4

Chi ha paura di $f(x) = 0$?

4.1 Introduzione

In molte applicazioni intervengono equazioni che non siamo in grado di risolvere analiticamente, o la cui risoluzione risulta molto complessa e laboriosa. Un'importante classe di problemi di questo tipo può essere formulata semplicemente come il problema di determinare i valori di x tali che

$$f(x) = 0, \tag{4.1}$$

dove f è una funzione opportuna. Tali valori sono comunemente chiamati *zeri* o *radici* dell'equazione. Il problema può essere chiaramente affrontato nella forma precedente nel caso di funzioni di singola variabile reale $f : I \subset \mathrm{R} \to \mathrm{R}$. Vedremo in seguito nel Capitolo 6 come alcune tecniche possano essere estese anche al caso di sistemi di equazioni non lineari. Consideriamo il seguente esempio.

Esempio 4.1 [Galleggiamento di una sfera nell'acqua]
Una sfera di raggio r composta di materiale galleggiante è immersa nell'acqua fino ad un'altezza d. Supponiamo che la sfera abbia raggio $r = 5\,cm$ e sia costituita da un materiale con una densità ϱ_s. Ci proponiamo di calcolare di quanto la sfera s'immerga una volta che venga messa nell'acqua, ossia l'altezza d (vedi Figura 4.1). La massa d'acqua m_a che la sfera occupa una volta immersa per una profondità d sarà data da

$$m_a = \varrho_a \int_0^d \pi(r^2 - (x - r)^2)\, dx = \varrho_a \frac{\pi}{3} d^2(3r - d),$$

dove ϱ_a è la densità dell'acqua. In particolare la massa della sfera sarà

$$m_s = \varrho_s \frac{4}{3}\pi r^3.$$

Applicando il principio di Archimede $m_a = m_s$, otteniamo la seguente equazione di terzo grado

$$\frac{\pi}{3}(d^3 - 3d^2 r + 4r^3 s) = 0,$$

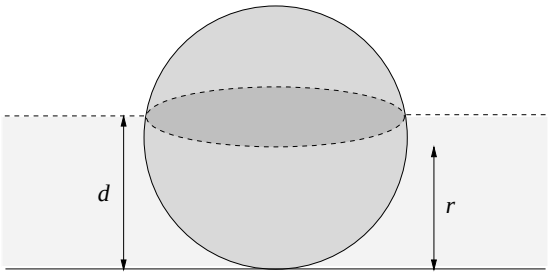

Figura 4.1 Sfera galleggiante di raggio r immersa nell'acqua fino all'altezza d.

con $s = \varrho_s/\varrho_a$ e che nel nostro caso si riduce a

$$\frac{\pi}{3}(d^3 - 15d^2 + 500s) = 0.$$

Se assumiamo che il rapporto tra le masse sia $s = 0.65$ (ad esempio nel caso in cui la sfera sia di legno) possiamo utilizzare MATLAB per realizzare il grafico del polinomio $p(d) = d^3 - 15d^2 + 500s$ per valori di d che vanno da 0 a 10 (il diametro della sfera) e visualizzarne l'intersezione con l'asse delle ascisse

```
% sferagalleggiante.m
%
s=0.65;
d=linspace(0,10);
p=d.^3-15*d.^2+500*s;
plot(d,p,d,zeros(1,100),':');
xlabel('d');
ylabel('p(d)');
```

Dalla Figura 4.2 si vede che il valore cercato di d è prossimo a 6 ossia poco più del raggio della sfera. Cambiando il valore di s nel precedente script ci accorgeremo che per $0 < s < 0.5$ la sfera galleggia con il centro sopra il livello della superficie dell'acqua ($d < 5$) mentre per $0.5 < s < 1$ la sfera galleggia con il centro sotto la superficie dell'acqua ($d > 5$). Infine se $s = 0.5$ allora avremo esattamente $d = r$. ■

Il metodo grafico considerato nel precedente esempio ci ha consentito di avere in modo rapido una prima approssimazione del risultato cercato. Per alcuni scopi pratici tale approssimazione può essere considerata sufficiente. Se però il problema che stiamo affrontando avesse davvero richiesto di determinare un valore d_* tale che $p(d_*) \approx 0$ allora basta osservare che $p(6) = 1$ per renderci conto dell'inadeguatezza di tale approssimazione.

Il precedente problema avremmo potuto risolverlo analiticamente in quanto esistono formule esplicite per il calcolo di radici di polinomi di grado minore di 5. L'applicazione di tali formule non è però così immediata come la nota formula per il calcolo di radici di polinomi di secondo grado e risulta sicuramente preferibile disporre di algoritmi numerici in grado di fornire soluzioni approssimate del problema.

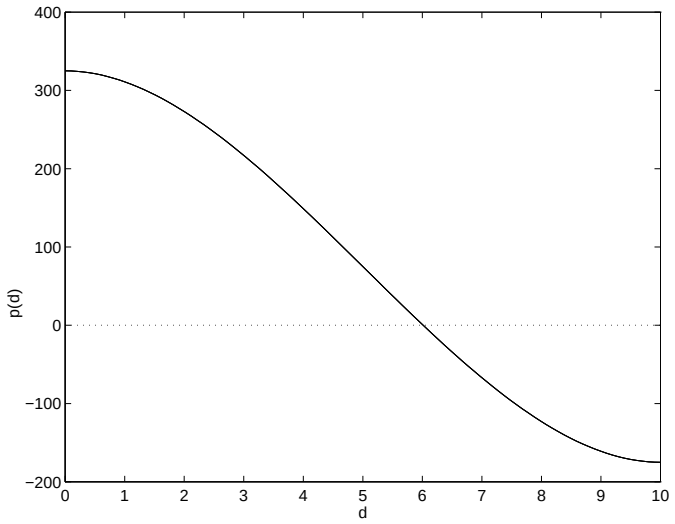

Figura 4.2 Il polinomio cubico $p(d) = d^3 - 15d^2 + 500s$, per $s = 0.65$.

Il calcolo di radici di funzioni polinomiali come

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n, \qquad n \geq 0,$$

è un caso particolare del problema generale che abbiamo considerato all'inizio della sezione, ossia della determinazione di x tale che $f(x) = 0$. Il caso di polinomi come vedremo risulta particolarmente difficile a causa della presenza di più radici a valori complessi e della possibilità di radici multiple. Ad esempio utilizzando 15 cifre significative il polinomio precedente ammette le tre radici

$$13.10867741620371, \quad 6.01389697307565, \quad -4.12257438927937,$$

della quali chiaramente solo la seconda ha senso fisico per il problema affrontato.

Esistono molti problemi che al contrario ammettono un'unica radice come nel seguente esempio

Esempio 4.2 [Decadimento di una sostanza chimica]
Supponiamo che una reazione chimica origini ad un certo istante t una concentrazione di un particolare ione data dalla legge $c(t) = 7e^{-5t} + 3e^{-2t}$. All'istante iniziale la concentrazione sarà $c(0) = 10$. Ci chiediamo a quale istante t_* la concentrazione si sarà dimezzata, ossia

$$c(t_*) = 5.$$

Tale problema sarà equivalente a quello di determinare uno zero della funzione

$$f(t) = 7e^{-5t} + 3e^{-2t} - 5.$$

In questo caso l'equazione $f(t) = 0$ ha un'unica radice t_*. Si lascia come esercizio la visualizzazione in MATLAB della funzione $f(t)$ (vedi Figura 4.3). ■

L'esempio precedente ha evidenziato l'equivalenza tra il problema della ricerca di una soluzione x per l'equazione

$$g(x) = y,$$

con $g : I \subset \mathrm{R} \to \mathrm{R}$ ed y assegnato, e la ricerca di uno zero per la funzione

$$f(x) = g(x) - y = 0.$$

Un'altra classe di problemi equivalente ad (4.1) è data dai cosiddetti *problemi di punto fisso*, nei quali si cerca un valore x tale che

$$g(x) = x,$$

con g funzione assegnata. La semplice sostituzione

$$f(x) = g(x) - x = 0,$$

consente di ridurre il problema a quello della ricerca di uno zero per f. Si noti che valendo anche la trasformazione inversa potremo sempre ricondurre la ricerca di uno zero di una funzione ad un problema di punto fisso. Come vedremo tale trasformazione è alla base di un'importante classe di metodi numerici.

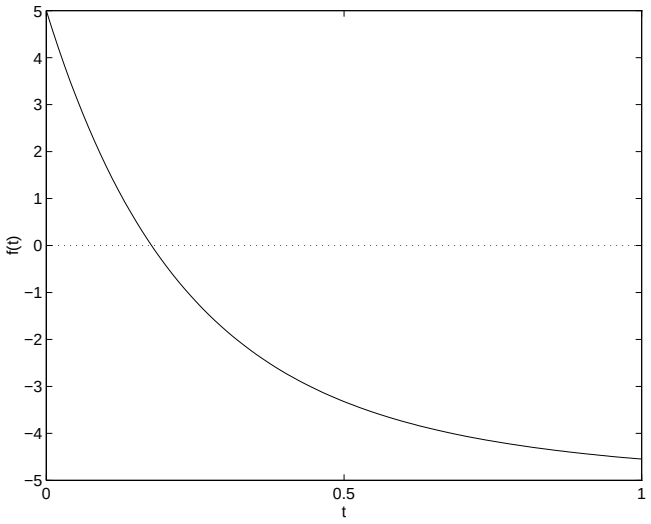

Figura 4.3 La funzione $f(t)$ nel decadimento di una sostanza chimica.

4.2 Repetita juvant: metodi iterativi

L'idea alla base dei metodi numerici per la determinazione di zeri di funzioni è il concetto di *iterazione*. In pratica si produce una sequenza di valori numeri-

ci x_0,x_1,...,x_k,... che sotto opportune ipotesi risulta convergere alla radice x_* cercata. Tipicamente i principali problemi che si devono affrontare sono

- Come iniziare l'iterazione.
- Se e con quale velocità l'iterazione converge.
- Quando terminare l'iterazione.

Vediamo un primo esempio di semplice processo iterativo.

Esempio 4.3 [Il problema della radice quadrata]
Dato A numero reale positivo ci poniamo il problema del calcolo della radice quadrata. Da un punto di vista geometrico si tratta di calcolare il lato x_* di un quadrato di area $A = x_*^2$. Avere una radice approssimata iniziale x_0 può essere visto geometricamente come un'approssimazione del quadrato di area A tramite un rettangolo di lati x_0 ed A/x_0 ed area A. Possiamo cercare di rendere il rettangolo più simile ad un quadrato sostituendo x_0 con il nuovo valore

$$x_1 = \frac{1}{2}\left(x_0 + \frac{A}{x_0}\right).$$

Questo perché il valore cercato sarà chiaramente compreso tra x_0 e A/x_0. L'idea a questo punto è quella di ripetere iterativamente il processo nella forma

$$x_{k+1} = \frac{1}{2}\left(x_k + \frac{A}{x_k}\right), \quad k \geq 0.$$

Possiamo realizzare in MATLAB uno script che visualizza il processo di trasformazione di un rettangolo in quadrato nel caso del calcolo della radice di 2.

```
% radice2.m
% Visualizza 4 iterazioni della costruzione
% geometrica della radice di due
%
A=2;
x(1)=1;
for k=1:4
   x(k+1)=0.5*(x(k)+A/x(k));
   subplot(2,2,k);
   xp=[0 x(k) x(k) 0 0];
   yp=[0 0 A/x(k) A/x(k) 0];
   fill(xp,yp,'y');
   xlabel(sprintf('x(%d)=%4.3f',k,x(k)));
   axis([0 A 0 A]);
   axis('square');
end
```

Si noti l'uso nel precedente script della funzione `fill` che può essere utilizzata con la stessa sintassi di `plot` per disegnare figure piane riempite con un colore a piacimento. Il risultato è riportato in Figura 4.4 ed in effetti il procedimento sembra convergere al valore $\sqrt{2} = 1.414\ldots$.

■

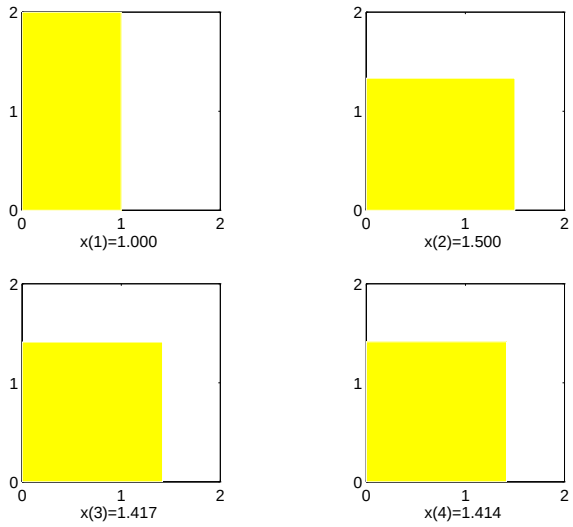

Figura 4.4 La costruzione geometrica della radice di due.

Abbiamo visto nell'esempio precedente un primo procedimento iterativo con il quale abbiamo costruito una successione di valori x_0,$x_1,\ldots,x_k\ldots$, che indicheremo con la notazione $\{x_k\}_{k\in\mathbf{N}}$, convergente al valore cercato, ossia tale che $x_k \to x_*$ per $k \to \infty$ con x_* radice dell'equazione. Chiaramente una misura della velocità di convergenza della successione sarà data dalla quantità

$$e_k = x_k - x_*.$$

Questo ci porta ad introdurre la seguente definizione

Definizione 4.1 *Data una successione $\{x_k\}_{k\in\mathbf{N}}$ convergente ad x_*, posto $e_k = x_k - x_*$, diremo che la successione ha ordine di convergenza p se esistono due numeri reali $p \geq 1$ e $c > 0$, detto costante asintotica di errore, tali che*

$$\lim_{k\to\infty} \frac{|e_{k+1}|}{|e_k|^p} = c,$$

dove $c < 1$ se $p = 1$. In particolare se $p = 1$ la convergenza si dice lineare, se $p = 2$ quadratica.

Nel caso dell'esempio precedente osserviamo che essendo $x_* = \sqrt{A}$ avremo

$$e_{k+1} = x_{k+1} - \sqrt{A} = \frac{1}{2}\left(x_k + \frac{A}{x_k}\right) - \sqrt{A} = \frac{1}{2x_k}\left(x_k - \sqrt{A}\right)^2$$

da cui

$$e_{k+1} = \frac{1}{2x_k} e_k^2.$$

Ne consegue che qualora il procedimento risulti convergente la convergenza risulterà quadratica con costante asintotica di errore pari a $1/(2\sqrt{A})$.

4.2.1 Bisezione e derivati

Il metodo di bisezione è sicuramente il più semplice metodo iterativo per approssimare gli zeri di una funzione. Il metodo è basato sul noto teorema degli zeri, ossia sul fatto che se una funzione $f(x)$ continua su un intervallo $[a,b]$ cambia di segno in tale intervallo, allora dovrà avere almeno una radice in $[a,b]$. Questo fatto può essere utilizzato per produrre una successione di intervalli sempre più piccoli contenenti una radice della funzione. Il metodo di bisezione è costruito semplicemente suddividendo ad ogni iterazione l'intervallo $[a,b]$ a metà e scegliendo come nuovo intervallo il sottointervallo nel quale si trova la radice cercata.

Possiamo procedere nel seguente modo. Supponiamo di avere localizzato un intervallo $[a,b]$ all'interno del quale la funzione f ha una unica radice x_* (ad esempio tabulando la funzione con un passo di tabulazione opportuno). Sia quindi $f(a)f(b) \leq 0$. Poniamo $x_0 = a$ ed $x_1 = b$ e definiamo

$$x_2 = \frac{1}{2}(a + b), \tag{4.2}$$

ossia il punto di mezzo dell'intervallo $[a,b]$. A questo punto per individuare se la radice si trova a destra o a sinistra di x_2 basterà calcolare $f(x_2)$. Avremo che

- se $f(x_2) = 0$ allora la radice è x_2,
- se $f(a)f(x_2) < 0$ allora la radice è in $[a,x_2]$,
- se $f(a)f(x_2) > 0$ allora la radice è in $]x_2,b]$.

Qualora $f(x_2) \neq 0$, definiremo

$$\begin{aligned} b &= x_2, \qquad \text{se} \qquad f(a)f(x_2) < 0, \\ a &= x_2, \qquad \text{se} \qquad f(a)f(x_2) > 0, \end{aligned}$$

$$x_3 = \frac{1}{2}(a + b).$$

Potremo a questo punto ripetere l'operazione sul nuovo sottointervallo $[a,b]$ che avrà ampiezza dimezzata rispetto all'intervallo di partenza ed iterare il procedimento su intervalli di ampiezza sempre più piccola. Ad ogni iterazione l'intervallo dimezzato da cui la relazione (vedi Figura 4.5)

$$|e_{k+1}| = |x_{k+1} - x_*| \leq \frac{b - a}{2^k}. \tag{4.3}$$

Da questo discende la convergenza del procedimento iterativo

$$\lim_{k\to\infty} |x_k - x_*| = 0 \quad \text{ossia} \quad \lim_{k\to\infty} x_k = x,$$

sotto la sola ipotesi di continuità della funzione f sull'intervallo.

Da un punto di vista pratico l'algoritmo può produrre intervalli troppo piccoli e la condizione di arresto $f(x_k) = 0$ può non risultare mai verificata esattamente.

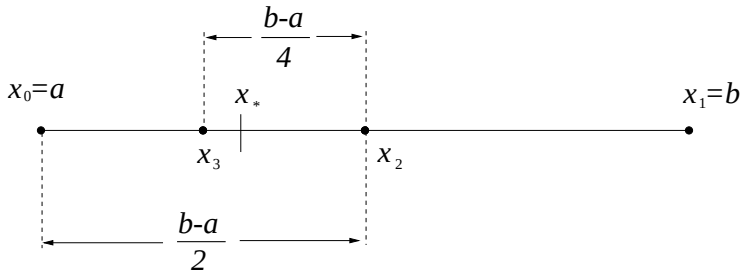

Figura 4.5 La radice x_* ed i valori x_0,x_1,x_2,x_3 dell'algoritmo di bisezione.

Un possibile criterio di arresto si ottiene arrestando l'iterazione quando l'ampiezza dell'intervallo scende al di sotto di una certa soglia. Ad esempio

$$\frac{b-a}{2^k} < \varepsilon,$$

con $\varepsilon > 0$ tolleranza prefissata. Dalla precedente disuguaglianza otteniamo che il numero di iterazioni k necessarie al fine di ottenere un errore assoluto sulla soluzione inferiore a ε soddisfa a

$$k > \log_2\left(\frac{b-a}{\varepsilon}\right) = \frac{\ln(b-a) - \ln(\varepsilon)}{\ln 2}.$$

Si noti che la scelta di un valore di ε molto piccolo corrispondente al numero macchina 0, comporterà un messaggio di errore in MATLAB

```
>> epsilon= 1e-324;
>> log(epsilon)
Warning: Log of zero
ans =
  -Inf
```

Per evitare questo problema da un punto di vista implementativo converrà utilizzare al posto ε il valore

$$\varepsilon + \texttt{eps}\max\{|a|,|b|\},$$

che garantisce il buon funzionamento dell'algoritmo anche nel caso $\varepsilon = 0$.

La seguente funzione MATLAB implementa il precedente metodo

```
function [xs,x]= Bisezione(fname,a,b,epsilon)
%
% Sintassi [xs,x]= Bisezione(fname,a,b,epsilon)
%    fname:   stringa che contiene il nome della funzione f(x)
%      a,b:   definiscono l'intervallo [a,b], f(a)f(b) <= 0
%  epsilon:   tolleranza prefissata
%        x:   successione x(k) convergente alla radice x*
%             t.c. f(x*)=0
%       xs:   approssimazione finale della radice
%
if nargin==3, epsilon = 0; end
```

```
epsilon = epsilon+eps*max(abs(a),abs(b));
x(1)=a; x(2)=b;
fa = feval(fname,a);
fb = feval(fname,b);
if fa*fb > 0
   disp('L''intervallo iniziale non e'' accettabile')
   return
end
N=ceil((log(b-a)-log(epsilon))/log(2));
for k=3:N+2
   x(k)=(a+b)/2;
   fxk=feval(fname,x(k));
   if fa*fxk <=0
      b=x(k);
   else
      a=x(k);
      fa=fxk;
   end
end
xs=(a+b)/2;
```

La funzione precedente richiede alcune osservazioni. Innanzitutto abbiamo usato la funzione MATLAB `feval` la cui sintassi è

Parametri in Uscita = feval(*Nome funzione*, *Parametri in Ingresso*),

dove *Nome funzione* è una stringa contenente il nome di una funzione MATLAB (tipicamente definita da un m file) oppure un'espressione che rappresenta una funzione alla quale sono passati i *Parametri in Ingresso* e che fornisce i *Parametri in Uscita*. La funzione `max(x,y)` restituisce il massimo tra i valori di x e y. In particolare `max(x)` con x vettore restituisce il massimo del vettore, se x è una matrice restituisce un vettore contente i massimi elementi di ogni colonna della matrice.

Infine osserviamo che il fatto che l'intervallo sia dimezzato ad ogni iterazione fa si che la stima dell'errore (4.3) sia dimezzata ad ogni iterazione. Il metodo pur non verificando la definizione data in precedenza sull'ordine di convergenza lineare[1] all'atto pratico si comporterà come un metodo con una convergenza lineare. Per illustrare questo aspetto applichiamo la precedente funzione `Bisezione` a $f(x) = \sin(x) - 1/4$ nell'intervallo [0,1.5] otteniamo

```
>> f=fcnchk('sin(x)-1/4');
>> [xs,x]=Bisezione(f,0,1.5,.001);
>> fprintf('%f\t%e\n',[x;abs(asin(1/4)-x)/asin(1/4)])
0.000000        1.000000e+00
1.500000        4.936356e+00
0.750000        1.968178e+00
0.375000        4.840891e-01
0.187500        2.579555e-01
0.281250        1.130668e-01
0.234375        7.244434e-02
0.257813        2.031122e-02
```

[1] In generale per il metodo di bisezione il limite del rapporto $|e_{k+1}|/|e_k|$ non esiste.

```
0.246094        2.606656e-02
0.251953        2.877669e-03
0.254883        8.716777e-03
0.253418        2.919554e-03
0.252686        2.094241e-05
```

dove la seconda colonna indica l'errore relativo. Si noti l'uso della funzione MATLAB

$$\textit{Variabile}=\texttt{fcnchk(}\textit{Stringa}\texttt{)},$$

che consente di definire semplici funzioni con una singola istruzione ed assegnarle ad una variabile. In particolare alla funzione può essere passato il nome di un'altra funzione MATLAB in forma di m-file. Questo consente di definire funzioni estremamente versatili alle quali passare tra apici nomi di m-files, oppure semplici funzioni elementari come `'sin(x)-1/4'`. Per maggiori informazioni si consulti l'help in linea per le funzioni `fcnchk` ed `inline`.

Una variante del metodo di bisezione è data dal *metodo di falsa posizione* o *regula falsi*. Data una funzione continua sull'intervallo $[a,b]$ con $f(a)f(b) < 0$, il metodo di bisezione assume il punto di mezzo dell'intervallo $(a+b)/2$ come prima approssimazione della radice cercata. Un'approssimazione migliore la otteniamo considerando come prima approssimazione della radice l'intersezione con l'asse delle x della retta passante per i punti $(a,f(a))$ e $(b(f(b))$ (vedi Figura 4.6).

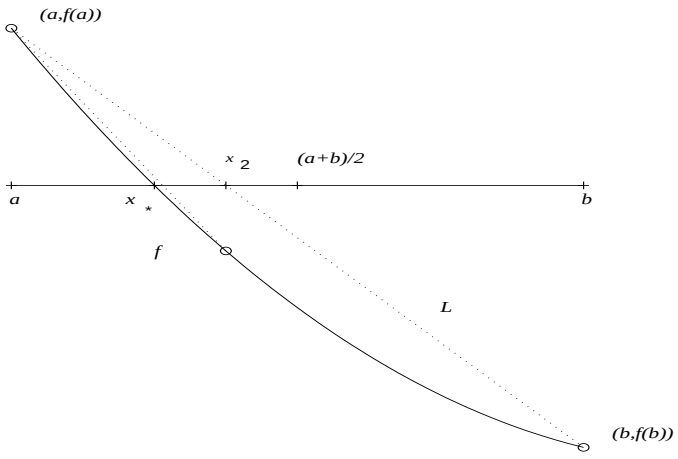

Figura 4.6 La radice x_* ed i primi valori dell'algoritmo di falsa posizione.

L'equazione della retta per $(a,f(a))$ e $(b(f(b))$ è data da

$$L(x) = f(b) + \frac{f(b)-f(a)}{b-a}(x-b).$$

Definiamo quindi come in precedenza $x_0 = a$, $x_1 = b$ ed x_2 tale che $L(x_2) = 0$. Avremo quindi

$$x_2 = b - f(b)\frac{b-a}{f(b)-f(a)}. \tag{4.4}$$

Possiamo poi procedere come nel caso della bisezione per individuare il sottointervallo di appartenenza della radice

- se $f(x_2) = 0$ allora la radice è x_2
- se $f(a)f(x_2) < 0$ allora la radice è in $[a,x_2]$
- se $f(a)f(x_2) > 0$ allora la radice è in $]x_2,b]$,

ed iterare il procedimento sul nuovo sottointervallo.

Si può dimostrare anche in questo caso sotto la sola ipotesi di continuità la convergenza della successione x_k così generata alla radice x_* tale che $f(x_*) = 0$. Un'importante differenza rispetto al metodo di bisezione è data dal fatto che l'ampiezza dell'intervallo contenente la radice pur diminuendo può non andare a zero. Ad esempio se il grafico di $f(x)$ è concavo in prossimità della radice x_* uno dei due estremi dell'intervallo risulterà fissato mentre l'altro si avvicinerà sempre più alla soluzione[2] (vedi Figura 4.6).

Di conseguenza il criterio di arresto che abbiamo utilizzato nell'algoritmo di bisezione non è utilizzabile in questo caso. Al fine di garantire l'arresto dell'algoritmo ne terminiamo l'iterazione se una qualunque delle seguenti tre condizioni risulta verificata

- La lunghezza dell'intervallo risulta minore di ε, tolleranza positiva prefissata. Ci garantisce che la distanza tra la radice calcolata e la radice esatta è minore di ε. Questo non significa che f sia piccola in corrispondenza della radice approssimata. Nell'esempio 4.1 abbiamo visto che la radice cercata è $d_* = 6.0138\ldots$. Se consideriamo la radice approssimata $\tilde{d} = 6$ allora $d_* - \tilde{d} = 0.0138\ldots$, mentre $p(\tilde{d}) = 1$.

- Il valore assoluto di f in corrispondenza della radice approssimata $\tilde{x}$ è minore di δ tolleranza positiva prefissata. Questo non significa necessariamente che $\tilde{x}$ sia prossimo alla radice esatta x_*. Ad esempio nel caso di $f(x) = (x-1)^5$, allora $\tilde{x} = 1.3$ fornisce $f(\tilde{x}) = 0.00243$ mentre $x_* - \tilde{x} = .3$.

- Il numero di iterazioni supera un valore *maxi* prefissato. Questo garantisce l'arresto dell'algoritmo qualora nessuna delle due precedenti condizioni sia soddisfatta.

Lo script MATLAB che implementa la precedente strategia è il seguente

[2]Se l'informazione sulla convessità della funzione è nota a priori non sarà necessario il test sull'intervallo di appartenenza della radice. In questa versione semplificata il metodo è detto *metodo delle corde*.

```
function [xs,x]= Fposizione(fname,a,b,epsilon,delta,maxi)
%
% Sintassi [xs,x]= Fposizione(fname,a,b,epsilon,delta,maxi)
%          fname:   stringa che contiene il nome della funzione f(x)
%            a,b:   definiscono l'intervallo [a,b], f(a)f(b) <= 0
% epsilon, delta:   tolleranze prefissate
%           maxi:   lunghezza massima della successione x(k)
%              x:   successione x(k) convergente a x t.c. f(x)=0
%             xs:   approssimazione della radice
%
x(1)=a; x(2)=b;
fa = feval(fname,a);
fb = feval(fname,b);
if fa*fb > 0
   disp('L''intervallo iniziale non e'' accettabile')
   return
end
k=2;
fxk=fa;
while (k<maxi)&(abs(b-a)>epsilon)&(abs(fxk)>delta)
   k=k+1;
   x(k)=b-fb*(b-a)/(fb-fa);
   fxk=feval(fname,x(k));
   if fa*fxk <=0
      b=x(k);
      fb=fxk;
   else
      a=x(k);
      fa=fxk;
   end
end
xs=b-fb*(b-a)/(fb-fa);
```

Se applichiamo la precedente funzione al problema della determinazione della radice di $f(x) = \sin(x) - 1/4$ in [0,1.5] otteniamo

```
>> f=fcnchk('sin(x)-1/4');
>> [xs,x]=Fposizione(f,0,1.5,0,0,7);
>> fprintf('%f\t%e\n',[x;abs(asin(1/4)-x)/asin(1/4)])
0.000000        1.000000e+00
1.500000        4.936356e+00
0.375942        4.878160e-01
0.255987        1.308830e-02
0.252751        2.816300e-04
0.252682        6.020314e-06
0.252680        1.286761e-07
```

da cui risulta evidente una maggiore accuratezza a parità di iterazioni rispetto al metodo di bisezione.

Osservazione 4.1 Anche per il metodo di falsa posizione si dimostra la convergenza sotto la sola ipotesi di continuità della funzione. Il metodo inoltre ha ordine di convergenza lineare equivalente all'algoritmo di bisezione. La maggiore accuratezza rispetto al metodo di bisezione osservata nell'esempio precedente è

dovuta alla particolare scelta della funzione $f(x)$. Per convincersi di questo invitiamo il lettore a sperimentare il comportamento del metodo di bisezione e di falsa posizione nel caso della determinazione di uno zero in $[0{,}1.5]$ per la funzione $f(x) = \sin(x) - 0.9$. ■

4.2.2 Il metodo di Newton e sue approssimazioni

Una caratteristica comune dei metodi sviluppati in precedenza è quella di utilizzare esclusivamente i valori della funzione f congiuntamente ad un meccanismo di scelta dell'intervallo. Supponiamo ora di conoscere oltre alla funzione f in $[a,b]$ anche la sua derivata prima f' in un punto c sufficientemente vicino alla radice cercata x_*. La retta tangente il grafico di f nel punto c di equazione

$$T(x) = f(c) + (x - c)f'(c),$$

può essere assunta come modello lineare della funzione in prossimità del punto c (vedi Figura 4.7).

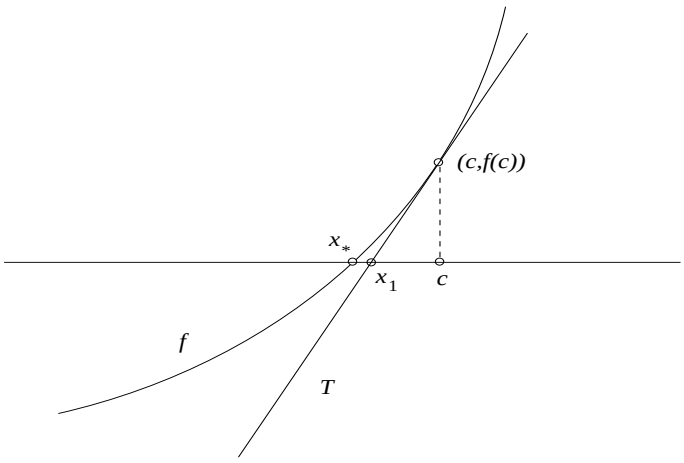

Figura 4.7 Costruzione geometrica del metodo di Newton.

L'intersezione di tale retta con l'asse delle ascisse, analogamente a quanto fatto nel metodo di falsa posizione, potrà essere utilizzata come approssimazione della radice cercata

$$x_1 = c - \frac{f(c)}{f'(c)}.$$

Il processo, qualora si conosca il valore della derivata di f anche in x_1 può essere ripetuto utilizzando x_1 al posto di c. Abbiamo quindi definito il procedimento iterativo per $k \geq 0$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \qquad x_0 = c. \tag{4.5}$$

A differenza dei metodi basati sulla scelta dell'intervallo, il metodo di Newton è definito tramite un processo iterativo a partire da un singolo punto. In particolare non avremo a priori nessun controllo sulla posizione del punto successivo all'interno di un certo intervallo dell'asse delle ascisse. Nella peggiore delle ipotesi il metodo può risultare divergente e causare un messaggio di tipo overflow (`Inf`) o not a number (`Nan`). Ad esempio se per un qualche k la derivata prima $f(x_k)$ è equivalente al numero macchina 0 allora la procedura per il calcolo di x_{k+1} comporterà una divisione per zero e quindi x_{k+1} non sarà definito (vedi Figura 4.8).

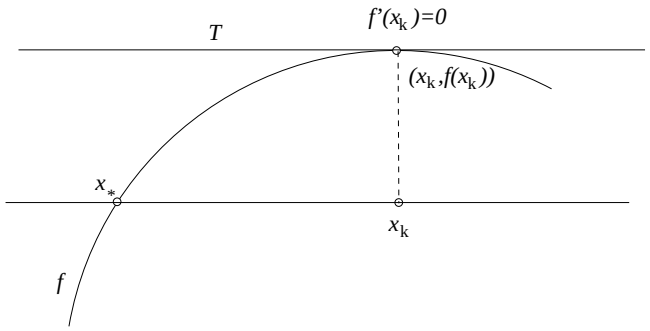

Figura 4.8 Il caso critico per il metodo di Newton.

Non solo, anche partendo da un valore iniziale molto vicino alla radice, se la derivata prima della funzione risulta nulla in prossimità di tale valore il metodo di Newton può risultare divergente. Come vedremo qualora siano soddisfatte opportune ipotesi il metodo non solo risulterà convergente ma la convergenza sarà quadratica invece che lineare. Più precisamente

Teorema 4.1 *Siano $f(x)$, $f'(x)$ definite su un intervallo $I = [x_* - \lambda, x_* + \lambda]$, con $\lambda > 0$ e $f(x_*) = 0$, ed esistano due costanti positive α e β tali che*

(a) $|f'(x)| \geq \beta$ *per tutti gli x in I.*

(b) $|f'(x) - f'(y)| \leq \alpha|x - y|$ *per tutti gli x,y in I.*

(c) $\lambda < 2\beta/\alpha$.

Se x_k è un punto dell'intervallo I allora

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

ed inoltre

$$|x_{k+1} - x_*| \leq \frac{\alpha}{2\beta}|x_k - x_*|^2 < |x_k - x_*|. \tag{4.6}$$

DIMOSTRAZIONE.
Poiché $f(x_*) = 0$ potremo scrivere

$$-f(x_k) = f(x_*) - f(x_k) = \int_{x_k}^{x_*} f'(y)dy$$

e quindi

$$-f(x_k) - (x_* - x_k)f'(x_k) = \int_{x_k}^{x_*} (f'(y) - f'(x_k))dy.$$

Dividendo per $f'(x_k)$ che sappiamo essere non nullo otteniamo

$$-\frac{f(x_k)}{f'(x_k)} - (x_* - x_k) = \frac{1}{f'(x_k)} \int_{x_k}^{x_*} (f'(y) - f'(x_k))dy,$$

che può essere riscritta come

$$x_{k+1} - x_* = \frac{1}{f'(x_k)} \int_{x_k}^{x_*} (f'(y) - f'(x_k))dy.$$

Passando ai valori assoluti ed utilizzando le ipotesi (a) e (b)

$$\begin{aligned} |x_{k+1} - x_*| &\leq \frac{1}{\beta} \int_{x_k}^{x_*} |f'(y) - f'(x_k)|dy \\ &\leq \frac{\alpha}{\beta} \int_{x_k}^{x_*} |y - x_k|dy = \frac{\alpha}{2\beta}|x_k - x_*|^2. \end{aligned}$$

Infine poiché $x_k \in I$ avremo $|x_k - x_*| \leq \lambda$ e quindi dalla (c)

$$|x_{k+1} - x_*| \leq \frac{\alpha}{2\beta}|x_k - x_*|^2 \leq \frac{\alpha\lambda}{2\beta}|x_k - x_*| < |x_k - x_*|.$$

■

La seguente definizione risulta utile per precisare alcuni concetti sulla convergenza del metodo di Newton.

Definizione 4.2 *Supponiamo che la funzione $f(x)$ assieme alle sue derivate $f'(x)$, $f''(x)$, ...,$f^{(n)}(x)$ siano definite e continue su un intervallo I contenente il punto x_*. Diremo che la funzione $f(x)$ ha una radice di ordine n in $x = x_*$ se*

$$f(x_*) = 0, \quad f'(x_*) = 0, \quad f''(x_*) = 0, \quad \ldots \quad f^{(n-1)}(x_*) = 0, \quad f^{(n)}(x_*) \neq 0.$$

Una radice di ordine $n = 1$ è detta radice semplice, se $n > 1$ è detta radice multipla.

Le ipotesi del precedente teorema garantiscono che x_* sia una radice semplice (ipotesi (a)) e che il comportamento della funzione non sia eccessivamente non lineare[3] (ipotesi (b)). Qualora la derivata seconda di f sia nota è possibile stimare il valore di α con il $\sup_{x \in I}\{|f''(x)|\}$.

Infine l'ipotesi (c) garantisce che il valore iniziale x_k sia scelto sufficientemente vicino alla radice x_*. In particolare in questa situazione avremo che il metodo di Newton risulterà convergente con una convergenza quadratica (vedi (4.6)). Più in generale si dimostra (vedi Esercizio 4.6) il seguente risultato.

[3] Tale ipotesi equivale alla richiesta che $f'(x)$ sia Lipschitziana sull'intervallo I (si veda il Capitolo 8).

Proposizione 4.1 *Supponiamo che il metodo di Newton produca una successione* $\{x_k\}$ *convergente alla radice* x_*. *Se* x_* *è una radice semplice, la convergenza è quadratica ed in particolare avremo*

$$\lim_{k\to\infty} \frac{|x_{k+1} - x_*|}{|x_k - x_*|^2} = \frac{|f''(x_*)|}{2|f'(x_*)|}. \tag{4.7}$$

Nel caso in cui x_* *sia una radice multipla di ordine* $n > 1$, *la convergenza è lineare ed avremo*

$$\lim_{k\to\infty} \frac{|x_{k+1} - x_*|}{|x_k - x_*|} = \frac{n-1}{n}. \tag{4.8}$$

Un implementazione MATLAB del metodo di Newton è data dalla seguente funzione

```
function [xs,x]= Newton(fname,f1name,x0,epsilon,delta,maxi)
%
%  Sintassi [xs,x]= Newton(fname,f1name,x0,epsilon,delta,maxi)
%          fname:   stringa che contiene il nome della funzione f(x)
%         f1name:   stringa che contiene il nome della derivata f'(x)
%             x0:   definisce l'approssimazione iniziale della radice
% epsilon, delta:   tolleranze prefissate
%           maxi:   lunghezza massima della successione x(k)
%              x:   successione x(k) convergente a x t.c. f(x)=0
%             xs:   approssimazione della radice
%
x(1)=x0;
fxk = feval(fname,x0);
f1xk = feval(f1name,x0);
if abs(f1xk) < eps*abs(fxk)
   disp('La derivata f''(x0) e'' nulla')
   return
end
for k=1:maxi-1
   x(k+1)=x(k)-fxk/f1xk ;
   fxk=feval(fname,x(k+1));
   f1xk = feval(f1name,x(k+1));
   if (abs(x(k+1)-x(k))<epsilon)|(abs(fxk)<delta)
      break;
   end
   if abs(f1xk) < eps*abs(fxk)
      fprint('La derivata f''(x(%d)) e'' nulla',k);
      return
   end
end
xs=x(k+1);
```

Possiamo applicarla al precedente problema del calcolo della radice di $\sin(x) - 1/4$ in [0,1.5]. Utilizzando come approssimazione iniziale l'estremo destro dell'intervallo otteniamo

```
>> f=fcnchk('sin(x)-1/4');
>> [xs,x]=Newton(f,'cos',0,0,0,5);
```

```
>> fprintf('%14.13f\t%e\n',[x;abs(asin(1/4)-x)/asin(1/4)])
0.0000000000000 1.000000e+00
0.2500000000000 1.060730e-02
0.2526793347751 3.642418e-06
0.2526802551420 4.327879e-13
0.2526802551421 0.000000e+00
```

In sole cinque iterazioni abbiamo ottenuto un risultato 'esatto' nei limiti della precisione di machina.

Esempio 4.4 [Convergenza quadratica e lineare nel metodo di Newton] Consideriamo il problema di calcolare la radice semplice $x = -2$ e la radice doppia $x = 1$ del polinomio

$$p(x) = x^3 - 3x + 2.$$

Se utilizziamo il metodo di Newton l'iterazione avrà l'espressione

$$x_{k+1} = x_k - \frac{x_k^3 - 3x_k + 2}{3x_k^2 - 3}, \quad k \geq 0.$$

In Tabella 4.1 riportiamo i valori ottenuti partendo da $x_0 = 2.4$ dai quali si può vedere la convergenza quadratica del metodo alla radice semplice. In particolare utilizzando la proposizione 4.1 avremo che per k sufficientemente grande

$$|E_{k+1}| \approx \frac{2}{3}|E_k|^2.$$

In modo analogo se ora consideriamo come valore iniziale $x_0 = 1.2$ il metodo risulta convergere alla radice doppia $x = 1$, ma con convergenza lineare. I valori sono mostrati in Tabella 4.2 e confermano quanto predetto dalla proposizione 4.1, ossia

$$|E_{k+1}| \approx \frac{1}{2}|E_k|.$$

k	x_k	$\lvert E_k\rvert = \lvert x_k - x_*\rvert$	$\lvert E_{k+1}\rvert / \lvert E_k\rvert^2$
0	-2.400000	4.000000e-01	0.476190
1	-2.076190	7.619048e-02	0.619469
2	-2.003596	3.596011e-03	0.664278
3	-2.000009	8.589972e-06	0.666661
4	-2.000000	4.919132e-11	

Tabella 4.1 Convergenza quadratica del metodo di Newton ad una radice semplice.

L'esempio precedente evidenzia la necessità di avere tecniche di accelerazione della convergenza del metodo di Newton nel caso di radici multiple. Nel caso in cui x_* sia una radice di ordine n ed il metodo di Newton converga linearmente a tale radice è possibile dimostrare che l'iterazione modificata

$$x_{k+1} = x_k - n\frac{f(x_k)}{f'(x_k)}, \qquad x_0 = c, \tag{4.9}$$

k	x_k	$\lvert E_k\rvert = \lvert x_k - x_*\rvert$	$\lvert E_{k+1}\rvert/\lvert E_k\rvert$
0	1.200000	2.000000e-01	0.515152
1	1.103030	1.030303e-01	0.508165
2	1.052356	5.235642e-02	0.504252
3	1.026401	2.640081e-02	0.502171
4	1.013258	1.325773e-02	0.501098
5	1.006643	6.643418e-03	0.500552
6	1.003325	3.325375e-03	0.500277
7	1.001664	1.663607e-03	

Tabella 4.2 Convergenza lineare del metodo di Newton ad una radice doppia.

produce una sequenza $\{x_k\}_{k\in\mathbf{N}}$ che converge in modo quadratico a x_*. Si lascia come esercizio la verifica di tale proprietà nel caso della radice doppia di $p(x) = x^3-3x+2$. Si noti che tale metodo richiede a priori la conoscenza della molteplicità della radice. In alternativa il metodo può essere applicato per valori crescenti di $n = 1,2,\ldots$ fino ad individuare la molteplicità cercata.

Osservazione 4.2 Una tecnica più efficiente si basa sulla possibilità di accelerare la convergenza di una qualunque successione che converge linearmente. Data una successione $\{x_k\}_{k\in\mathbf{N}}$ convergente linearmente a x_* è possibile definire la nuova successione

$$\hat{x}_k = x_k - \frac{(\Delta x_k)^2}{\Delta^2 x_k} = x_k - \frac{(x_{k+1}-x_k)^2}{x_{k+2}-2x_{k+1}+x_k}. \tag{4.10}$$

Sotto opportune ipotesi si dimostra che la convergenza di $\hat{x}_k$ ad x_* è più rapida della convergenza di x_k a x_* nel senso che

$$\lim_{k\infty} \frac{|\hat{x}_k - x_*|}{|x_k - x_*|} = 0.$$

Tale procedimento è noto con il nome di *metodo Δ^2 di Aitken.* ■

Uno dei problemi associati all'uso del metodo di Newton è dato dal fatto che tale metodo richiede un'espressione esplicita per f e per la sua derivata prima. In molte applicazioni quest'ultima può risultare difficile se non impossibile da calcolare. Ricordando la definizione di derivata prima come limite per del rapporto incrementale (o quoziente di Newton)

$$f'(x_k) = \lim_{h_k\to 0} \frac{f(x_k+h_k)-f(x_k)}{h_k},$$

potremo utilizzare tale quoziente come approssimazione della derivata per valori sufficientemente piccoli di h_k.

Il *metodo delle secanti* si basa sullo scegliere ad ogni passo $h_k = x_k - x_{k-1}$. In questo modo otteniamo il metodo iterativo per $k \geq 1$

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k)-f(x_{k-1})}. \tag{4.11}$$

A differenza del metodo di Newton, il metodo delle secanti richiede due valori iniziale x_0 e x_1 ma utilizza una sola valutazione della funzione f ad ogni iterazione.

Tipicamente si può pensare di utilizzare un valore iniziale x_0 ed un valore $x_1 = x_0 + \delta$ con δ piccola perturbazione. Da un punto di vista geometrico il metodo utilizza come approssimazione della radice l'intersezione della retta per $(x_k, f(x_k))$ e $(x_{k-1}, f(x_{k-1}))$ e l'asse delle ascisse (vedi Figura 4.9).

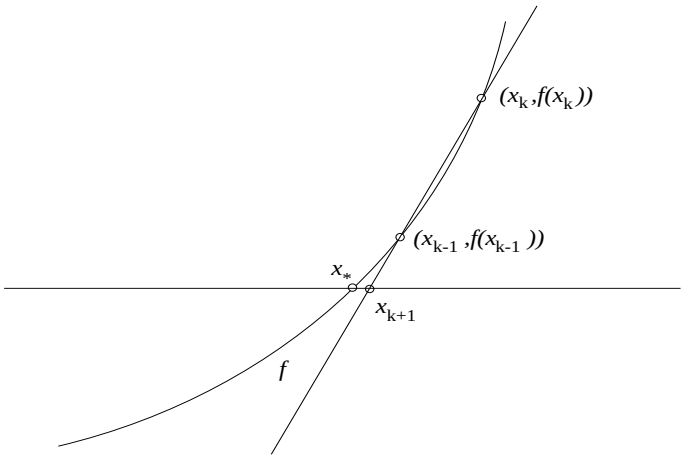

Figura 4.9 Costruzione geometrica del metodo delle secanti.

L'implementazione del metodo segue le linee del metodo di Newton e risulta nella funzione MATLAB

```
function [xs,x]= Secanti(fname,x0,x1,epsilon,delta,maxi)
%
% Sintassi [xs,x]= Secanti(fname,x0,x1,epsilon,delta,maxi)
%          fname:   stringa che contiene il nome della funzione f(x)
%          x0,x1:   definiscono le approssimazioni iniziali della radice
% epsilon, delta:   tolleranze prefissate
%           maxi:   lunghezza massima della successione x(k)
%              x:   successione x(k) convergente a x t.c. f(x)=0
%             xs:   approssimazione della radice
%
x(1)=x0;
x(2)=x1;
fxk0 = feval(fname,x0);
fxk1 = feval(fname,x1);
if abs(fxk1-fxk0) < eps*max(abs(fxk1),abs(fxk0))
   disp('La derivata approssimata f''(x0) e'' nulla')
   return
end
for k=2:maxi-1
   x(k+1)=x(k)-fxk1*(x(k)-x(k-1))/(fxk1-fxk0) ;
   fxk0 = fxk1;
   fxk1=feval(fname,x(k+1));
   if (abs(x(k+1)-x(k))<epsilon)|(abs(fxk1)<delta)
```

```
      break;
   end
   if abs(fxk1-fxk0) < eps*max(abs(fxk1),abs(fxk0))
      fprint('La derivata approssimata f''(x(%d)) e'' nulla',k);
      return
   end
end
xs=x(k+1);
```

Come applicazione della precedente funzione consideriamo il problema test $f(x) = \sin(x) - 1/4$ utilizzando $x_0 = 0$ e $x_1 = 0.1$

```
>> f=fcnchk('sin(x)-1/4');
>> [xs,x]=Secanti(f,0,0.1,0,0,7);
>> fprintf('%14.13f\t%e\n',[x;abs(asin(1/4)-x)/asin(1/4)])
0.0000000000000 1.000000e+00
0.1000000000000 6.042429e-01
0.2504171532909 8.956386e-03
0.2526452093505 1.386962e-04
0.2526802449363 4.039001e-08
0.2526802551420 1.827815e-13
0.2526802551421 0.000000e+00
```

Si può dimostrare che sotto opportune ipotesi la convergenza del metodo ad una radice semplice è di ordine

$$r = \frac{1+\sqrt{5}}{2} \approx 1.6,$$

ossia di poco inferiore a quella del metodo di Newton. Nel caso di radici multiple potranno essere adottate le stesse tecniche di accelerazione viste per il metodo di Newton.

Osservazione 4.3 Da un punto di vista pratico la realizzazione di un metodo di tipo Newton o secanti richiede molta più attenzione rispetto ai metodi di bisezione e falsa posizione. In questo senso i metodi di bisezione e di falsa posizione definiscono metodi numerici più *robusti*. Algoritmi efficienti si possono ottenere combinando in modo opportuno i due diversi approcci. Ad esempio partendo da un intervallo [a,b] contenete la radice x_* si puø' utilizzare il metodo di bisezione per fornire una stima iniziale per il metodo di Newton. In particolare si potrà definire un metodo ibrido nel quale si utilizza il metodo di bisezione in alternativa al metodo di Newton ogni qualvolta il nuovo valore definito dal metodo di Newton risulti esterno all'intervallo.

La scelta di un metodo al posto di un altro dipenderà da diversi fattori, quali la disponibilità di una stima iniziale accurata della radice, il costo del calcolo delle funzioni f e f', il comportamento delle derivate di f, ecc... Non esiste quindi un metodo che a priori possa essere definito migliore di un altro ma dipenderà dal tipo di applicazione considerata. ■

4.3 Problemi di punto fisso

Una caratteristica comune dei metodi esaminati in precedenza è la struttura di tipo iterativo del tipo

$$x_{k+1} = g_k(x_k,x_{k-1},\ldots,x_{k-r+1}), \quad r \geq 1, \quad k \geq 0, \tag{4.12}$$

dove g_k rappresenta la funzione di iterazione del metodo e può dipendere oppure no dal passo k. In generale g_k dipenderà dalla funzione f ed eventualmente dalle sue derivate nei punti $x_k,x_{k-1},\ldots,x_{k-r+1}$. Nella forma (4.12) il metodo è detto ad r punti. Ad esempio il metodo di bisezione utilizza due punti con funzione dipendente dal passo k. Al contrario il metodo di Newton è un metodo ad un punto con funzione di iterazione indipendente dal passo data da

$$g(x_k) = x_k - \frac{f(x_k)}{f'(x_k)}.$$

Analogamente il metodo delle secanti un metodo a due punti con funzione di iterazione data

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

In questa sezione ci occuperemo dei procedimenti iterativi ad un punto

$$x_{k+1} = g(x_k), \qquad k \geq 0, \tag{4.13}$$

con g funzione di iterazione assegnata.

A questo scopo introduciamo la seguente

Definizione 4.3 *Definiamo* punto fisso *di una funzione g assegnata un numero reale x_* tale che*

$$x_* = g(x_*).$$

Osserviamo che qualora la funzione g risulti continua e la successione x_k definita dalla (4.13) convergente ad un valore x_* allora avremo

$$g(x_*) = g\left(\lim_{k\to\infty} x_k\right) = \lim_{k\to\infty} g(x_k) = \lim_{k\to\infty} x_{k+1} = x_*.$$

Di conseguenza x_* sarà un punto fisso per la funzione g. Per questa ragione il processo iterativo (4.13) è chiamato *iterazione di punto fisso.*

Esempio 4.5 [Iterazione di punto fisso]
Consideriamo il problema del calcolo di x tale che

$$x = e^{-x}.$$

Ossia un problema di punto fisso con $g(x) = e^{-x}$. Potremo considerare il procedimento iterativo

$$x_{k+1} = e^{-x_k}, \qquad k \geq 0.$$

A questo scopo definiamo la semplice funzione MATLAB

```
function x=iterapf(fname,x0,N)
%
% Sintassi x=iterapf(fname,x0,N)
% fname : nome funzione di iterazione
%    x0 : valore iniziale
%     N : numero di iterazioni
%
x(1)=x0;
for k=1:N
  gxk=feval(fname,x(k));
  x(k+1)=gxk;
end
```

Utilizzando la precedente funzione con $N = 10$ a partire dalla stima iniziale $x(1) = x_0 = 0.5$ otteniamo la seguente tabella di valori

```
x(1)=0.500000
x(2)=0.606531
x(3)=0.545239
x(4)=0.579703
x(5)=0.560065
x(6)=0.571172
x(7)=0.564863
x(8)=0.568438
x(9)=0.566409
x(10)=0.567560
x(11)=0.566907
```

dalla quale si vede la convergenza della successione al valore $x_* = 0.567143\ldots$ tale che $x_* = e^{x_*}$.

Il risultato principale sulla convergenza di un iterazione di punto fisso è riportato nel seguente teorema.

Teorema 4.2 *Supponiamo che g e g' siano funzioni continue sull'intervallo $[a,b]$ e che $a \leq g(x) \leq b$ per ogni x tale che $a \leq x \leq b$. Dato x_0 nell'intervallo $[a,b]$ e C costante positiva allora se*

$$|g'(x)| \leq C < 1, \tag{4.14}$$

per ogni x nell'intervallo $[a,b]$ allora l'iterazione di punto fisso (4.13) converge all'unico punto fisso x_* di g nell'intervallo $[a,b]$.

DIMOSTRAZIONE. Dalle ipotesi del teorema discende innanzitutto che g ha un unico punto fisso nell'intervallo $[a,b]$. L'esistenza di un punto fisso si verifica immediatamente. Se $a = g(a)$ oppure $b = g(b)$ chiaramente l'esistenza sarà verificata. Altrimenti la funzione $f(x) = x - g(x)$ avrà la proprietà che

$$f(a) = a - g(a) < 0, \qquad f(b) = b - g(b) > 0.$$

Di conseguenza la funzione $f(x)$ ha almeno uno zero interno all'intervallo $[a,b]$ e quindi $g(x)$ ha almeno un punto fisso su tale intervallo.

Supponiamo ora che esistano due punti fissi distinti x_* e $\tilde{x}_*$ interni ad $[a,b]$. Dal teorema del valor medio avremo che esisterà un punto d con $a \leq d \leq b$ tale che

$$g(x_*) - g(\tilde{x}_*) = g'(d)(x_* - \tilde{x}_*).$$

Utilizzando il fatto che x_* e x'_* sono punti fissi otteniamo

$$g'(d) = \frac{g(x_*) - g(\tilde{x}_*)}{x_* - \tilde{x}_*} = \frac{x_* - \tilde{x}_*}{x_* - \tilde{x}_*} = 1.$$

Poiché questo contraddice l'ipotesi che $|g'(x)| < 1$ sull'intervallo $[a,b]$ non sarà possibile l'esistenza di due punti fissi. Quindi g ha un unico punto fisso in $[a,b]$.

Sia ora x_* l'unico punto fisso di g. Dal teorema del valor medio abbiamo che esiste un punto d_0 interno ad $[a,b]$ tale che

$$|x_1 - x_*| = |g(x_0) - g(x_*)| = |g'(d_0)||x_0 - x_*| \leq C|x_0 - x_*|.$$

Ragionando per induzione possiamo dimostrare la disuguaglianza

$$|x_k - x_*| \leq C^k|x_0 - x_*|, \qquad k \geq 1.$$

Il caso $k = 1$ lo abbiamo appena visto. Usando l'ipotesi induttiva

$$|x_{k-1} - x_*| \leq C^{k-1}|x_0 - x_*|$$

ed il teorema del valor medio come nel caso $k = 1$ otteniamo

$$\begin{aligned} |x_k - x_*| &\leq |g(x_{k-1}) - g(x_*)| = |g'(d_k)||x_{k-1} - x_*| \leq C|x_{k-1} - x_*| \\ &\leq CC^{k-1}|x_0 - x_*| \leq C^{k-1}|x_0 - x_*|, \end{aligned}$$

con d_k punto opportuno interno ad $[a,b]$.

Infine, essendo $0 < C < 1$ avremo

$$\lim_{k\to\infty} |x_k - x_*| \leq \lim_{k\to\infty} C^k|x_0 - x_*| = 0,$$

ossia la convergenza della successione x_k al punto fisso x_* ■

Osserviamo che come conseguenza della precedente dimostrazione abbiamo ottenuto le seguenti stime sull'errore commesso al generico passo k

$$|x_k - x_*| \leq C^k|x_0 - x_*|, \qquad k \geq 1 \tag{4.15}$$

$$|x_k - x_*| \leq \frac{C^k}{1-C}|x_1 - x_0|, \qquad k \geq 1. \tag{4.16}$$

Osservazione 4.4 Se nelle ipotesi del precedente teorema assumiamo

$$|g'(x)| > 1,$$

per ogni x nell'intervallo $[a,b]$ allora è possibile dimostrare, utilizzando argomenti analoghi a quelli del precedente teorema, che l'iterazione di punto fisso (4.13) per $x_0 \neq x_*$ risulterà non convergere al punto fisso x_*, ossia sarà localmente divergente. ■

Esempio 4.6 [Convergenza e divergenza dell'iterazione di punto fisso] Si consideri il problema del calcolo dei punti fissi della funzione polinomiale

$$g(x) = 1 + x - \frac{x^2}{4}.$$

In questo caso la derivata prima risulta

$$g'(x) = 1 - \frac{x}{2}.$$

I punti fissi possono essere calcolati direttamente risolvendo un equazione di secondo grado e sono $x = -2$ ed $x = 2$.

Applicando la funzione MATLAB `iterpf` alla funzione g per $N = 3$ a partire dal valore iniziale $x_0 = 1.05$ otteniamo

```
x(1)=1.050000
x(2)=1.774375
x(3)=1.987273
x(4)=1.999960
```

Quindi la convergenza della successione come previsto dal teorema 4.2 essendo $|g'(x)| \leq 1/2$ in [1,3].

Se ora cambiamo il valore iniziale e scegliamo un valore vicino alla radice -2, ad esempio -2.05 otteniamo

```
x(1)=-2.050000
x(2)=-2.100625
x(3)=-2.203781
x(4)=-2.417944
```

Quindi la successione risulta divergente. Infatti in questo secondo caso $|g'(x)| > 3/2$ su $[-3,1]$. I due differenti comportamenti sono illustrati graficamente in Figura 4.10.

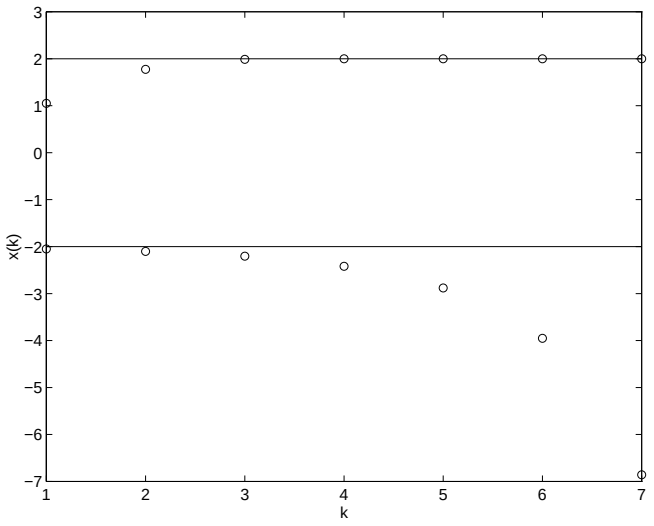

Figura 4.10 Comportamento dell'iterazione di punto fisso per $g(x) = 1 + x - x^2/4$.

Osserviamo infine che se la funzione di iterazione g è sufficientemente regolare vale l'espansione di Taylor in un intorno $I = [x_* - \lambda, x_* + \lambda]$ di x_*

$$g(x) = g(x_*) + (x - x_*)g'(x_*) + \frac{(x - x_*)^2}{2!}g''(x_*) + \ldots + \frac{(x - x_*)^n}{n!}g^{(n)}(\xi), \qquad \xi \in I.$$

Se x_* è un punto fisso per g dalla precedente relazione per $x = x_k$ otteniamo

$$x_{k+1} - x_* = (x_k - x_*)g'(x_*) + \frac{(x_k - x_*)^2}{2!}g''(x_*) + \ldots + \frac{(x_k - x_*)^n}{n!}g^{(n)}(\xi).$$

Di conseguenza se $|g'(x_*)| = C < 1$ con $C \neq 0$ avremo

$$\frac{e_{k+1}}{e_k} = \frac{x_{k+1} - x_*}{x_k - x_*} = g'(x_*) + \frac{(x_k - x_*)}{2!} g''(x_*) + \ldots + \frac{(x_k - x_*)^{n-1}}{n!} g^{(n)}(\xi), \quad (4.17)$$

e quindi la convergenza di x_k ad x_* risulterà lineare essendo

$$\lim_{k \to \infty} \frac{|e_{k+1}|}{|e_k|} = |g'(x_*)| = C.$$

Al contrario se $g'(x_*) = 0, g''(x_*) = 0, \ldots, g^{(n-1)}(x_*) = 0$ ma $g^{(n)}(x) \neq 0$ per $x \in I$ allora

$$\frac{e_{k+1}}{e_k^{n-1}} = \frac{x_{k+1} - x_*}{(x_k - x_*)^{n-1}} = \frac{g^{(n)}}{n!}(\xi), \quad (4.18)$$

e la convergenza del metodo sarà di ordine $n - 1$ in quanto

$$\lim_{k \to \infty} \frac{|e_{k+1}|}{|e_k|^{n-1}} = \frac{|g^n(x_*)|}{n!}.$$

Osservazione 4.5 Nel contesto delle iterazioni di punto fisso l'applicazione dell'accelerazione Δ^2 di Aitken è nota con il nome di *metodo di Steffensen* ed origina il procedimento iterativo

$$x_{k+1} = x_k - \frac{(g(x_k) - x_k)^2}{g(g(x_k)) - 2g(x_k) + x_k}. \quad (4.19)$$

■

4.4 Le funzioni MATLAB `fzero` e `roots`

I metodi per la ricerca di zeri di funzioni discussi finora sono basati su semplici algoritmi che possiedono doti di robustezza (bisezione e falsa posizione) oppure rapidità di convergenza (Newton e secanti). La realizzazione di un algoritmo efficiente dovrà cercare di combinare i pregi dell'uno e dell'altro metodo in un codice di tipo ibrido. La funzione MATLAB `fzero` è un esempio di questo tipo in quanto al suo interno utilizza in modo sofisticato il metodo di bisezione, il metodo delle secanti e interpolazione quadratica inversa.

L'interpolazione quadratica inversa è basata sulla scelta di tre coppie di valori $(x_1, (f(x_1))$, $(x_2, f(x_2))$ e $(x_3, f(x_3))$ con $f(x_1)$, $f(x_2)$ ed $f(x_3)$ distinti utilizzati per costruire un polinomio di tipo quadratico interpolante l'inversa $f^{-1}(y)$ nella forma

$$x = \alpha_1 y^2 + \alpha_2 y + \alpha_3.$$

Si utilizzerà poi l'intersezione di tale polinomio con l'asse delle x, ossia il valore $x_4 = \alpha_3$, come approssimazione della radice cercata. Il procedimento potrà poi essere iterato scartando il valore tra $\{x_1, x_2, x_3\}$ più distante da x_4.

Tale metodo può essere considerato come la naturale estensione del metodo delle secanti utilizzando polinomi di secondo grado al posto di un interpolazione lineare basata su rette.

La sintassi base della funzione è

Radice = `fzero`(*Stringa Nome Funzione*, *Valore Iniziale*, *Tolleranza*).

Se il parametro opzionale *Tolleranza* omesso allora MATLAB restituisce il valore approssimato della radice calcolato con un margine di errore `eps`$|$*Radice*$|$.

Esempio 4.7 [Uso di `fzero` nel problema della sfera galleggiante]
Il problema della determinazione dell'altezza alla quale una sfera galleggiante con densità ϱ_s risulta immersa nell'acqua visto nell'Esempio 4.1 comporta la risoluzione dell'equazione di terzo grado

$$p(d) = d^3 - 15d^2 + 500s = 0,$$

dove $s = \varrho_s/\varrho_a$ è il rapporto tra la densità della sfera e quella dell'acqua.

La funzione che dovremo passare ad `fzero` è una funzione di singola variabile scalare, quindi non potremo utilizzare una funzione MATLAB nella forma

```
function z=polsfera(d,s)
z=d^3-15*d^2+500*s;
```

in quanto tale funzione utilizza due parametri. Possiamo ovviare a questo inconveniente definendo s come variabile globale ed utilizzando la funzione

```
function z=polsfera(d)
global s
z=d^3-15*d^2+500*s;
```

Se utilizziamo come stima iniziale della radice il valore $d = 6$ otteniamo in `format long`

```
>> global s
>> s=0.65;
>> fzero('polsfera',6);
Zero found in the interval: [5.8303, 6.1697].
ans =
      6.01389697307565
```

Riportiamo infine una tabella di valori per s e d

```
-------------------
s        d
-------------------
0.50     5.00000
0.55     5.33383
0.60     5.67069
0.65     6.01390
0.70     6.36743
0.75     6.73648
0.80     7.12859
0.85     7.55598
0.90     8.04200
```

ottenuta con il seguente script file

```
% tabellasfera.m
%
global s
x0=6;
disp('-----------------------');
```

```
fprintf('s \t s\n');
disp('-----------------------');
for s=0.5:0.5:0.8
   d=fzero('polsfera',x0);
   fprintf('%2.1f \t %5.4f\n',s,d);
end
```

■

Chiaramente tutte le tecniche viste finora possono essere applicate a funzioni polinomiali. In generale tali funzioni comportano delle difficoltà aggiuntive dovute alla presenza di radici multiple, oppure radici molto vicine, o radici complesse. In particolare esistono formule esplicite per il calcolo di radici di polinomi fino al grado 4. Da un punto di vista numerico abbiamo visto nel capitolo 1 l'importanza dell'algoritmo utilizzato per il calcolo delle radici di un polinomio di secondo grado

$$ax^2 + bx + c = 0,$$

tramite la formula esplicita

$$x_{\pm} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Per ridurre gli errori dovuti all'esecuzione di operazioni in aritmetica finita converrà infatti utilizzare la procedura di calcolo in due passi

$$q = -\frac{1}{2}\left(b + \text{sign}(b)\sqrt{b^2 - 4ac}\right),$$

$$x_+ = \frac{q}{a}, \qquad x_- = \frac{c}{q}.$$

Esistono differenti metodi numerici disegnati appositamente per il calcolo di radici di polinomi, quali il metodo di Bairstow e il metodo di Laguerre. Riferiamo ad esempio a [9] per ulteriori approfondimenti sugli algoritmi suddetti.

La funzione MATLAB `roots` consente di determinare le radici di un polinomio scritto nella forma

$$c_1 x^n + c_2 x^{n-1} + \ldots + c_n x + c_{n+1} = 0.$$

La tecnica utilizzata si basa sul calcolo degli autovalori della *matrice companion* associata a tale polinomio ed il problema del calcolo delle radici è ricondotto in questo modo ad un problema di calcolo di autovalori (si veda il Capitolo 2).

Ad esempio la matrice companion associata al polinomio di grado quattro

$$c_1\lambda^4 + c_2\lambda^3 + c_3\lambda^2 + c_4\lambda + c_5 = 0,$$

sarà

$$A = \begin{pmatrix} -c_4/c_5 & -c_3/c_5 & -c_2/c_5 & -c_1/c_5 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

La sintassi della funzione è

$$\textit{Vettore Radici} = \texttt{roots}(\textit{Vettore Coefficienti}),$$

dove *Vettore Coefficienti*= $(c_1,c_2,\ldots,c_n,c_{n+1})$ ed il calcolo di tutte le radici avviene in una singola chiamata alla funzione. Non esiste in questo caso la possibilità di modificare la tolleranza nella precisione con la quale i calcoli sono effettuati.

Ad esempio nel caso del polinomio cubico che interviene nel problema della sfera galleggiante potremo scrivere

```
>> s=0.65;
>> p=[1 -15 0 500*s];
>> r=roots(p)
>> r
ans =
  13.10867741620371
   6.01389697307565
  -4.12257438927937
```

dove il risultato è stato calcolato in `format long` per evidenziare come coincida con quello trovato tramite la funzione `fzero` nei limiti della precisione di macchina. Abbiamo quindi oltre alla radice cercata le due altre radici del polinomio che chiaramente sono prive di senso fisico nell'ambito della applicazione considerata. Si noti come nella definizione del vettore dei coefficienti p il coefficiente del termine di primo grado deve comparire come zero e non può essere omesso.

La funzione potrà essere utilizzata anche nel caso in cui il polinomio abbia radici e/o coefficienti complessi. Ad esempio

```
>> roots([1 3 5])
ans =
  -1.5000 + 1.6583i
  -1.5000 - 1.6583i
>> roots([1+2i -1-2i 5])
ans =
   1.3325 + 1.2012i
  -0.3325 - 1.2012i
```

4.5 Minimi e massimi di funzioni

In molte applicazioni si richiede la risoluzione di un problema che richiede il calcolo del minimo oppure del massimo locale di una funzione f assegnata su un certo insieme. Ricordiamo a questo fine la seguente definizione

Definizione 4.4 *La funzione f è detta avere un* minimo locale *(massimo locale) in $x = x_*$ se esiste un intervallo I contenete x_* tale che $f(x_*) \leq f(x)$ $(f(x_*) \geq f(x))$ per tutti gli x appartenenti ad I.*

In particolare se consideriamo una funzione f definita su un intervallo $[a,b]$ che ha un minimo oppure un massimo locale x_* interno ad $[a,b]$, allora se f è differenziabile in x_* avremo

$$f'(x_*) = 0.$$

Il risultato precedente qualora sia nota l'espressione della derivata prima di f in un intervallo, consente di applicare i metodi sviluppati nei paragrafi precedenti al calcolo dei punti in cui f ha minimi e massimi locali.

Esempio 4.8 [Massimi e minimi locali di un polinomio]
Se consideriamo la funzione polinomiale

$$f(x) = x^3 + x^2 - x + 1,$$

sull'intervallo $[-2,2]$, la determinazione dei minimi e massimi locali della funzione può essere ricondotta la calcolo delle radici della derivata prima

$$f'(x) = 3x^2 + 2x - 1.$$

In questo caso senza utilizzare procedimenti numerici vediamo subito che le radici sono $x = 1/3$ ed $x = -1$. In questo caso data la continuità della funzione, per determinare se tali radici rappresentano dei massimi o dei minimi potremo valutare il segno della derivata prima a sinistra e a destra delle radici

$$f'(-2) = 7 > 0, \quad f'(0) = -1 < 0, \quad f'(2) = 15 > 0,$$

da cui discende che $x = -1$ è un massimo locale ed $x = 1/3$ un minimo locale. In alternativa potremo utilizzare il test sul segno della derivata seconda $f''(x) = 6x+2$

$$f''(1/3) = 4 > 0, \qquad f''(-1) = -4 < 0,$$

che porta allo stesso risultato. ■

In molti casi la possibilità di calcolare la derivata prima rappresenta una forte restrizione ed in particolare non consente il calcolo di minimi e massimi locali nel caso in cui la funzione non sia differenziabile con continuità in un certo intervallo.

Consideriamo quindi il problema del calcolo del minimo di una funzione f su un insieme S

$$\min_{x \in S} f(x), \tag{4.20}$$

con f ed S assegnati. Il corrispondente problema di massimo potrà essere posto nella stessa forma semplicemente osservando che il massimo di f è il minimo di $-f$. In questa forma il problema rappresenta un *problema di ottimizzazione* monodimensionale. Abbiamo visto nel caso della ricerca di zeri di funzioni come qualora le valutazioni della funzione non siano costose si possa adottare una semplice strategia di visualizzazione grafica per avere un'idea della localizzazione delle radici. La stessa strategia potrà essere adottata per individuare i valori massimi e minimi di una funzione.

Esempio 4.9 [Distanza minima tra Mercurio e la Terra]
In prima approssimazione possiamo assumere che le traiettorie dei pianeti Mercurio e Terra (M e T) siano governate dalla seguenti equazioni

$$\begin{aligned}
x_T(t) &= a_T + b_T \cos\left(\frac{2\pi t}{T_T}\right), \\
y_T(t) &= c_T \sin\left(\frac{2\pi t}{T_T}\right), \\
x_M(t) &= a_M + b_M \cos\left(\frac{2\pi t}{T_M}\right), \\
y_M(t) &= c_M \sin\left(\frac{2\pi t}{T_M}\right),
\end{aligned}$$

dove $a_T \approx -2.5$, $b_T \approx c_T \approx 150$, $T_T \approx 365$, $a_M \approx -12$, $b_M \approx C_M \approx 57$, $T_M \approx 88$ e la variabile temporale t è misurata in giorni.

La funzione che descrive la distanza tra i due pianeti sarà quindi

$$r(t) = \sqrt{(x_T(t) - x_M(t))^2 + (y_T(t) - y_M(t))^2}.$$

Supponiamo di volere calcolare la distanza minima tra Mercurio e la Terra nei prossimi tre anni, ossia approssimativamente per $0 \leq t \leq 1095$.

A questo scopo potremo realizzare il grafico della funzione $r(t)$ nell'intervallo $[L,R]$ tramite la funzione MATLAB

```
function plotdistMT(L,R,N)
%
% Sintassi plotdistMT(L,R,N)
% realizza il grafico della distanza
% approssimata tra Mercurio e la Terra usando
% N punti nell'intervallo [L,R]
% Tramite il mouse e' possibile identificare
% i valori di massimi e minimi relativi
%
aT=-2.5; bT= 150; cT=150; TT=365;
aM=-12; bM=57; cM=57; TM=88;
t=linspace(L,R,N);
xT=aT+bT*cos(2*pi*t/TT);
yT=cT*sin(2*pi*t/TT);
xM=aM+bM*cos(2*pi*t/TM);
yM=cM*sin(2*pi*t/TM);
r=sqrt((xT-xM).^2+(yT-yM).^2);
tasto=0;
while (tasto~=3)
  plot(t,r);
  xlabel('Giorni');
  ylabel('Distanza');
  [x,y,tasto]=ginput(1);
  if tasto ~= 3
    text(x,y,sprintf('(x=%3.1f y=%3.1f)',x,y));
    pause(1);
  end
end
```

L'istruzione

$$[X, Y, Tasto] = \texttt{ginput}(N),$$

restituisce i vettori X e Y contenenti le coordinate di N punti ottenuti posizionando il cursore del mouse sul grafico e premendo un tasto del mouse. Il vettore di interi *Tasto* conterrà i valori 1,2 o 3 a seconda del tasto del mouse premuto. L'istruzione

$$\texttt{text}(X, Y, Stringa),$$

visualizza nella posizione di coordinate X, Y il contenuto della stringa di caratteri *Stringa*. Potremo in questo modo interattivamente individuare la posizione del minimo. Il grafico ottenuto con 200 punti su tutto l'intervallo è riportato in figura 4.11 (sinistra).

Il minimo sembra sia assunto nell'intervallo [900,950]. Possiamo quindi utilizzare nuovamente la funzione per avere un'indicazione più precisa realizzandone il

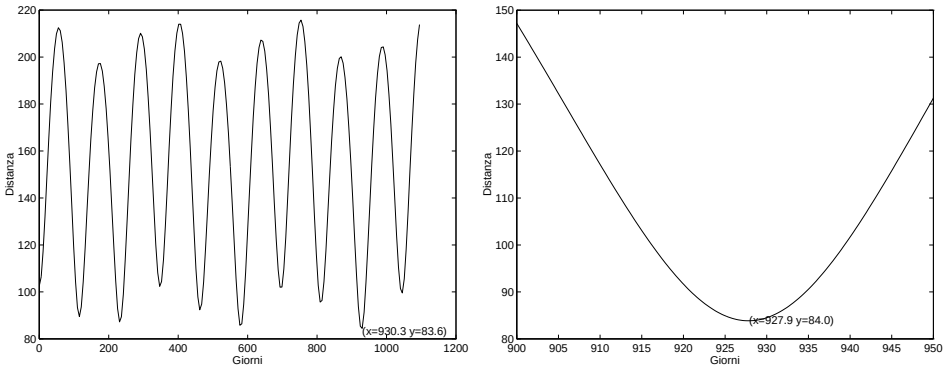

Figura 4.11 Distanza tra Mercurio e la Terra.

grafico in tale intervallo (vedi figura 4.11 (destra)) ed esplorando con il mouse per identificare il valore approssimato del minimo. Indicativamente quindi potremo collocare il minimo al giorno 928 con un valore di circa 84. ■

4.5.1 Metodo della sezione aurea

Chiaramente il metodo precedente può essere utilizzato qualora il costo della valutazione della funzione non sia elevato. Altrimenti per ridurre il numero di valutazioni della funzione sarà importante avere una buona strategia per determinare dove valutare la funzione $f(x)$.

Uno dei metodi più efficienti è il metodo della sezione aurea che deve il proprio nome al modo particolare di scegliere i punti in cui valutare la funzione.

Consideriamo l'intervallo [0,1]. Sia $0.5 < r < 1$ un punto interno a tale intervallo. Allora $0 < 1 - r < 0.5$ e potremo suddividere l'intervallo in tre sottointervalli $[0,1-r]$, $[1-r,r]$ e $[r,1]$. A questo punto un processo decisionale come nell'algoritmo di bisezione è utilizzato per la scelta del nuovo sottointervallo che sarà $[0,r]$ oppure $[1-r,1]$. Questo sottointervallo sarà a sua volta suddiviso in tre sottointervalli utilizzando gli stessi rapporti usati nell'intervallo [0,1]. Vogliamo scegliere il punto iniziale r in modo tale che il punto utilizzato per la vecchia suddivisione venga mantenuto nella nuova (vedi figura 4.12).

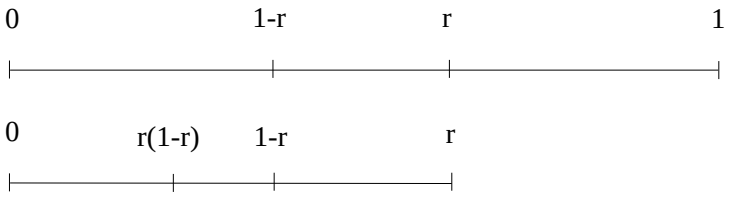

Figura 4.12 Divisione in sottointervalli nel metodo della sezione aurea.

Questo implica che il rapporto $(1-r) : r$ deve essere uguale al rapporto $r : 1$. Quindi r deve soddisfare l'equazione $1 - r = r^2$ ossia

$$r^2 + r - 1 = 0.$$

La soluzione che soddisfa $0.5 < r < 1$ è

$$r = \frac{-1+\sqrt{5}}{2} \approx 0.618,$$

nota come *rapporto aureo* o *sezione aurea*.

La precedente strategia nella scelta dei punti consente di minimizzare il numero di valutazioni della funzione nella ricerca di un minimo locale. La funzione $f(x)$ dovrà però soddisfare una condizione speciale tale da garantire l'esistenza di un minimo nell'intervallo.

Definizione 4.5 *La funzione $f(x)$ è detta* unimodale *sull'intervallo $I = [a,b]$ se esiste unico un punto x_* nell'intervallo I tale che*

1) *$f(x)$ è strettamente decrescente su $[a,x_*]$*

2) *$f(x)$ è strettamente crescente su $[x_*,b]$.*

In altre parole la funzione $f(x)$ ha un unico punto di minimo nell'intervallo $[a,b]$.

Data una funzione unimodale $f(x)$ nel metodo della sezione aurea si scelgono due punti interni all'intervallo $[a,b]$ definiti da

$$c = a + (1-r)(b-a), \qquad d = a + r(b-a),$$

dove r è il precedente rapporto aureo. Avremo quindi $a < c < d < b$. Se $f(c) \leq f(d)$ allora dall'unimodalità segue che f è crescente in $[d,b]$ e quindi x_* è in $[a,d]$. In questo caso scegliamo come nuovo sottointervallo l'intervallo $[a,d]$. Se invece $f(c) > f(d)$ allora f è decrescente in $[a,c]$ e quindi x_* è in $[c,b]$. In questo caso il nuovo sottointervallo sarà $[c,b]$. Potremo quindi ripetere lo stesso ragionamento sul sottointervallo così ottenuto.

L'algoritmo può essere organizzato nella seguente funzione MATLAB

```
function [xs,c,d]=sezaurea(fname,a,b)
%
% Sintassi [xs,c,d]=sezaurea(fname,a,b)
%
% fname: nome funzione
%   a,b: estremi intervallo
%    xs: valore approssimato del minimo
%   c,d: vettori contenenti i valori dei punti c e d
%
r=(sqrt(5)-1)/2;
c(1)=a+(1-r)*(b-a);
d(1)=a+r*(b-a);
fc=feval(fname,c(1));
fd=feval(fname,d(1));
k=1;
while (d(k)-c(k)) > eps*max(abs(c(k)),abs(d(k)))
   if fc <= fd
```

```
      z = a+(1-r)*(d(k)-a);
      b=d(k);
      d(k+1)=c(k);
      c(k+1)=z;
      fd=fc;
      fc=feval(fname,z);
   else
      z = c(k)+r*(b-c(k));
      a=c(k);
      c(k+1)=d(k);
      d(k+1)=z;
      fc=fd;
      fd=feval(fname,z);
   end
   k=k+1;
end
xs=(c(k)+d(k))/2;
```

Il criterio di arresto utilizzato è basato sul controllo dell'ampiezza dell'intervallo $[c,d]$.

Esempio 4.10 [Minimo di una funzione unimodale]
Consideriamo il problema del calcolo del minimo della funzione unimodale

$$f(x) = x^2 - \sin(x),$$

sull'intervallo [0,1].

Possiamo utilizzare la precedente funzione MATLAB. In tabella 4.3 riportiamo i risultati ottenuti nelle prime 10 iterazioni dove si vede la convergenza al valore $0.45\ldots$.

k	c_k	d_k
0	0.38196601125011	0.61803398874989
1	0.23606797749979	0.38196601125011
2	0.38196601125011	0.47213595499958
3	0.47213595499958	0.52786404500042
4	0.43769410125095	0.47213595499958
5	0.41640786499874	0.43769410125095
6	0.43769410125095	0.45084971874737
7	0.45084971874737	0.45898033750315
8	0.44582472000673	0.45084971874737
9	0.45084971874737	0.45395533876251

Tabella 4.3 Convergenza del metodo della sezione aurea per $f(x) = x^2 - \sin(x) = 0$.

Se utilizziamo la differenziabilità della funzione

$$f'(x) = 2x - \cos(x),$$

ed applichiamo il metodo delle secanti otteniamo il risultato riportato in tabella 4.4 che fornisce l'approssimazione del punto di minimo cercato nei limiti della precisione di macchina $x_* = 0.45018361129487$.

■

k	x_k
0	0.00000000000000
1	1.00000000000000
2	0.40655402588120
3	0.44651232728937
4	0.45021370870598
5	0.45018359084167
6	0.45018361129476
7	0.45018361129487

Tabella 4.4 Convergenza del metodo delle secanti per $f'(x) = 2x - \cos(x) = 0$.

Nonostante nel precedente esempio il metodo della sezione aurea risulti più lento a differenza del metodo delle secanti può essere applicato anche nel caso in cui la funzione non sia differenziabile. L'ipotesi di unimodalità è essenziale per la convergenza del metodo. In tale ipotesi si può dimostrare che la convergenza del metodo è lineare come per l'algoritmo di bisezione.

4.6 La funzione MATLAB `fmin`

Analogamente alla funzione MATLAB `fzero`, la funzione MATLAB `fmin` combina le caratteristiche di robustezza dell'algoritmo della sezione aurea con la velocità di convergenza di una tecnica di tipo Newton. Per ottenere questo senza dovere affrontare il problema del calcolo della derivata prima, tale funzione combina il metodo della sezione aurea con un metodo basato su interpolazione parabolica che utilizza esclusivamente valutazioni della f. La parabola

$$y = c_1 x^2 + c_2 x + c_3,$$

costruita, ad esempio, utilizzando tre punti $(x_0,f(x_0))$, $(x_1,f(x_1))$ e $(x_3,f(x_3))$ forniti dal metodo della sezione aurea. Si utilizza poi il minimo di tale parabola

$$x_4 = -\frac{c_2}{2c_1},$$

come approssimazione del minimo della funzione f. Il metodo potrà poi essere iterato scegliendo i due valori tra $\{x_1,x_2,x_3\}$ che sono più prossimi a x_4.

La sintassi della funzione è

Minimo=`fmin`(*Stringa Nome Funzione*, *Intervallo*, *Opzioni*).

Se i parametri opzionali *Opzioni* sono omessi allora MATLAB restituisce il valore approssimato del minimo calcolato con una tolleranza di 10^{-4}. Si veda `help fmin` e l'esempio successivo per una descrizione più dettagliata dell'uso del vettore *Opzioni* contenente le diverse possibili opzioni.

Esempio 4.11 [Uso di `fmin` nel problema Mercurio-Terra]
Possiamo utilizzare la funzione `fmin` per determinare la distanza minima tra Mercurio e la Terra secondo le equazioni introdotte nell'esempio 4.9.

Definendo la funzione

```
function r=distMT(t)
%
% Sintassi r=distMT(t)
% restituisce la distanza
% approssimata tra Mercurio e la Terra
% al tempo t
%
aT=-2.5; bT= 150; cT=150; TT=365;
aM=-12; bM=57; cM=57; TM=88;
xT=aT+bT*cos(2*pi*t/TT);
yT=cT*sin(2*pi*t/TT);
xM=aM+bM*cos(2*pi*t/TM);
yM=cM*sin(2*pi*t/TM);
r=sqrt((xT-xM).^2+(yT-yM).^2);
```

potremo scrivere

```
>> tmin=fmin('distMT',900,950)
tmin =
  927.8130
>> distMT(tmin)
ans =
   83.8537
```

Se utilizziamo il vettore di opzioni `[1 1.e-8]` faremo si che vengano visualizzati in uscita i risultati parziali delle iterazioni e fisseremo la nuova tolleranza a 10^{-8}

```
>> tmin=fmin('distMT',900,950,[1 1.e-08])
 Func evals     x          f(x)         Procedure
    1        919.098      93.4325       initial
    2        930.902      85.1433       golden
    3        938.197      97.2195       golden
    4        927.844      83.8539       parabolic
    5        927.732      83.8546       parabolic
    6        927.813      83.8537       parabolic
    7        927.813      83.8537       parabolic
    8        927.813      83.8537       parabolic
    9        927.813      83.8537       parabolic

tmin =
  927.8130
```

■

4.7 Esercizi

Esercizio 4.1 Per ogni funzione determinare un intervallo $[a,b]$ tale che $f(a)f(b) < 0$ ed applicare l'algoritmo di bisezione per determinare uno zero della funzione.

(a) $f(x) = e^x - 2 - x$.

(b) $f(x) = \cos(x) + 1 - x$.

(c) $f(x) = \ln(x) - 5 + x$.

(d) $f(x) = x^2 - 10x + 23$.

Esercizio 4.2 Si denotino con $[a_0,b_0]$, $[a_1,b_1]$, ..., $[a_k,b_k]$ gli intervalli successivi che intervengono nell'algoritmo di bisezione. Dimostrare che indicato con $c_k = (a_k + b_k)/2$ il punto di mezzo di ogni intervallo si ha

$$\lim_{k\to\infty} a_k = \lim_{k\to\infty} c_k = \lim_{k\to\infty} b_k.$$

Esercizio 4.3 Date le seguenti funzioni e valori iniziali per il metodo di Newton determinare il procedimento iterativo che caratterizza il metodo di Newton e dire se la successione generata dal metodo converge linearmente o quadraticamente.

(a) $f(x) = x^2 - x + 2, \qquad x_0 = -1.5.$

(b) $f(x) = x^2 - x - 3, \qquad x_0 = 1.6.$

(c) $f(x) = (x-2)^4, \qquad x_0 = 2.1.$

(d) $f(x) = x^3 - 3x - 2, \qquad x_0 = 2.1.$

Esercizio 4.4 Analizzare il comportamento dell'iterazione di punto fisso nel caso della funzione

$$g(x) = -4 + 4x - \frac{1}{2}x^2,$$

e mostrare che $x_* = 2$ e $x_* = 4$ sono punti fissi.

Esercizio 4.5 Stabilire se l'iterazione di punto fisso converge nel caso delle seguenti funzioni e rispettivi valori iniziali

(a) $g(x) = \sqrt{6+x}, \qquad x_0 = 7,$

(b) $g(x) = 1 + 2/x, \qquad x_0 = 4,$

(c) $g(x) = x^2/3, \qquad x_0 = 3.5,$

(d) $g(x) = -x^2 + 2x + 2, \qquad x_0 = 2.5.$

Esercizio 4.6 Utilizzare le relazioni (4.17) e (4.18) nel caso del metodo di Newton per dimostrare il risultato enunciato dalla Proposizione 4.1.

Esercizio 4.7 [Metodo di Halley] Il metodo di Halley è caratterizzato dal procedimento iterativo

$$x_{k+1} = x_k - \frac{f(x)}{f'(x)}\left(1 - \frac{f(x)f''(x)}{2(f'(x))^2}\right)^{-1},$$

dove il termine tra parentesi può essere interpretato come una modifica del metodo di Newton. Se convergente, tale metodo ha una convergenza cubica ($p = 3$) nel caso di radici semplici di $f(x)$. Si realizzi una funzione MATLAB chiamata `Halley(fname,fname1,fname2,x0,epsilon,delta,maxi)`, dove `fanme2` è una stringa che contiene il nome della derivata seconda, che implementa il procedimento iterativo che caratterizza il metodo di Halley. Si applichi poi tale funzione al calcolo di

(a) $f(x) = x^2 - 5, \qquad x_0 = 2,$

(b) $f(x) = x^3 - 3x + 2, \qquad x_0 = -2.4.$

Esercizio 4.8 Utilizzare come base di partenza la funzione MATLAB per l'iterazione di punto fisso `iterpf(fname,x0,N)` per realizzare una funzione MATLAB chiamata `Steffensen(fname,x0,N)` che implementa il metodo di Steffensen (4.19).

Esercizio 4.9 Modificare la funzione MATLAB `iterpf(fname,x0,N)` in modo tale da arrestare l'iterazione in base a due tolleranze prefissate ε e δ utilizzando i criteri di arresto $|x_{k+1} - x_k| < \varepsilon$ e $|f(x_k)| < \delta$.

Esercizio 4.10 Utilizzare come base di partenza la funzione MATLAB relativa al metodo delle secanti `Secanti(fname,x0,x1,epsilon,delta,maxi)` per realizzare la funzione `ASecanti(fname,x0,x1,epsilon,delta,maxi)` che implementa la corrispondente iterazione Δ^2 di Aitken (4.10).

Esercizio 4.11 Confrontare il comportamento delle funzioni `iterpf` e `Steffensen`, e delle funzioni `Secanti` e `ASecanti` per calcolare le radici di ordine n delle seguenti funzioni

(a) $f(x) = (x-2)^5, \quad n = 5, \quad x_0 = 1,$

(b) $f(x) = \sin(x^3), \quad n = 3, \quad x_0 = 1,$

(c) $f(x) = (x-1)\ln(x), \quad n = 2, \quad x_0 = 2.$

Esercizio 4.12 Il metodo di Aitken può essere utilizzato per velocizzare la convergenza di una serie. Se indichiamo con

$$S_k = \sum_{j=1}^{n} A_j,$$

mostrare che l'accelerazione di Aitken fornisce

$$\tilde{S}_k = S_k + \frac{A_{k+1}^2}{A_{k+1} - A_{k+2}}.$$

Esercizio 4.13 [Metodo di Muller] Il metodo di Muller è una generalizzazione del metodo delle secanti ed utilizza tre valori iniziali distinti x_0, x_1 ed x_2 per costruire la parabola che passa per $(x_0,f(x_0))$, $(x_1,f(x_1))$ e $(x_2,f(x_2))$. Le radici $x_3^{(1)}$ e $x_3^{(2)}$ di tale parabola potranno essere calcolate utilizzando la formula per il calcolo di radici di polinomi di grado due. Se assumiamo inizialmente x_2 come stima più accurata della radice, per ragioni di stabilità delle due radici ottenute risolvendo l'equazione di secondo grado converrà tenere quella che ha distanza minore da x_2 e quindi porre

$$x_3 = \min\{|x_3^{(1)} - x_2|, |x_3^{(2)} - x_2|\}.$$

Il procedimento sarà poi iterato scartando il punto tra $\{x_0,x_1,x_2\}$ più distante da x_3. Nel caso la ricerca di radici sia limitata a radici reali una semplice strategia per evitare radici complesse è basata sul porre uguale a zero la parte immaginaria della radice calcolata ad ogni iterazione. Si descriva una possibile implementazione del metodo e si realizzi una funzione MATLAB che la attua chiamata `Muller(fname,x0,x1,x2,epsilon,delta,maxi)`.

Esercizio 4.14 Si supponga che le equazioni del moto di un proiettile siano

$$y(t) = 9600(1 - e^{-t/15}) - 480t, \qquad x(t) = 2400(1 - e^{-t/15}),$$

determinare il tempo che trascorre prima dell'impatto con il terreno ($(x,y) = (0,0)$) e la gittata con una accuratezza di 10 cifre decimali.

Esercizio 4.15 Quale è l'area del più grande triangolo i cui vertici appartengono all'ellisse di equazione

$$\frac{1}{4}x^2 + \frac{1}{9}y^2 = 1.$$

Risolvere il problema utilizzando le funzioni `sezaurea` e `fmin`.

Esercizio 4.16 Determinare il punto (i punti) sulla circonferenza di equazione

$$x^2 + y^2 = 25,$$

più distante (distanti) dalla corda AB con $A = (3,4)$ e $B = (-1,\sqrt{24})$. Risolvere il problema utilizzando le funzioni `sezaurea` e `fmin`.