Soluzioni: Laboratorio Calcolo Numerico labor3.pdf

Esercizio 1

function newtonfun con flag di controllo.

```
function [xv, fxv, n, flag] = newtonfun (f, f1, x0, toll, nmax)
%NEWTONFUN Metodo di Newton
%
% [xv, fxv, n, flag] = newtonfun (f, f1, x0, toll, nmax)
%
% Dati di ingresso:
%   f:      funzione
%   f1:     derivata prima
%   x0:     valore iniziale
%   toll:   tolleranza richiesta per il valore assoluto
%           della differenza di due iterate successive
%   nmax:   massimo numero di iterazioni permesse
%
% Dati di uscita:
%   xv:     vettore contenente le iterate
%   fxv:    vettore contenente i corrispondenti residui
%   n:      numero di iterazioni effettuate
%   flag:   Se flag = 1 la derivata prima si e' annullata
%           in un'iterata


n = -1;        % Inizializza il contatore n
flag = 0;      % inizializza la variabile di seggnalazione
diff = toll+1; % Inizializza diff ad un valore fittizio > toll
               % per poter entrare nel ciclo while iterativo
x = x0;        % Definisce la prima iterata

% Inizializza i vettori xv e fxv
xv  = [];
fxv = [];

% Ciclo iterativo del metodo
while (diff >= toll) && (n < nmax) && (flag ==0) % TEST DI ARRESTO
    f1x = feval(f1,x);    % Calcola il valore f'(x_n)
    if f1x == 0
        flag = 1;
    else
        fx = feval(f,x);  % Calcola il valore f(x_n)
        xv = [xv; x];     % aggiunge al vettore la nuova iterata
        fxv = [fxv; fx];  % aggiunge al vettore il nuovo residuo
        diff = -fx/f1x;   % Calcola il valore -f(x_n)/f'(x_n)
        x = x + diff;     % Calcola il valore x_{n+1}
        diff = abs(diff); % Calcola il valore |x_{n+1}-x_n|
        n = n+1;          % incrementa il contatore delle iterate
    end
end
```

script newtonscript

```
%====================================================================
% Script per il Metodo di Newton
% NEWTONSCRIPT.M
% Necessita delle Function newtonfun e risultati
%====================================================================

% Ingresso dati

disp('METODO DI NEWTON');
disp(' ');   % lascia una riga bianca in uscita su video
exprf = input('funzione (senza apici) = ','s');
exprf1 = input('derivata (senza apici) = ','s');
x0 = input('valore iniziale = ');
toll = input('tolleranza = ');
nmax = input('numero massimo di iterazioni = ');

% Non necessario per il metodo. Solo per avere un riferimento di dove
% si trova la radice di interesse
a = input('estremo sinistro a = ');
b = input('estremo destro   b = ');


% Visualizzazione di controllo dati inseriti

disp('funzione = ')
disp(exprf);
disp('derivata = ')
disp(exprf1);
disp('valore iniziale =');
disp(x0);
disp('tolleranza =');
disp(toll);
disp('numero massimo di iterazioni =');
disp(nmax);

% Non necessario per il metodo. Solo per avere un riferimento di dove
% si trova la radice di interesse
disp('estremo sinistro a =');
disp(a);
disp('estremo destro b =');
disp(b);


% Parte esecutiva

% trasforma le stringhe in funzioni
f = inline(exprf);
f1 = inline(exprf1);

% chiede l'esecuzione della function che implementa il metodo di Newton
[xv, fxv, n, flag] = newtonfun (f, f1, x0, toll, nmax);
```

```
if flag == 1 % controlla l'annullamento della derivata prima
   disp('La derivata si annulla nell''iterata');
   disp(n);
   disp('Ripetere l''esecuzione cambiando il valore iniziale');

elseif n ~= nmax  % Controlla il raggiungimento del numero massimo di iterazioni
   % visualizza i risultati
   risultati(a, b, f, xv, fxv, 'newton')

else % Raggiunto il numero massimo di iterazioni
   disp('Raggiunto il numero massimo di iterazioni possibili');
   disp('Ripetere l''esecuzione aumentando le iterazioni');
end
```

Si utilizzi lo script tre volte, definendo come valori iniziali i due estremi dell'intervallo ed un valore interno (con tolleranza `toll = 1e-8`).

Risultati con $\varepsilon \to$ `toll` $= 1e-8$, $n_{\max} = 20$, $x_0 = 0.6$.

```
funzione =
x.^2-1+exp(-x)
derivata =
2.*x-exp(-x)
valore iniziale =
    0.6000
tolleranza =
  1.0000e-008
numero massimo di iterazioni =
    20
estremo sinistro a =
    0.6000
estremo destro b =
    0.8000

f: x.^2-1+exp(-x)  Intervallo:  a=0.6 b=0.8  metodo: newton

n      x_n  f(x_n)   x_{n+1}-x_n
0     0.600000000000000   -9.12e-002   1.4003e-001
1     0.740033773575138    2.47e-002   -2.4675e-002
2     0.715359262603328    7.55e-004   -8.0203e-004
3     0.714557236677131    8.01e-007   -8.5193e-007
4     0.714556384743971    9.03e-013   0.0000e+000
```

Risultati con $\varepsilon \to$ `toll` $= 1e-8$, $n_{\max} = 20$, $x_0 = 0.8$.

```
funzione =
x.^2-1+exp(-x)
derivata =
2.*x-exp(-x)
valore iniziale =
    0.8000
```

```
tolleranza =
  1.0000e-008
numero massimo di iterazioni =
    20
estremo sinistro a =
    0.6000
estremo destro b =
    0.8000

f: x.^2-1+exp(-x)  Intervallo:  a=0.6 b=0.8  metodo: newton

n      x_n  f(x_n)   x_{n+1}-x_n
0     0.800000000000000    8.93e-002   -7.7632e-002
1     0.722367938940351    7.42e-003   -7.7324e-003
2     0.714635492296055    7.43e-005   -7.9099e-005
3     0.714556393030362    7.79e-009   0.0000e+000
```

Risultati con $\varepsilon \to$ `toll` $=$ `1e` $-$ `8`, $n_{\max} = 20$, $x_0 = 0.75$.

```
funzione =
x.^2-1+exp(-x)
derivata =
2.*x-exp(-x)
valore iniziale =
    0.7500
tolleranza =
  1.0000e-008
numero massimo di iterazioni =
    20
estremo sinistro a =
    0.6000
estremo destro b =
    0.8000

f: x.^2-1+exp(-x)  Intervallo:  a=0.6 b=0.8  metodo: newton

n      x_n  f(x_n)   x_{n+1}-x_n
0     0.750000000000000    3.49e-002   -3.3929e-002
1     0.716071021886243    1.43e-003   -1.5116e-003
2     0.714559410733616    2.84e-006   -3.0260e-006
3     0.714556384755138    1.14e-011   0.0000e+000
```

Che cosa si può notare paragonando i vari risultati di Newton ed anche i risultati del Metodo di bisezione ottenuti con la stessa tolleranza?.

È evidente che la convergenza di Newton è di ordine più elevato rispetto alla bisezione. Tra l'altro il residuo ottenuto dalla bisezione dopo 25 iterate è dell'ordine di 10^{-9}, come per il metodo di Newton partendo da $x_0 = 0.8$, ma ottenuto con **solo 3 iterazioni**. Con $x_0 = 0.6$ in 4 iterazioni si raggiunge un residuo dell'ordine di circa 9.03×10^{-13} e con $x_0 = 0.75$, più vicino alla soluzione esatta, in 3 iterazioni si raggiunge un residuo dell'ordine di circa 1.14×10^{-11}.
Il teorema di convergenza locale del Metodo di Newton per la radice α_2 (Teorema 3.12 del libro)

nell'intervallo $\mathcal{I}$ ci garantisce che, prendendo come valore iniziale un qualsiasi $x_0 \in \mathcal{I}$ avremo convergenza assicurata, con ordine almeno 2.

Si utilizzi poi nuovamente lo script, definendo come valore iniziale $x_0 = -10$ e stessa tolleranza.

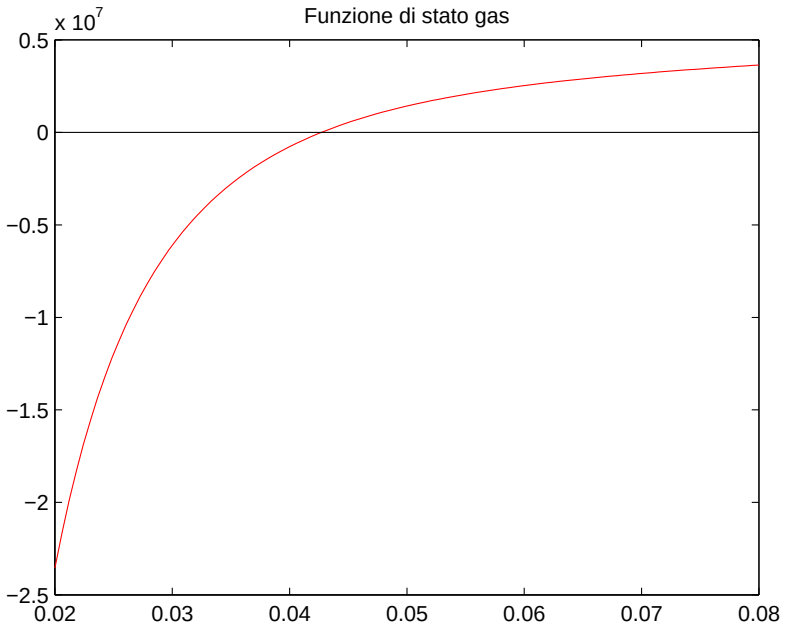
Risultati con $\varepsilon \to$ `toll` $=$ `1e` $-$ `8`, $n_{\max} = 20$, $x_0 = -10$.

```
funzione =
x.^2-1+exp(-x)
derivata =
2.*x-exp(-x)
valore iniziale =
    -10
tolleranza =
  1.0000e-008
numero massimo di iterazioni =
    20
estremo sinistro a =
    0.6000
estremo destro b =
    0.8000

f: x.^2-1+exp(-x)  Intervallo:  a=0.6 b=0.8  metodo: newton

n      x_n  f(x_n)   x_{n+1}-x_n
0   -10.000000000000000    2.21e+004   1.0036e+000
1    -8.996416659217161    8.15e+003   1.0077e+000
2    -7.988761942153644    3.01e+003   1.0158e+000
3    -6.972956019615069    1.11e+003   1.0311e+000
4    -5.941812390396354    4.15e+002   1.0571e+000
5    -4.884688655616120    1.55e+002   1.0922e+000
6    -3.792511994860075    5.78e+001   1.1116e+000
7    -2.680908185263113    2.08e+001   1.0414e+000
8    -1.639553193166244    6.84e+000   8.1132e-001
9    -0.828236399817244    1.98e+000   5.0060e-001
10    -0.327633716568730    4.95e-001   2.4231e-001
11    -0.085324837317694    9.64e-002   7.6486e-002
12    -0.008838734722866    8.96e-003   8.7244e-003
13    -0.000114378419606    1.14e-004   1.1436e-004
14    -0.000000019617401    1.96e-008   1.9617e-008
15    -0.000000000000001    6.66e-016   0.0000e+000
```

Che cosa si può dedurre dai risultati ottenuti?.

Vi è ancora convergenza, ma alla radice $\alpha_1 = 0$. Inizialmente il residuo è molto alto ed inizia a decrescere in modo coerente con l'ordine quadratico solo verso l'iterazione 12, per raggiungere, all'iterazione 16 un residuo dell'ordine di circa 6.66×10^{-16}.
Il valore $x_0 = -10$ appartiene all'intervallo di convergenza di α_1 e non a quello di α_2.

Esercizio 2
Grafico della funzione nell'intervallo $[0.02, 0.08]$



script statogas

```
%=========================================================================
% Script per risoluzione dell'equazione di stato di un gas
% STATOGAS.M
%=========================================================================

% Ingresso dati

disp('Risoluzione dell''equazione di stato di un gas');
disp(' ');   % lascia una riga bianca in uscita su video

T = input('T = ');
p = input('p = ');
nmol = input('N = ');
alpha = input('alpha = ');
beta = input('beta = ');
a = input('estremo sinistro a = ');
b = input('estremo destro   b = ');
toll = input('tolleranza = ');
nmax = input('numero massimo di iterazioni = ');

% Definizione dell'equazione e della costante

k = 1.3806503e-23;
exprf = ['(',num2str(p),'+',num2str(alpha),'.*((',num2str(nmol),...
         './x).^2)).*(x-',num2str(nmol),'*',num2str(beta),')-(',...
         num2str(k,'%15.8e'),'*',num2str(nmol),'*',num2str(T),')'];

% visualizza i dati inseriti
fprintf('\n METODO DI BISEZIONE \n');
fprintf(' per l''equazione di stato di un gas \n\n');
```

```
fprintf('funzione = %s \n\n', exprf);
fprintf('temperatura T = %6.0d K \n', T);
fprintf('pressione p = %6.2g Pa \n', p);
fprintf('N = %6.0d \n', nmol);
fprintf('alpha = %6.3g Pa m^6\n', alpha);
fprintf('beta = %6.3g m^3\n', beta);
fprintf('costante Boltzmann k = %13.8g J K^-1\n', k);
fprintf('estremo sinistro a = %6.3g \n', a);
fprintf('estremo destro   b = %6.3g \n', b);
fprintf('tolleranza = %6.0d \n', toll);
fprintf('numero massimo di iterazioni = %d \n', nmax);

% Parte esecutiva

% inline(exprf) trasforma la stringa exprf in funzione
f = inline(exprf);

% chiede l'esecuzione della function
[xv, fxv, n] = bisezfun (f, a, b, toll, nmax);


% Calcola e stampa il numero minimo di iterazioni per ottenere
% una radice approssimata con una certa accuratezza
tau = toll/2;
nminiter = ceil(log10(abs(b-a)/(tau))/log10(2) -1);
fprintf('accuratezza richiesta tau = %6.0d \n', tau);
fprintf('numero minimo di iterazioni richieste = %d \n\n', nminiter);

if n ~= nmax  % Controlla il raggiungimento del numero massimo di iterazioni
    risultati(a, b, f, xv, fxv, 'bisezione')
    fprintf('\n Approssimazione volume V = %13.8g m^3\n', xv(end));

else % Raggiunto il numero massimo di iterazioni
   disp('Raggiunto il numero massimo di iterazioni possibili');
   fprintf('Ripetere l''esecuzione con almeno %d iterazioni \n\n', nminiter);
end
```

Risultati con $\varepsilon \to$ `toll` $=$ `1e-10`, $n_{\max} = 100$, $a = 0.02, b = 0.08$.

```
 METODO DI BISEZIONE
 per l'equazione di stato di un gas

funzione = (35000000+0.401.*((1000./x).^2)).*(x-1000*4.27e-005)-(1.38065030e-023*1000*300)

temperatura T =    300 K
pressione p = 3.5e+007 Pa
N =   1000
alpha =  0.401 Pa m^6
beta = 4.27e-005 m^3
costante Boltzmann k = 1.3806503e-023 J K^-1
estremo sinistro a =   0.02
estremo destro   b =   0.08
tolleranza = 1e-010
numero massimo di iterazioni = 100
```

```
accuratezza richiesta tau = 5e-011
numero minimo di iterazioni richieste = 30


f: (35000000+0.401.*((1000./x).^2)).*(x-1000*4.27e-005)-(1.38065030e-023*1000*300)
Intervallo:  a=0.02  b=0.08  metodo: bisezione

n      x_n    f(x_n)         b_n-a_n
0     0.050000000000000    1.43e+006   6.0000e-002
1     0.035000000000000   -2.79e+006   3.0000e-002
2     0.042500000000000   -5.14e+004   1.5000e-002
3     0.046250000000000    7.90e+005   7.5000e-003
4     0.044375000000000    4.00e+005   3.7500e-003
5     0.043437500000000    1.83e+005   1.8750e-003
6     0.042968750000000    6.78e+004   9.3750e-004
7     0.042734375000000    8.75e+003   4.6875e-004
8     0.042617187500000   -2.12e+004   2.3437e-004
9     0.042675781250000   -6.18e+003   1.1719e-004
10     0.042705078125000    1.29e+003   5.8594e-005
11     0.042690429687500   -2.44e+003   2.9297e-005
12     0.042697753906250   -5.73e+002   1.4648e-005
13     0.042701416015625    3.61e+002   7.3242e-006
14     0.042699584960938   -1.06e+002   3.6621e-006
15     0.042700500488281    1.28e+002   1.8311e-006
16     0.042700042724609    1.09e+001   9.1553e-007
17     0.042699813842773   -4.75e+001   4.5776e-007
18     0.042699928283691   -1.83e+001   2.2888e-007
19     0.042699985504150   -3.70e+000   1.1444e-007
20     0.042700014114380    3.60e+000   5.7220e-008
21     0.042699999809265   -4.86e-002   2.8610e-008
22     0.042700006961823    1.77e+000   1.4305e-008
23     0.042700003385544    8.63e-001   7.1526e-009
24     0.042700001597404    4.07e-001   3.5763e-009
25     0.042700000703335    1.79e-001   1.7881e-009
26     0.042700000256300    6.53e-002   8.9407e-010
27     0.042700000032783    8.36e-003   4.4703e-010
28     0.042699999921024   -2.01e-002   2.2352e-010
29     0.042699999976903   -5.89e-003   1.1176e-010
30     0.042700000004843    1.23e-003   5.5879e-011

 Approssimazione volume V =        0.0427 m^3
```

Esercizio 3: Si desidera calcolare la radice approssimata positiva e **non nulla** della funzione

$$f(x) = 1 - x - e^{-2x}.$$

con il metodo iterativo della **secante fissa**

$$x_{n+1} = x_n - f(x_n)\,\frac{x_1 - x_0}{f(x_1) - f(x_0)}$$

che permette di determinare la radice reale contenuta nell'intervallo $[x_0, x_1]$.

function `secfis`

```
function [xv, fxv, n] = secfis (f, x0, x1, toll, nmax)
%SECFIS Metodo della secante fissa per equazione non lineare
%
% Uso:
%   [xv, fxv, n] = secfis(f, x0, x1, toll, nmax)
%
% Dati di ingresso:
%   f:      funzione
%   x0:     prima iterata
%   x1:     seconda iterata
%   toll:   tolleranza richiesta per il valore assoluto
%           tra due iterate successive
%   nmax:   massimo numero di iterate permesse
%
% Dati di uscita:
%   xv:     vettore contenente le iterate
%   fxv:    vettore contenente i corrispondenti residui
%   n:      numero di iterate della successione

n = 2;         % Inizializza il numero di iterate a 2
diff = toll+1; % Inizializza diff ad un valore fittizio > toll
               % per poter entrare nel ciclo while iterativo
fx0 = feval(f,x0);
fx1 = feval(f,x1);
% Inizializza i vettori xv e fxv
xv  = [x0;x1];
fxv = [fx0;fx1];
% Calcola il rapporto fisso
rapp = (x1-x0)/(fx1-fx0);
x = x1;
fx = fx1;

% Ciclo iterativo del metodo
while (diff >= toll) && (n < nmax)  % TEST DI ARRESTO
    diff = -fx*rapp;  % Calcola il valore -f(x_n)*rapp
    x = x + diff;     % Calcola il valore x_{n+1}
    xv = [xv; x];     % aggiunge al vettore la nuova iterata x_{n+1}
    fx = feval(f,x);  % Calcola il valore f(x_{n+1})
    fxv = [fxv; fx];  % aggiunge al vettore il nuovo residuo
    diff = abs(diff); % Calcola il valore |x_{n+1}-x_n|
    n = n+1;          % incrementa il contatore delle iterate
end
```

script secfisscript

```
%=====================================================================
% Script per il Metodo della secante fissa
% Necessita della Function SECFIS
%=====================================================================

clear all

% Ingresso dati

disp('METODO della Secante fissa');
disp(' ');   % lascia una riga bianca in uscita su video
exprf = input('funzione (senza apici) = ','s');
x0 = input('estremo sinistro e prima iterata x0 = ');
x1 = input('estremo destro e seconda iterata x1 = ');
toll = 1.e-8;
nmax = 20;

% Parte esecutiva

% inline(exprf) trasforma la stringa exprf in funzione
f = inline(exprf);

% chiede l'esecuzione della function che implementa il metodo iterativo
[xv, fxv, n] = secfis (f, x0, x1, toll, nmax);

% Uscita risultati

% Visualizza i risultati

% Visualizza l'indice dell'ultima iterata calcolata
disp(' ');
disp('Numero di iterate');
disp(n);
if n == nmax
    disp('Massimo indice dell''iterata raggiunto')
end

% Controllo
xv=xv(:);fxv=fxv(:);
if length(xv)~=length(fxv)
    error('Il vettore delle iterate e quello dei residui hanno dimensioni diverse');
end
disp('Ultima iterata');
format long e; disp(xv(end)); format short;
disp('Residuo corrispondente');
disp(fxv(end));

% Grafico dei residui
k=0:n-1;
semilogy(k,abs(fxv))
title(['Residui assoluti Secante fissa per funzione ', exprf ] );
```

script secfisscript, versione con tabella completa

```
%=====================================================================
% Script per il Metodo della secante fissa
% Necessita della Function SECFIS
%=====================================================================

clear all

% Ingresso dati

disp('METODO della Secante fissa');
disp(' ');   % lascia una riga bianca in uscita su video
exprf = input('funzione (senza apici) = ','s');
x0 = input('estremo sinistro e prima iterata x0 = ');
x1 = input('estremo destro e seconda iterata x1 = ');
toll = 1.e-8;
nmax = 20;

% Parte esecutiva

% inline(exprf) trasforma la stringa exprf in funzione
f = inline(exprf);

% chiede l'esecuzione della function che implementa il metodo iterativo
[xv, fxv, n] = secfis (f, x0, x1, toll, nmax);

% Uscita risultati

% Visualizza la tabella dei risultati

xv=xv(:);
fxv=fxv(:);
n1=n-1;
fprintf('\n funzione %s \tIntervallo:  x0=%g , x1=%g ',formula(f),x0,x1);
fprintf('\n tolleranza toll = %g \t Numero massimo iterate nmax = %g \n\n',toll,nmax);
fprintf('n \t    x_n \t\t\t\t f(x_n) \t\t |x_{n+1}-x_n|\n');
fprintf('%d\t %20.15f \t %10.2e \t %10.4e \n', ...
[0:n1;xv';fxv';[abs(xv(2:end)-xv(1:end-1));0]']);
fprintf('\n Numero di iterate  n = %g   \n',n);
if n == nmax
    fprintf('\n Massimo numero di iterate raggiunto \n');
end

% Grafico dei residui
k=0:n1;
semilogy(k,abs(fxv))
title(['Residui Secante fissa per funzione ', exprf ] );
```

Esercizio 4

Formula per la stima dell'ordine di convergenza di una successione.

$$p = \frac{\log \dfrac{|x_n - x_{n-1}|}{|x_{n-1} - x_{n-2}|}}{\log \dfrac{|x_{n-1} - x_{n-2}|}{|x_{n-2} - x_{n-3}|}}.$$

script `stimap`

```
function p = stimap (xv, nt)
%STIMAP  Stima dell'ordine di convergenza di una successione
%
% p = stimap (xv, nt)
%
% Dati di ingresso:
%   xv:     vettore colonna contenente le iterate della successione.
%   nt:     numero dei termini della successione.
%
% Dati di uscita:
%   p:      vettore colonna contenente le stime dell'ordine calcolate.
%           Le prime tre componenti del vettore sono sempre nulle.

% prealloca il vettore p
p = zeros(nt,1);

% Calcola le stime
for n = 4:nt
    diff1 = abs(xv(n)-xv(n-1));
    diff2 = abs(xv(n-1)-xv(n-2));
    diff3 = abs(xv(n-2)-xv(n-3));
    % Controlla se le differenze sono nulle per evitare i casi
    % indeterminati (numeratore e/o denominatore nullo)
    if diff1 == 0 || diff2 == 0 || diff3 == 0
        p(n) = 0;
    else
        p(n) = log(diff1/diff2) / log(diff2/diff3);
    end
end
p=p(:);
```

script `stimap`, versione con controlli

```
function p = stimap (xv, nt)
%STIMAP  Stima dell'ordine di convergenza di una successione
%
% p = stimap (xv, nt)
%
% Dati di ingresso:
%   xv:     vettore colonna contenente le iterate della successione.
%   nt:     numero dei termini della successione.
%
```

```
% Dati di uscita:
%   p:      vettore contenente le stime dell'ordine calcolate.
%           Le prime tre componenti del vettore sono sempre nulle.

% Controlla se vi sono sufficienti termini della successione
if length(xv) < nt
    p = 0;
    error (' Non ci sono abbastanza termini della successione')
else
    % prealloca il vettore p
    p = zeros(nt,1);
    % Calcola le stime
    for n = 4:nt
        diff1 = abs(xv(n)-xv(n-1));
        diff2 = abs(xv(n-1)-xv(n-2));
        diff3 = abs(xv(n-2)-xv(n-3));
        % Controlla se le differenze sono nulle per evitare i casi
        % indeterminati (numeratore e/o denominatore nullo)
        if diff1 == 0 || diff2 == 0 || diff3 == 0
            p(n) = 0;
        else
            p(n) = log(diff1/diff2) / log(diff2/diff3);
        end
    end
end
```

script secfisscript, versione con stima ordine

```
%=====================================================================
% Script per il Metodo della secante fissa
% Necessita della Function SECFIS
%=====================================================================
.........................
.........................
.........................

% Stima ordine di convergenza

p = stimap(xv,length(xv));

disp('Stime ordine di convergenza');
disp(p(4:end));
```