zeros(3,2)

O =

     0     0
     0     0
     0     0

% matrice identitÃ
I4 = eye(4)

I4 =

     1     0     0     0
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

```
% matrice diagonale di elementi
% assegnati
D = diag([1 3 -5])

D =

     1     0     0
     0     3     0
     0     0    -5

% per calcolare gli autovalori di D
spettroD = eig(D)

spettroD =

    -5
     1
     3

% calcolo il raggio spettrale di D
% max |lambda|
rho = max( abs( eig(D) ) )

rho =

     5

% la function max calcola il valore
% massimo degli elementi di un vettore
% E se la applico ad una matrice?
A=[1 2 3;4 5 6]

A =

     1     2     3
     4     5     6

max(A)

ans =

     4     5     6

% ho trovato un vettore che contiene
% il valore massimo di ogni colonna
% Allora e' facile trovare il valore
% massimo di A
maxA = max( max(A) )

maxA =

     6

% la funzione min calcola il valore
% minimo e lavora come la max
B=[-1 2 -4;3 -2 5]

B =

    -1     2    -4
     3    -2     5

min( min(B) )

ans =
```

```
    -4

% supponiamo di voler calcolare
% la somma di tutt gli elementi
% di B elevati al quadrato

% per fare la somma c'e' il comando sum


sum([1 2 -4])

ans =

    -1

% voglio sommare i primi cento numeri
% naturali
sum( 1:100 )

ans =

        5050

% e per matrici? Come lavora la sum?
B

B =

    -1     2    -4
     3    -2     5

sum(B)

ans =

     2     0     1

% somma per colonne!
sum( sum(B.^2) )

ans =

    59

% e se volessi fare la radice quadrata
% della somma dei quadrati di tutti gli
% elementi di A?
sqrt( sum( sum(B.^2) ) )

ans =

    7.6811

% vogliamo sommare tutti gli
% elementi di una matrice
B

B =

    -1     2    -4
     3    -2     5

sum( B(:) )

ans =
```

```
     3

% supponiamo di voler porre tutti gli
% elementi della seconda riga di B
% al valore 1
B(2,:)=1

B =

    -1     2    -4
     1     1     1

% lo stesso posso fare per le colonne.
% poniamo a 3 tutti gli elementi dela
% terza colonna
B(:,3)=3

B =

    -1     2     3
     1     1     3

% supponiamo di voler cancellare
% la prima e la terza colonna
B(:,[1 3])=[]

B =

     2
     1

help diary
 DIARY Save text of MATLAB session.
    DIARY FILENAME causes a copy of all subsequent command window
input
    and most of the resulting command window output to be appended to
the
    named file.  If no file is specified, the file 'diary' is used.

    DIARY OFF suspends it.
    DIARY ON turns it back on.
    DIARY, by itself, toggles the diary state.

    Use the functional form of DIARY, such as DIARY('file'),
    when the file name is stored in a string.

    See also <a href="matlab:help save">save</a>.

    Reference page in Help browser
       <a href="matlab:doc diary">doc diary</a>

% clear




A=rand(4)

A =

    0.8147    0.6324    0.9575    0.9572
    0.9058    0.0975    0.9649    0.4854
    0.1270    0.2785    0.1576    0.8003
    0.9134    0.5469    0.9706    0.1419
```

```
eig(A)

ans =

    2.4021
   -0.0346
   -0.7158
   -0.4400

help eig
 EIG    Eigenvalues and eigenvectors.
    E = EIG(X) is a vector containing the eigenvalues of a square
    matrix X.

    [V,D] = EIG(X) produces a diagonal matrix D of eigenvalues and a
    full matrix V whose columns are the corresponding eigenvectors so
    that X*V = V*D.

    [V,D] = EIG(X,'nobalance') performs the computation with
balancing
    disabled, which sometimes gives more accurate results for certain
    problems with unusual scaling. If X is symmetric, EIG
(X,'nobalance')
    is ignored since X is already balanced.

    E = EIG(A,B) is a vector containing the generalized eigenvalues
    of square matrices A and B.

    [V,D] = EIG(A,B) produces a diagonal matrix D of generalized
    eigenvalues and a full matrix V whose columns are the
    corresponding eigenvectors so that A*V = B*V*D.

    EIG(A,B,'chol') is the same as EIG(A,B) for symmetric A and
symmetric
    positive definite B.  It computes the generalized eigenvalues of
A and B
    using the Cholesky factorization of B.
    EIG(A,B,'qz') ignores the symmetry of A and B and uses the QZ
algorithm.
    In general, the two algorithms return the same result, however
using the
    QZ algorithm may be more stable for certain problems.
    The flag is ignored when A and B are not symmetric.

    See also <a href="matlab:help condeig">condeig</a>, <a
href="matlab:help eigs">eigs</a>, <a href="matlab:help ordeig">ordeig
</a>.

    Overloaded methods:
       <a href="matlab:help lti/eig">lti/eig</a>
       <a href="matlab:help sym/eig">sym/eig</a>

    Reference page in Help browser
       <a href="matlab:doc eig">doc eig</a>

diary off
```