Esercitazione 3:
Ricerca di zeri di funzioni non–lineari

Richiami di teoria: Seprazione radici

Gli algoritmi visti a lezione necessitano di separare graficamente le zeri.
L'idea di un algormitmo di ricerca degli intervalli si basa sulla suddivisione di un intervallo $x_{\min}$, $x_{\min}$ in n suddivisioni e cercare tutti gli intervalli in cui la funzione cambia segno.

Una routine per la separazione grafica degli zeri potrebbe essere basata sul seguente pseudo-codice:

$\mathbf{Xrad} \leftarrow$ sepzeri (f, $x_{\min}$, $x_{\max}$, n)
% Dimensioni di un intervallo
$dx \leftarrow (x_{\max} - x_{\min})/n$
% Estermo sinistra di ricerca
$a \leftarrow x_{\min}$
% Contatore iniziale
$i \leftarrow 0$
% Ricerca intervallo
while $i < n$
    % Incremento contatore
    $i \leftarrow i + 1$
    % Estermo destro dell'intervallo
    $b \leftarrow a + dx$
    % Verifico cambiamento segno
    if $f(x)$ change sign in $[a, b]$
        salva intervallo $[a, b]$ per un succ. raffinamento in $\mathbf{Xrad}$
    end if
    % Aggiorna estremo sinistro
    $a \leftarrow b$
end while

Un esempio di function MatLab che implementa il metodo precedente è il seguente:

```
function Xrad = sepzeri(fun,xmin,xmax,nx,verb)
% sepzeri   Ricerca sottointervalli con variazione di segno di f(x)
%
% Sintassi: Xrad = sepzeri(fun,xmin,xmax)
%           Xrad = sepzeri(fun,xmin,xmax,nx)
%
% Input:    fun  : Funzione di cui ricercari gli zeri.
%           xmin : Punto iniziale di ricerca.
%           xmax : Punto finale di ricerca.
%           nx   : Numero di suddivisione degli intervalli da considerare.
%                  Di default sono 20.
%           verb : True (default) se richiesta visualizzazione grafica
%                  della funzione.
%
% Output:   Xrad : Matrice di due colonne che contiene i limiti delle
%                  radici. La radice k-esiama risulta contenuta
%                  nell'intervallo [Xrad(k,1) , Xrad(k,1)].
%                  Se non viene trovato nulla Xrad = [].
%
% Esempio:
%           Xrad = sepzeri(@(x) sin(x),-4*pi-0.1,4*pi+0.1)
%           Xrad = sepzeri(@(x) sin(x),-4*pi-0.1,4*pi+0.1,50,true)



% Imposta num. sudd. iniziali se non definite
if nargin<5, verb=true; end
if nargin<4, nx=20; end

% Inizializzazione variabili di uscita
Xrad = [];

% Inizializzazione parametri di ricerca
x = linspace(xmin,xmax,nx);     % Vettore dei potenziali limiti delle radici
f = feval(fun,x);               % Vettore delle f(x) nei punti degli intervalli

% Ricerca degli intervalli e loro visualizzazione
for k=1:length(x)-1
    % Verifica se il segno di f(x) cambia
    if sign(f(k)) ~= sign(f(k+1))
        % Salvataggio degli estermi dell'intervallo
        Xrad(end+1,:) = x(k:k+1);
    end
end


% Stampa un warning se non trovato
if isempty(Xrad)
```

```
    warning('Nessuna separazione degli zeri trovata.');
end


% Visualizzazione grafica della soluzione trovata se richiesta
if verb
    % Disegna funzione con almeno 101 punti o quattro volte quelli iniziali
    xp = linspace(xmin,xmax,max(nx*4+1,101));
    yp = feval(fun,xp);

    % box di visualizzazione grafica scalata (5% dell'originale)
    % per la visualizzazione di un rettangolo su ogni radice.
    ytop = 0.05*max(yp); ybot = 0.05*min(yp);    % Coordinate y
    ybox = [ybot ytop ytop ybot ybot];           % Box
    c = [0.5 0.5 0.5];       % Colore
    indx = [1 1 2 2 1];      % Indice dei punti Xrad fissato k della box

    % Disegno prima i rettangoli delle radici e poi la funzione
    figure;
    clf;
    hold on
    for i=1:size(Xrad,1)
        fill(Xrad(i,indx),ybox,c);
    end
    hold off

    % Funzione
    hold on; h=plot(xp,yp,'r-'); set(h,'LineWidth',2); hold off;
    grid on; xlabel('x'); ylabel('f(x)'); title('Separazione radici');
    axis tight;
end
```

L'esempio seguente mostra l'applicazione della function precedente alla separazione degli zeri della funzione $f(x) = \sin(x)$ nell'intervallo $[-4\pi, 4\pi]$.

```
Xrad = sepzeri(@(x) sin(x),-4*pi-0.1,4*pi+0.1);
```

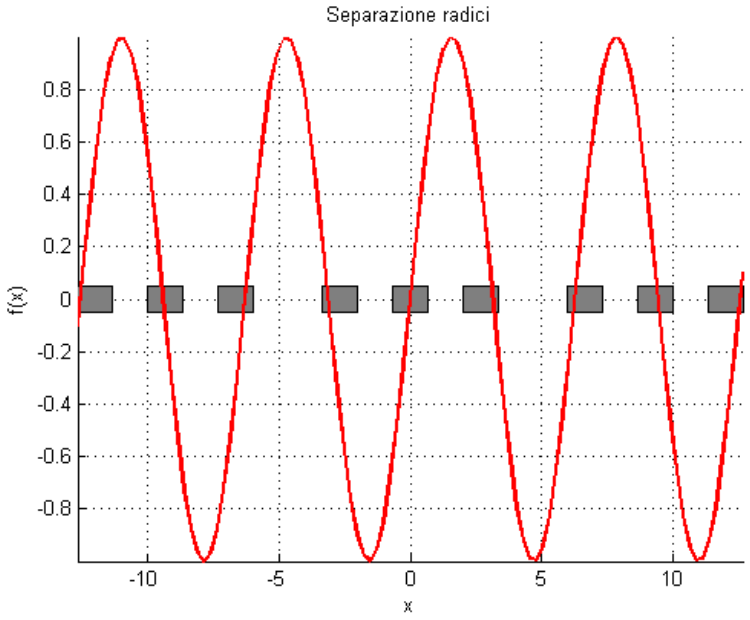

Figura 1: Esempio di separazione grafica di radici.

ottenendo i seguenti risultati:

```
Xrad =
  -12.6664  -11.3331
   -9.9998   -8.6665
   -7.3332   -5.9999
   -3.3333   -2.0000
   -0.6667    0.6667
    2.0000    3.3333
    5.9999    7.3332
    8.6665    9.9998
   11.3331   12.6664
```

Richiami di teoria: Bisezione

Il metodo di bisezione è basato sul seguente teorema:
Teorema delgi zeri delle funzioni continue: *Se* $f \in \mathbb{R}([a,b])$ *e se* $f(a) \cdot f(b) < 0$, *allora esiste (almeno) un punto* $\xi \in]a,b[$ *tale che* $f(\xi) = 0$.

Lo pseudo-codice dell'algoritmo di bisezione è il seguente:

$[x,fx] \leftarrow$ bisezione (f, a, b, tol_x, tol_f)
% Verifica correttezza intervallo iniziale
if $f(a) \cdot f(b) > 0$
    metodo di bisezione non applicabile
end if
% Ciclo principale di convergenza
while $\mathrm{iter} < \mathrm{maxiter}$
    % Calcolo del punto medio
    $xm \leftarrow (a+b)/2$
    % Valutazione della funzione nel punto medio
    $fxm \leftarrow f(xm)$
    % Test di convergenza
    if $|f(xm)| < \mathrm{tol}_f$ **or** $|dx| < \mathrm{tol}_x$
        trovata radice
    end if
    % Aggiorno gli estremi intervallo
    if $\mathrm{sign}(f(a)) == \mathrm{sign}(f(xm))$
        % zero tra $[xm, b]$
        $a \leftarrow xm$ e $fa \leftarrow fxm$
    else
        % zero tra $[a, xm]$
        $b \leftarrow xm$ e $fb \leftarrow fxm$
    end if
end while

Un esempio di routine MatLab che implementa il metodo di bisezione è la seguente:

```
function [x,fx,exitflag,output] = bisezione(fun,a,b,options,varargin)
% bisezione Ricerca di uno zero mediante metodo di bisezione
%
% Sintassi: x = bisezione(fun,a,b)
%           x = bisezione(fun,a,b,options)
%           x = bisezione(fun,a,b,options,varargin)
%           [x,fx] = bisezione(...)
%           [x,fx,exitflag] = bisezione(...)
%           [x,fx,exitflag,output] = bisezione(...)
%
% Input:    fun      : Funzione.
%           a        : Intervallo di sinistra di ricerca.
%           b        : Intervallo di destra di ricerca.
%           options  : Struttura con parametri dell'algoritmo.
%                      I seguenti parametri sono utilizzati:
%           options.xtol    : Tolleranza relativa lungo x.
%           options.ftol    : Tolleranza relativa lungo f.
%           options.maxiter : Num. max di iterazioni.
%           options.verbose : Livello di verbose (true o false).
%           varargin : Eventuali parametri aggiuntivi alla funzione fun.
%
% Output:   x        : Lo zero trovato. Vuoto se non converge.
%           fx       : Il valore dello zero trovato. Vuoto se non converge.
%           exitflag : Flag che indica se il metodo ha raggiunto la
%                      convergenza (1) oppure meno (-1).
%           output   : Struttura con informazioni di uscita dell'algoritmo.
%                      Sono restituiti i seguenti campi:
%           output.iter : Iterazioni.
%           output.a    : Estremo sinistro.
%           output.b    : Estremo destro.
%           output.xm   : Punto medio.
%           output.fxm  : Valore della funzione nel punto medio.
%
% Esempio:  % Esempio 1 (base)
%           x = bisezione(@(x)x.^2-2,0,2)
%
%           % Esempio 2 (Completo con calcolo degli errori)
%           options = struct('xtol',1e-14,'ftol',1e-14,'maxiter',1000,'verbose',true);
%           [x,fx,exitflag,output] = bisezione(@(x)x.^2-2,0,2,options);
%           xvera = sqrt(2);
%           h = semilogy(abs((output.xm-xvera)/xvera));
%           set(h,'LineWidth',2); grid on;
%           xlabel('iter'); ylabel('err. rel.'); title('Met. bisezione');
%
%           % Esempio 3 (Con passaggio di argomenti alla funzione)
%           options = struct('xtol',1e-14,'ftol',1e-14,'maxiter',1000,'verbose',false);
%           f = inline('x.^2-alpha','x','alpha');
%           alpha = 2;
```

```
%           x = bisezione(f,0,2,options,alpha)
%
%           % Esempio 4 (Con chiamata a m-file)
%           alpha = 2;
%           x = bisezione('functest1',0,2,options,alpha)



% Impostazioni di default
if nargin<4,
    xtol = 1e-8; ftol = 1e-6; maxiter = 100;
    verbose = false;
else
    xtol = options.xtol; ftol = options.ftol;
    maxiter = options.maxiter; verbose = options.verbose;
end

% Inizializzazione variabili di uscita
x = []; fx = []; exitflag = -1;
output = struct('iter',[],'a',[],'b',[],'xm',[],'fxm',[]);

% Verifica degli estremi dell'intervallo
if(a ≥ b)
    error(sprintf('Metodo di bisezione non applicabile (%f ≥ %f)!',a,b));
end

% Re-imposta le tolleranze se troppo piccole
xtol = max(xtol,5*eps);
ftol = max(ftol,5*eps);

% Funzione agli estermi
fa = feval(fun,a,varargin{:});
fb = feval(fun,b,varargin{:});

% Verifica della funzione agli estremi
if (fa>0) == (fb>0)
    error(sprintf('Metodo di bisezione non applicabile (fa*fb>0) in [%f,%f].',a,b));
end

% Costanti per il test di convergenza
xref = abs(b-a);
fref = max(abs([fa,fb]));

% Inizializzazioni
iter = 0;                          % Num. corrente di iterazioni
% output viene valutata solo se necessario
if nargout == 4
    output.iter = 0;
    output.a = repmat(NaN,1,maxiter);
    output.b = repmat(NaN,1,maxiter);
    output.xm = repmat(NaN,1,maxiter);
```

```
    output.fxm = repmat(NaN,1,maxiter);
end

% Stampa informazioni se disponibile
if verbose
    fprintf('%30s\n%4s %12s %12s\n%30s\n',...
        repmat('-',1,30),'iter','xm','f(xm)',repmat('-',1,30));
end

% Ciclo principale di convergenza
while iter ≤ maxiter
    % Aggiorno contatore delle iterazioni
    iter = iter + 1;

    % Calcolo del punto medio
    dx = (b-a)/2;
    xm = a + dx;

    % Valuto la funzione nel punto medio
    fxm = feval(fun,xm,varargin{:});

    % Salvataggio informazioni se richieste
    if nargout == 4
        output.iter = iter;
        output.a(iter) = a;
        output.b(iter) = b;
        output.xm(iter) = xm;
        output.fxm(iter) = fxm;
    end

    % Stampa informazioni se disponibile
    if verbose, fprintf('%4d %12.4e %12.4e \n',iter,xm,fxm); end

    % Test di convergenza
    if (abs(fxm)/fref < ftol) || (abs(dx)/xref < xtol)
        % trovata radice ed esci dal ciclo
        x = xm;
        fx = fxm;
        exitflag = 1;
        break;
    end

    % Aggiorno gli estremi dell'intervallo in base al segno della funzione
    if sign(fa) == sign(fxm)
        % zero tra [xm,b].
        a = xm;
        fa = fxm;
    else
        % zero tra [a,xm]
        b = xm;
        fb = fxm;
```

```
    end
end

% Sistemazione dell'output finale
if nargout == 4
    output.a = output.a(1:output.iter);
    output.b = output.b(1:output.iter);
    output.xm = output.xm(1:output.iter);
    output.fxm = output.fxm(1:output.iter);
end

if verbose, fprintf('%30s\n',repmat('-',1,30)); end
```

L'esempio seguente mostra l'applicazione della function bisezione per la ricerca dello zero della funzione $f(x) = x^2 - 2$ nell'intervallo $[0, 2]$.

```
options = struct('xtol',1e-14,'ftol',1e-14,'maxiter',1000,'verbose',true);
[x,fx,exitflag,output] = bisezione(@(x)x.^2-2,0,2,options);
xvera = sqrt(2);
semilogy(abs((output.xm-xvera)/xvera));
xlabel('iter'); ylabel('err. rel.'); title('Met. bisezione');
```

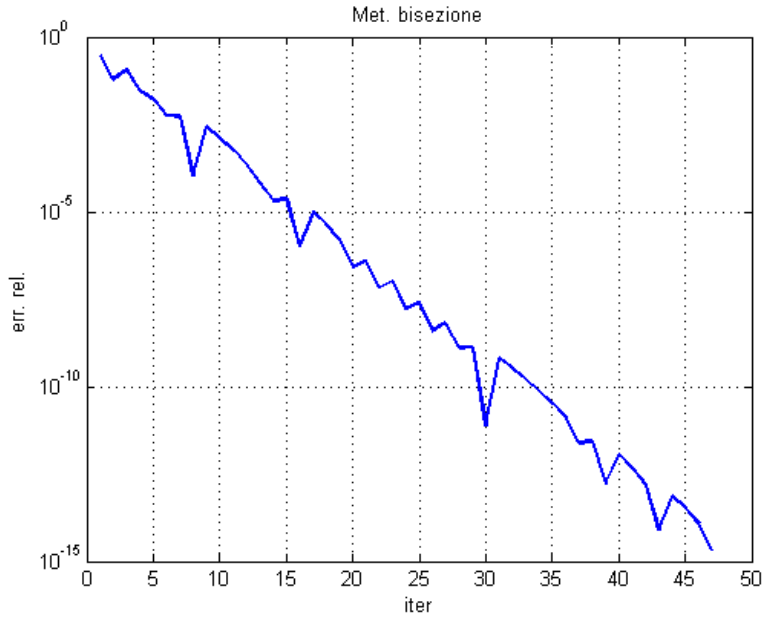

Figura 2: Andamento dell'errore relativo dell'esempio di bisezione.

I risultati delle iterazioni sono i seguenti:

```
------------------------------
iter           xm         f(xm)
------------------------------
   1  1.0000e+000 -1.0000e+000
   2  1.5000e+000  2.5000e-001
   3  1.2500e+000 -4.3750e-001
   4  1.3750e+000 -1.0938e-001
   5  1.4375e+000  6.6406e-002
   6  1.4063e+000 -2.2461e-002
   7  1.4219e+000  2.1729e-002
   8  1.4141e+000 -4.2725e-004
   9  1.4180e+000  1.0635e-002
```

```
  10  1.4160e+000  5.1003e-003
  11  1.4150e+000  2.3355e-003
  12  1.4146e+000  9.5391e-004
  13  1.4143e+000  2.6327e-004
  14  1.4142e+000 -8.2001e-005
  15  1.4142e+000  9.0633e-005
  16  1.4142e+000  4.3148e-006
  17  1.4142e+000 -3.8843e-005
  18  1.4142e+000 -1.7264e-005
  19  1.4142e+000 -6.4748e-006
  20  1.4142e+000 -1.0800e-006
  21  1.4142e+000  1.6174e-006

  ..........................

  45  1.4142e+000 -1.2856e-013
  46  1.4142e+000 -4.8184e-014
  47  1.4142e+000 -7.9936e-015
-----------------------------
```

Richiami di teoria: Newton

Il metodo di Newton è basato sulla seguente iterazione:

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}\,.$$

Il metodo di Newton applicato alla ricerca dello zero della funzione $f(x) = x^2 - 2$ con punto iniziale $x_0 = 1$ porta hai seguenti risultati:

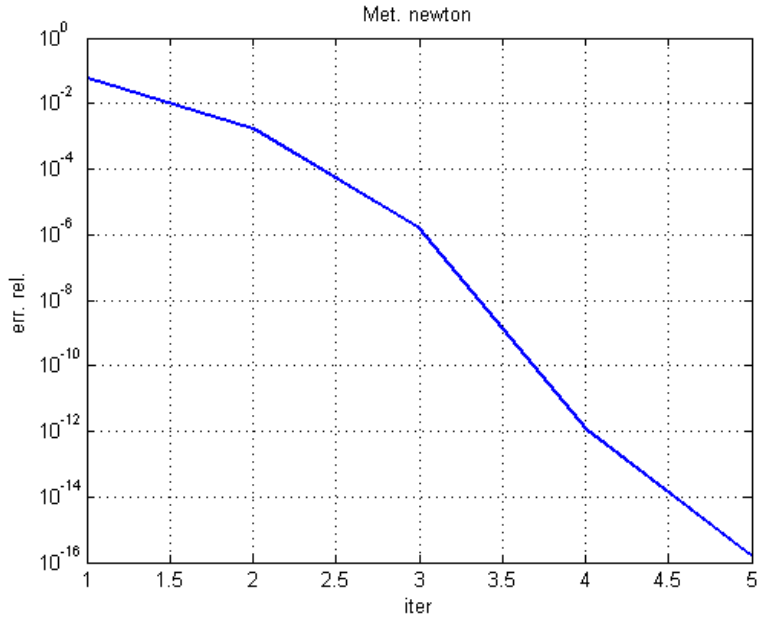

Figura 3: Andamento dell'errore relativo dell'esempio di newton.

I risultati delle iterazioni sono i seguenti:

```
-----------------------------
iter            x         f(x)
-----------------------------
   1  1.5000e+000  2.5000e-001
   2  1.4167e+000  6.9444e-003
   3  1.4142e+000  6.0073e-006
   4  1.4142e+000  4.5106e-012
   5  1.4142e+000  4.4409e-016
-----------------------------
```

Implementazione: Newton

Ad esempio, in MatLab/Octave, per definire la funzione $f(x) = x^2 - \alpha$ con la sua derivata prima $f'(x) = 2$ e seconda $f''(x) = 2$ si può utilizzare il seguente prototipo di m-file

```
function [f,df,d2f] = functest1(x,alpha)
% functest1 Funzione test 1: x^2 - alpha
%
% Sintassi: f = functest1(x,alpha)
%           [f,df] = functest1(...)
%           [f,df,d2f] = functest1(...)
%
% Input:    x      : Punto in cui valutare la funzione ed eventualmente la
%                    sua derivata prima e seconda.
%           alpha  : Parametro della funzione.
%
% Output:   f   : Funzione in f nel punto x.
%           df  : Derivata prima di f nel punto x.
%           d2f : Derivata seonda di f nel punto x.
%
% Esempio:  % Esempio
%           f = functest1(1,2)
%           [f,df] = functest1(1,2)
%           [f,df,d2f] = functest1(1,2)



% Calcola la funzione nel punto x
f = x.^2 - alpha;

if nargout > 1
    % Derivata prima richiesta nel punto x
    df = 2*x;
    if nargout > 2
        % Derivata seconda richiesta nel punto x
        d2f = 2*ones(size(x));
    end
end
```

Per richiarmare la sola valutazione della funzione si utilizzerà un comando del tipo:

```
fx = feval(fun,x,alpha);
```

mentre se si desidera la funzione e la derivata prima in x si utilizzerà il comando

```
[fx,dfx] = feval(fun,x,alpha);
```

mentre per la funzione, derivata prima e seconda in x si dovrà utilizzare la seguente istruzione

```
[fx,dfx,d2fx] = feval(fun,x,alpha);
```

Richiami di teoria: Metodo delle secanti

Il metodo delle secanti utilizza una relazione del tipo

$$x_{k+1} = x_k - f(x_k) \cdot \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})},$$

dove è necessario specificare due valori iniziali x_0 e x_1.
Il metodo delle secanti applicato alla ricerca dello zero della funzione $f(x) = x^2 - 2$ con punti iniziali $x_0 = 3$ e $x_1 = 2$ porta hai seguenti risultati:

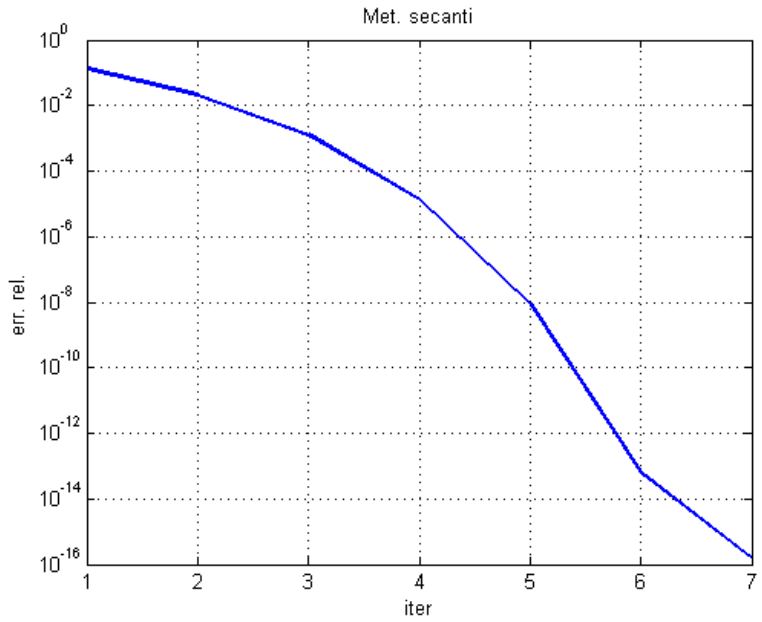

Figura 4: Andamento dell'errore relativo dell'esempio delle secanti.

I risultati delle iterazioni sono i seguenti:

```
-----------------------------
iter            x         f(x)
-----------------------------
   1  1.6000e+000  5.6000e-001
   2  1.4444e+000  8.6420e-002
   3  1.4161e+000  5.2214e-003
   4  1.4142e+000  5.5146e-005
   5  1.4142e+000  3.5945e-008
   6  1.4142e+000  2.4780e-013
   7  1.4142e+000 -4.4409e-016
-----------------------------
```

Richiami di teoria: Metodo di punto fisso

Il metodo delle secanti utilizza una relazione del tipo

$$x_{k+1} = g(x_k)\,,$$

dove è necessario specificare un valore iniziale x_0.
Il metodo di punto fisso applicato alla seguente iterazione $x_{k+1} = (4 + 11x_k - x_k^4)/11$ per la soluzione dell'equazione $f(x) = x^4 - 4 = 0$ con punto iniziale $x_0 = 1$ porta hai seguenti risultati:

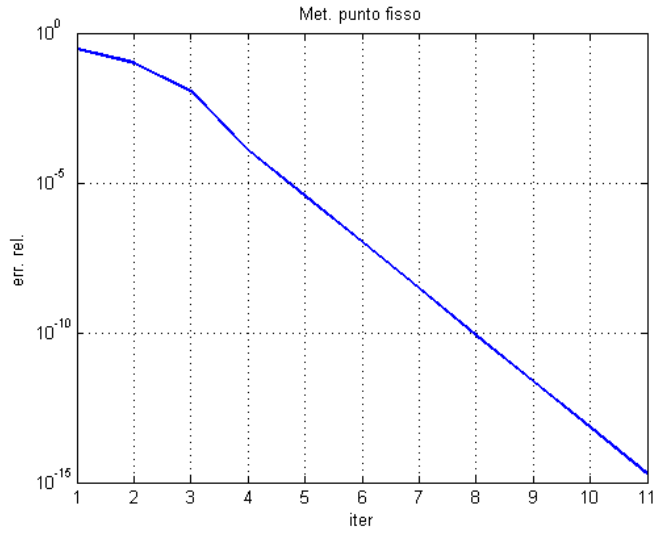

Figura 5: Andamento dell'errore relativo dell'esempio di punto fisso.

I risultati delle iterazioni sono i seguenti:

```
-----------------------------
iter         xnew    xnew-xold
-----------------------------
   1  1.2727e+000  2.7273e-001
   2  1.3978e+000  1.2510e-001
   3  1.4144e+000  1.6560e-002
   4  1.4142e+000 -1.8175e-004
   5  1.4142e+000  5.2174e-006
   6  1.4142e+000 -1.4877e-007
   7  1.4142e+000  4.2427e-009
   8  1.4142e+000 -1.2100e-010
   9  1.4142e+000  3.4508e-012
  10  1.4142e+000 -9.8588e-014
  11  1.4142e+000  2.8866e-015
-----------------------------
```

Esercizio 1

Si consideri l'equazione

$$f(x) = x^3 + 4x\cos(x) - 2\,.$$

1. Mostrare che lo zero esiste ed è unico nell'intervallo $I = [0.2, 1.2]$;
2. Utilizzare il metodo di bisezione per approssimare la soluzione;
3. Utilizzare il metodo di Newton per approssimare la soluzione;
4. Utilizzare il metodo delle secanti per approssimare la soluzione;
5. Usare un'iterazione di punto fisso del tipo $x^{k+1} = g(x^k)$ con

   $$g(x) = \frac{2 - x^3}{4\cos(x)}\,;$$

   per approssimare la soluzione;
6. Riportare l'andamento dell'errore relativo rispetto ad una soluzione di riferimento calcolata utilizzando la function `fzero` di MatLab/Octave oppure la soluzione calcolata con il metodo di Newton. Per utilizzare `fzero` si scrivano le seguenti istruzioni:

```
xrif = fzero(@(x)x.^3 + 4*x.*cos(x) - 2,0.5);
```

Risultati

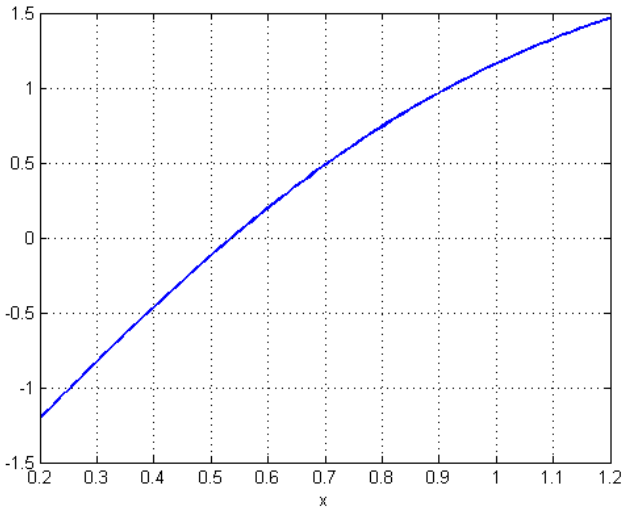

Figura 6: Andamento della funzione.

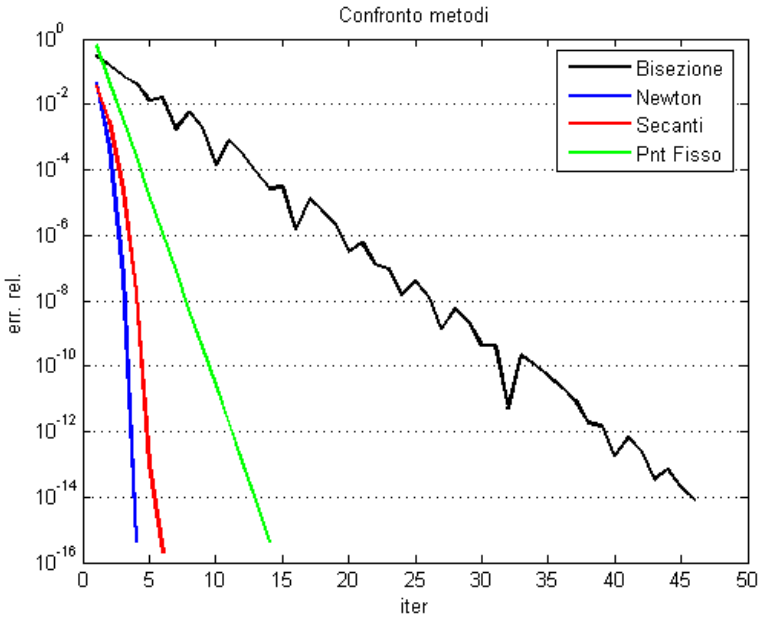

Figura 7: Andamento dell'errore relativo.

Esercizio 2

Per trovare lo zero della seguente funzione $f(x) = x - x^{1/3} - 2$ si possono utilizzare i seguenti iterazioni di punto fisso:

$$g_1(x) = x^{1/3} + 2\,,$$
$$g_2(x) = (x-2)^3\,,$$

e

$$g_3(x) = \frac{6 + 2x^{1/3}}{3 - x^{-2/3}}\,.$$

Confrontare le tre iterazioni di punto fisso utilizzando come punto iniziale il valore $x_0 = 3$.

Risultati

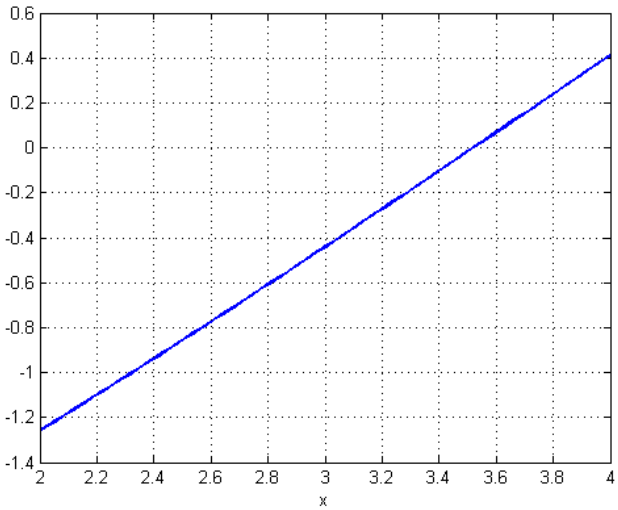

Figura 8: Andamento della funzione.

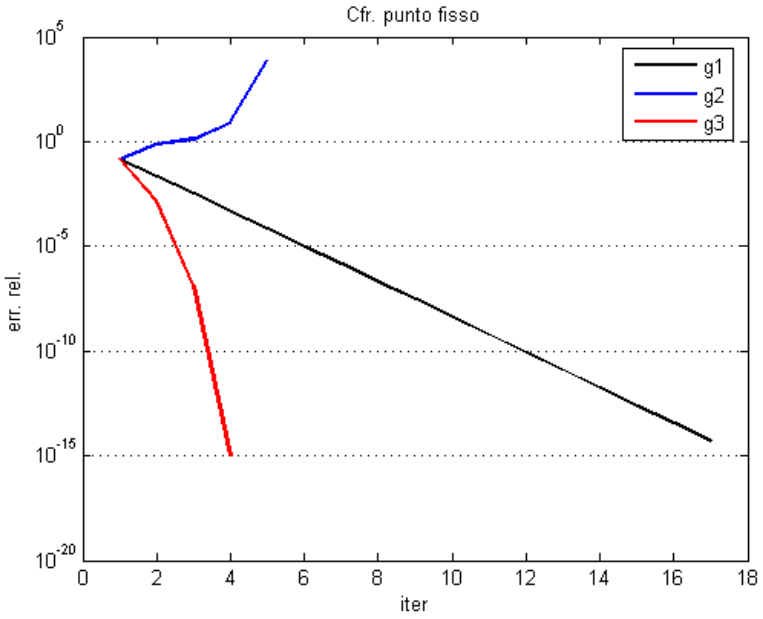

Figura 9: Andamento dell'errore relativo.

Esercizio 3

Si trovi lo zero della funzione

$$f(x) = x\cos(x) - 2\log(x)$$

nell'intervallo $(0, 4)$ mediante un'opportuna iterazione di punto fisso. Come punto iniziale si scelga il valore $x = 1.1$.

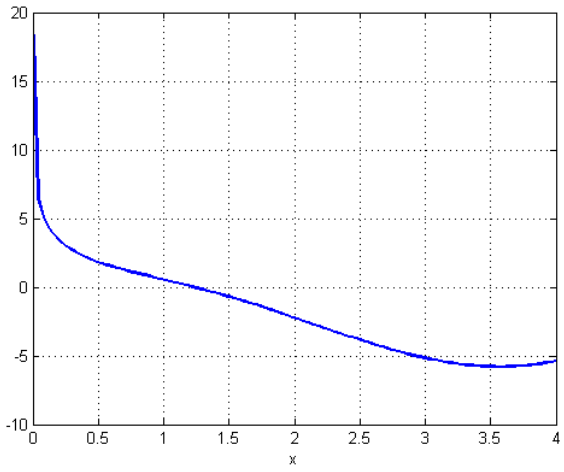

Figura 10: Andamento della funzione.

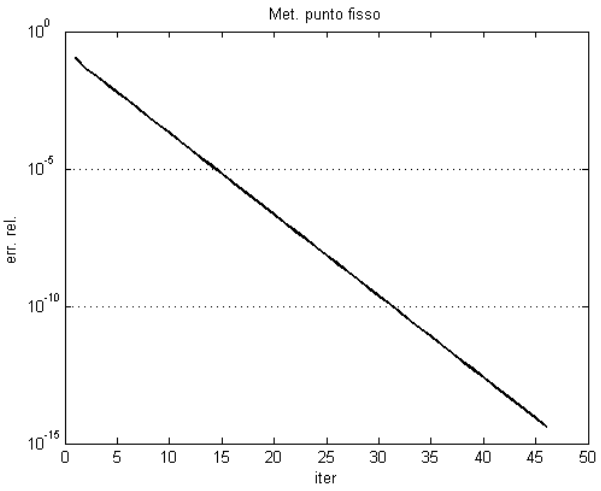

Figura 11: Andamento dell'errore relativo.

Esercizio 4 (Simulazione di un circuito elettrico)[1]

Si consideri il seguente circuito elettrico riportato nella figura sottostante, dove V rappresenta un generatore di tensione, R un resistore e D un diodo (di tipo tunnel).

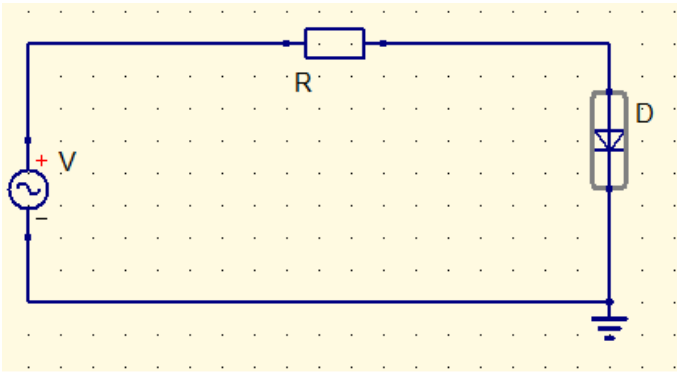

Figura 12: Circuito elettrico.

Calcolare il punto di lavoro del circuito sapendo che i tre elementi sono stati modellati nel seguente modo:

- il generatore di tensione V fornisce una tensione costante pari a 0.4 V;
- la caduta di tensione in un resistore viene modellata dalla seguente legge lineare del tipo

$$\Delta v_R = R \cdot i_R$$

  con $R = 3.3\,\mathrm{k}\,\Omega$;
- il diodo viene modellato per da un'equazione del tipo

$$i_D = \alpha \cdot (e^{v_D/\beta} - 1) - \mu v_D (v_D - \gamma) \cdot \mathrm{sign}(v)$$

  con $\alpha = 10^{-12}$ A, $\beta = 40$ V, $\gamma = 0.4$ V e $\mu = 10^{-3}$ A V^{-2}.

Risultati

Indicanto con i la corrente e con v la caduta di tensione ai capi del diodo si deve risolvere la seguente equazione non lineare:

$$f(v) = v(1/R + \mu\gamma) - \mu v^2 + \alpha(e^{v/\beta} - 1) - E/R$$

Per i dati forniti abbiamo la soluzione è a circa $v^* \approx 0.3$ V. La figura mostra il punto di lavoro del circuito.

[1] Facoltativo. Applicazione della ricerca di zeri ad un problema reale.

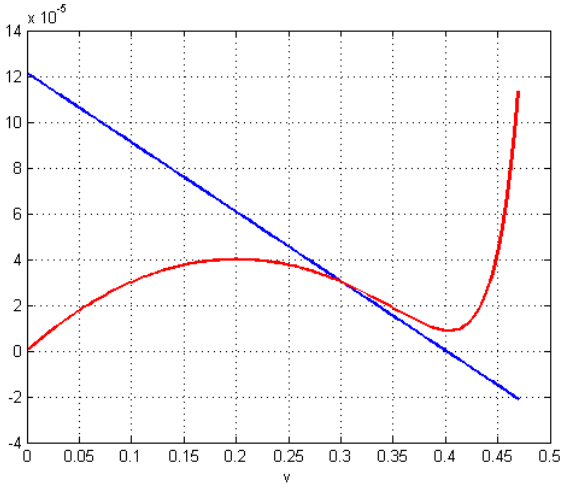

Figura 13: Punto di lavoro del circuito elettrico.