2

Equazioni non lineari

Il calcolo degli *zeri* di una funzione f reale di variabile reale o, equivalentemente, delle *radici* dell'equazione $f(x) = 0$, è un problema assai ricorrente nel Calcolo Scientifico. In generale non è possibile approntare metodi numerici che calcolino gli zeri di una generica funzione in un numero finito di passi. Abbiamo ad esempio ricordato nella Sezione 1.6.1 che un teorema dell'Algebra esclude la possibilità di calcolare con un numero finito di operazioni gli zeri di un generico polinomio di grado maggiore di 4. La situazione è ancor più complicata quando f è una funzione non polinomiale.

I metodi numerici per la risoluzione di questo problema sono pertanto necessariamente *iterativi*. A partire da uno o più dati iniziali, scelti convenientemente, essi generano una successione di valori $x^{(k)}$ che, sotto opportune ipotesi, convergerà ad uno zero α della funzione f studiata.

Inizieremo il capitolo formulando alcuni semplici problemi di interesse applicativo che conducono ad equazioni non lineari. La risoluzione di tali problemi verrà poi svolta nel seguito, dopo aver introdotto ed analizzato i diversi metodi numerici. Questa impostazione verrà poi riproposta in tutti i Capitoli che seguono.

2.1 Alcuni problemi

Problema 2.1 (Piano di investimento) Si vuol calcolare il tasso medio di interesse r di un fondo di investimento su più anni. Supponiamo che all'inizio di ogni anno si investano nel fondo v euro e che alla fine dell'ennesimo anno si sia accumulato un montante pari a M euro. Essendo M legato a r dalla seguente relazione

$$M = v\sum_{k=1}^{n}(1+r)^k = v\frac{1+r}{r}\left[(1+r)^n - 1\right],$$

© Springer-Verlag Italia Srl. 2017

A. Quarteroni et al., *Calcolo Scientifico*, 6a edizione,
UNITEXT – La Matematica per il 3+2 105, DOI 10.1007/978-88-470-3953-7_2

deduciamo che r è la radice dell'equazione non lineare

$$f(r) = 0, \quad \text{dove } f(r) = M - v\frac{1+r}{r}[(1+r)^n - 1].$$

Per la soluzione di questo problema, si veda l'Esempio 2.1. ■

Problema 2.2 (Equazione di stato di un gas) Si vuole determinare il volume V occupato da un gas ad una temperatura T e soggetto ad una pressione p. L'equazione di stato (ossia l'equazione che lega p, V e T) è

$$\left[p + a(N/V)^2\right](V - Nb) = kNT, \tag{2.1}$$

nella quale a e b sono dei coefficienti che dipendono dallo specifico tipo di gas, N è il numero di molecole di gas contenute nel volume V e k è la cosiddetta costante di Boltzmann. Dobbiamo quindi risolvere un'equazione non lineare la cui radice è V. Per la soluzione di questo problema si veda l'Esercizio 2.2. ■

Problema 2.3 (Statica) Consideriamo il sistema meccanico costituito dalle quattro aste rigide $\mathbf{a}_i$ di Figura 2.1; si vuole stabilire, in corrispondenza di un fissato angolo β, quale sia l'angolo α fra le aste $\mathbf{a}_1$ e $\mathbf{a}_2$. A partire dall'identità vettoriale

$$\mathbf{a}_1 - \mathbf{a}_2 - \mathbf{a}_3 - \mathbf{a}_4 = \mathbf{0}$$

ed osservando che l'asta $\mathbf{a}_1$ è sempre allineata con l'asse delle ascisse, è possibile ricavare la seguente relazione tra β e α

$$\frac{a_1}{a_2}\cos(\beta) - \frac{a_1}{a_4}\cos(\alpha) - \cos(\beta - \alpha) = -\frac{a_1^2 + a_2^2 - a_3^2 + a_4^2}{2a_2a_4}, \tag{2.2}$$

avendo indicato con a_i la lunghezza dell'i-esima asta. Evidentemente tale equazione, detta di Freudenstein, si può riscrivere come $f(\alpha) = 0$, essendo

$$f(x) = \frac{a_1}{a_2}\cos(\beta) - \frac{a_1}{a_4}\cos(x) - \cos(\beta - x) + \frac{a_1^2 + a_2^2 - a_3^2 + a_4^2}{2a_2a_4}.$$

Essa può essere risolta analiticamente solo per particolari valori di β. Si tenga inoltre conto che non per tutti i valori di β la soluzione esiste o, se esiste, è unica. Per la sua risoluzione nel caso generale in cui β assuma un valore arbitrario compreso fra 0 e π si dovrà ricorrere ad un metodo numerico (si veda l'Esercizio 2.9). ■

Problema 2.4 (Dinamica delle popolazioni) Nello studio della dinamica delle popolazioni (di batteri, ad esempio) l'equazione $x^+ =$

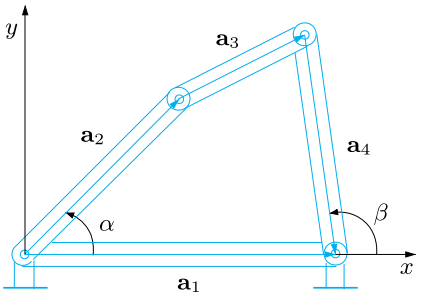

Figura 2.1. Il sistema di quattro aste del Problema 2.3

$\phi(x) = xR(x)$ stabilisce un legame fra il numero x di individui di una generazione ed il numero x^+ di individui della generazione successiva. La funzione $R(x)$ modella il tasso di variazione della popolazione in esame e può essere scelta in vari modi. Tra i più noti, ricordiamo:

1. il modello di Malthus (Thomas Malthus, 1766–1834),

$$R(x) = R_M(x) = r, \quad r > 0;$$

2. il modello di crescita in presenza di risorse limitate, (noto come modello di Beverton-Holt o modello discreto di Verhulst)

$$R(x) = R_V(x) = \frac{r}{1 + xK}, \quad r > 0, \ K > 0, \tag{2.3}$$

   che migliora il modello di Malthus tenendo conto del fatto che la crescita della popolazione è limitata dalle risorse disponibili;
3. il modello predatore/preda con saturazione

$$R(x) = R_P = \frac{rx}{1 + (x/K)^2}, \tag{2.4}$$

   che può essere visto come l'evoluzione del modello di Beverton-Holt in presenza di una popolazione antagonista.

La dinamica di una popolazione è quindi descritta dal processo iterativo

$$x^{(k)} = \phi(x^{(k-1)}), \quad k \geq 1, \tag{2.5}$$

dove $x^{(k)}$ rappresenta il numero di individui presenti k generazioni dopo la generazione iniziale $x^{(0)}$. Inoltre, gli stati stazionari (o di equilibrio) x^* della popolazione considerata sono definiti come le soluzioni del problema

$$x^* = \phi(x^*),$$

o, equivalentemente, $x^* = x^* R(x^*)$, ovvero $R(x^*) = 1$. La (2.5) è un esempio di iterazione di punto fisso (si veda la Sezione 2.6). ■

2.2 Il metodo di bisezione

Sia f una funzione continua in $[a, b]$ tale che $f(a)f(b) < 0$. Sotto tali ipotesi f ammette almeno uno zero in (a, b). (Questo risultato è noto come *Teorema degli zeri di una funzione continua.*) Supponiamo per semplicità che ne abbia uno solo che indicheremo con α. Nel caso in cui f presenti più zeri è sempre possibile, ad esempio attraverso uno studio grafico con il comando `fplot`, individuare un intervallo che ne contenga uno solo.

La strategia del metodo di bisezione consiste nel dimezzare l'intervallo di partenza, selezionare tra i due sotto-intervalli ottenuti quello nel quale f cambia di segno agli estremi ed applicare ricorsivamente questa procedura all'ultimo intervallo selezionato. Più precisamente, detto $I^{(0)} = (a, b)$ e, più in generale, $I^{(k)}$ il sotto-intervallo selezionato al passo k-esimo, si sceglie come $I^{(k+1)}$ il semi-intervallo di $I^{(k)}$ ai cui estremi f cambia di segno. Con tale procedura si è certi che ogni $I^{(k)}$ così individuato conterrà α. La successione $\{x^{(k)}\}$ dei punti medi dei sotto-intervalli $I^{(k)}$ dovrà ineluttabilmente convergere a α, in quanto la lunghezza dei sotto-intervalli tende a 0 per k che tende all'infinito.

Formalizziamo questa idea, ponendo

$$a^{(0)} = a, \qquad b^{(0)} = b, \qquad I^{(0)} = (a^{(0)}, b^{(0)}), \qquad x^{(0)} = (a^{(0)} + b^{(0)})/2.$$

Al generico passo $k \geq 1$ il metodo di bisezione calcolerà allora il semi-intervallo $I^{(k)} = (a^{(k)}, b^{(k)})$ dell'intervallo $I^{(k-1)} = (a^{(k-1)}, b^{(k-1)})$ nel modo seguente:

$$\begin{aligned}
&\text{dato } x^{(k-1)} = (a^{(k-1)} + b^{(k-1)})/2,\\
&\text{se } f(x^{(k-1)}) = 0,\\
&\quad \text{allora } \alpha = x^{(k-1)}\\
&\quad \text{ed il metodo si arresta;}
\end{aligned}$$

altrimenti,

$$\begin{aligned}
&\text{se } f(a^{(k-1)})f(x^{(k-1)}) < 0\\
&\quad \text{si pone } a^{(k)} = a^{(k-1)},\ b^{(k)} = x^{(k-1)};\\
&\text{se } f(x^{(k-1)})f(b^{(k-1)}) < 0\\
&\quad \text{si pone } a^{(k)} = x^{(k-1)},\ b^{(k)} = b^{(k-1)},
\end{aligned}$$

quindi si definisce $x^{(k)} = (a^{(k)} + b^{(k)})/2$ e si incrementa k di uno.

Ad esempio, nel caso rappresentato in Figura 2.2 che corrisponde alla scelta $f(x) = x^2 - 1$, a partire da $a^{(0)} = -0.25$ e $b^{(0)} = 1.25$, otterremmo

$$\begin{aligned}
I^{(0)} &= (-0.25, 1.25), & x^{(0)} &= 0.5,\\
I^{(1)} &= (0.5, 1.25), & x^{(1)} &= 0.875,\\
I^{(2)} &= (0.875, 1.25), & x^{(2)} &= 1.0625,\\
I^{(3)} &= (0.875, 1.0625), & x^{(3)} &= 0.96875.
\end{aligned}$$

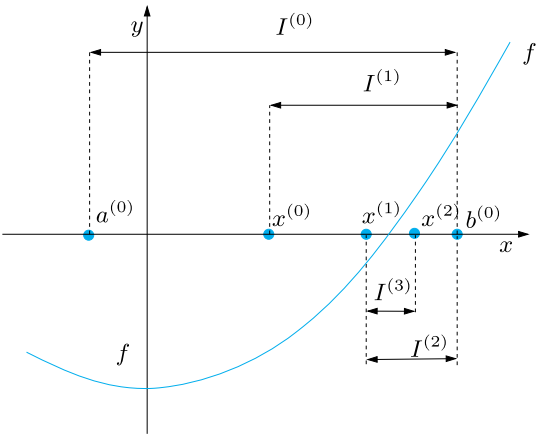

Figura 2.2. Alcune iterazioni del metodo di bisezione

Si noti che ognuno degli intervalli $I^{(k)}$ contiene lo zero α. Inoltre, la successione $\{x^{(k)}\}$ converge necessariamente allo zero α in quanto ad ogni passo l'ampiezza $|I^{(k)}| = b^{(k)} - a^{(k)}$ dell'intervallo $I^{(k)}$ si dimezza. Essendo allora $|I^{(k)}| = (1/2)^k |I^{(0)}|$, l'errore al passo k sarà tale che

$$|e^{(k)}| = |x^{(k)} - \alpha| < \frac{1}{2}|I^{(k)}| = \left(\frac{1}{2}\right)^{k+1} (b-a).$$

Al fine di garantire che $|e^{(k)}| < \varepsilon$ per una assegnata tolleranza ε, basta allora fermarsi dopo $k_{\min}$ iterazioni, essendo $k_{\min}$ il primo intero che soddisfa la disuguaglianza

$$k_{\min} > \log_2\left(\frac{b-a}{\varepsilon}\right) - 1 \tag{2.6}$$

Naturalmente, questa disuguaglianza non dipende dalla particolare funzione f scelta in precedenza.

Il metodo di bisezione è implementato nel Programma 2.1: `fun` è un *function handle* associato alla funzione f, `a` e `b` sono gli estremi dell'intervallo di ricerca, `tol` la tolleranza ε e `kmax` il massimo numero consentito di iterazioni. `fun` oltre al primo argomento relativo alla variabile indipendente, può accettare altri argomenti opzionali impiegati nella definizione di f.

In uscita, `zero` contiene lo zero calcolato, `res` il residuo, ovvero il valore assunto da f in `zero`, e `niter` il numero di iterazioni effettuate. Il comando `find(fx==0)` serve per trovare gli indici del vettore `fx` corrispondenti ad elementi nulli, mentre il comando `varargin` permette alla *function* `fun` di accettare un numero di parametri d'ingresso variabile.

find
varargin

Programma 2.1. bisection: il metodo di bisezione

```
function [zero,res,niter]=bisection(fun,a,b,tol,...
                                    kmax,varargin)
%BISECTION Trova uno zero di una funzione.
%  ZERO=BISECTION(FUN,A,B,TOL,KMAX) approssima uno
%  zero della funzione FUN nell'intervallo [A,B] con
%  il metodo di bisezione. FUN deve essere definita
%  su variabile di tipo array e puo' essere una anony-
%  mous function od una function definita in un M-file.
%  Se la ricerca dello zero di FUN fallisce, il
%  programma restituisce un messaggio d'errore.
%
%  ZERO=BISECTION(FUN,A,B,TOL,KMAX,P1,P2,...) passa
%  i parametri P1, P2,... alla funzione
%  FUN(X,P1,P2,...).
%
%  [ZERO,RES,NITER]= BISECTION(FUN,...) restituisce
%  il valore del residuo RES in ZERO ed il numero di
%  iterazioni effettuate per calcolare il valore ZERO.

x = [a, (a+b)*0.5, b];
fx = fun(x,varargin{:});
if fx(1)*fx(3) > 0
  error([' Il segno della funzione agli estremi',...
   ' dell''intervallo [A,B] deve essere diverso']);
elseif fx(1) == 0
    zero = a; res = 0; niter = 0; return
elseif fx(3) == 0
    zero = b; res = 0; niter = 0; return
end
niter = 0;
I = (b - a)*0.5;
while I >= tol && niter < kmax
   niter = niter + 1;
   if fx(1)*fx(2) <  0
      x(3) = x(2);
      x(2) = x(1)+(x(3)-x(1))*0.5;
      fx = fun(x,varargin{:});
      I = (x(3)-x(1))*0.5;
   elseif fx(2)*fx(3) < 0
      x(1) = x(2);
      x(2) = x(1)+(x(3)-x(1))*0.5;
      fx = fun(x,varargin{:});
      I = (x(3)-x(1))*0.5;
   else
       x(2) = x(find(fx==0)); I = 0;
   end
end
if  (niter==kmax && I > tol)
 fprintf(['Il metodo di bisezione si e'' arrestato',...
 ' senza soddisfare la tolleranza richiesta\n',...
 'avendo raggiunto il numero massimo di iterazioni\n']);
end
zero = x(2); x = x(2);
res = fun(x,varargin{:});
```

Esempio 2.1 (Piano di investimento) Risolviamo con il metodo di bisezione il Problema 2.1, supponendo che v sia pari a 1000 euro e che, dopo 5 anni, M sia uguale a 6000 euro. Dal grafico della funzione f, ottenuto con le seguenti istruzioni

```
M=6000; v=1000; f=@(r) (M-v*(1+r).*((1+r).^5-1)./r);
fplot(f,[0.01,0.3]);
```

(si ricorda che stiamo deliberatamente omettendo il *prompt* allo scopo di alleggerire le notazioni) si ricava che f presenta un'unica radice nell'intervallo $(0.01, 0.1)$, pari a circa 0.06. Eseguiamo quindi il Programma 2.1 con `a` $= 0.01$, `b` $= 0.1$, `tol` $= 10^{-12}$ e `kmax=1000` ed il comando

```
[zero,res,niter]=bisection(f,0.01,0.1,1.e-12,1000)
```

Il metodo converge dopo 36 iterazioni al valore 0.061402411536183, in perfetto accordo con la stima (2.6) per la quale $k_{\min} = 36$. Si può quindi concludere che il tasso di interesse r è pari a circa il 6.14%.

Invece di lavorare con una *anonymous function*, avremmo potuto generare la *function* `Rfuncv.m`

```
function y=Rfuncv(r,M,v)
% RFUNCV function per l'Esempio 2.1
y=M - v*(1+r)./r.*((1+r).^5 - 1);
end
```

ed eseguire le seguenti istruzioni:

```
M=6000; v=1000;
[zero,res,niter]=bisection(@Rfuncv,0.01,0.1,...
1.e-12,1000,M,v)
```

Osserviamo che nel primo caso abbiamo richiamato la *function* `bisection.m` con 5 parametri di input, in quanto i valori di `M` e `v` sono incorporati nel *function handle* `f` al momento della sua definizione. Al contrario, nel secondo caso dobbiamo passare a `bisection.m` anche `M` e `v`, essi saranno memorizzati nella variabile `varargin` e poi passati a `Rfuncv` al momento della sua valutazione. ■

Il metodo di bisezione non garantisce una riduzione monotona dell'errore, ma solo il dimezzamento dell'ampiezza dell'intervallo all'interno del quale si cerca lo zero. Per questo motivo possono essere inavvertitamente scartate approssimazioni di α assai accurate se si usa come unico criterio d'arresto quello sulla lunghezza dell'intervallo $I^{(k)}$.

Questo metodo non tiene infatti conto del reale andamento di f ed in effetti, a meno che l'intervallo di partenza non sia simmetrico rispetto allo zero cercato, esso non converge allo zero in un solo passo neppure se f è una funzione lineare.

Si vedano gli Esercizi 2.1–2.5.

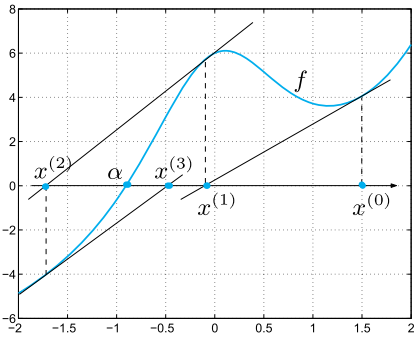

Figura 2.3. Prime iterate generate dal metodo di Newton a partire dal dato iniziale $x^{(0)}$ per la funzione $f(x) = x + e^x + 10/(1 + x^2) - 5$

2.3 Il metodo di Newton

Il metodo di bisezione si limita ad utilizzare il segno che la funzione f assume in certi punti (gli estremi dei sotto-intervalli). Vogliamo ora introdurre un metodo che sfrutti maggiori informazioni su f, precisamente i suoi valori e quelli della sua derivata (nell'ipotesi che quest'ultima esista). A tal fine ricordiamo che l'equazione della retta tangente alla curva $(x, f(x))$ nel punto $x^{(k)}$ è

$$y(x) = f(x^{(k)}) + f'(x^{(k)})(x - x^{(k)}).$$

Se cerchiamo $x^{(k+1)}$ tale che $y(x^{(k+1)}) = 0$, troviamo

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}, \quad k \geq 0 \tag{2.7}$$

purché $f'(x^{(k)}) \neq 0$. La (2.7) consente di calcolare una successione di valori $x^{(k)}$ a partire da un dato iniziale $x^{(0)}$. Il metodo così ottenuto è noto come *metodo di Newton* ed equivale a calcolare lo zero di f sostituendo localmente a f la sua retta tangente (si veda la Figura 2.3).

In effetti, se sviluppiamo f in serie di Taylor in un intorno di un generico punto $x^{(k)}$ troviamo

$$f(x^{(k+1)}) = f(x^{(k)}) + \delta^{(k)} f'(x^{(k)}) + \mathcal{O}\big((\delta^{(k)})^2\big), \tag{2.8}$$

dove $\delta^{(k)} = x^{(k+1)} - x^{(k)}$. Imponendo che $f(x^{(k+1)})$ sia nullo e trascurando il termine $\mathcal{O}((\delta^{(k)})^2)$, possiamo ricavare $x^{(k+1)}$ in funzione di $x^{(k)}$ come nella (2.7). In questo senso la (2.7) può essere vista come una approssimazione della (2.8).

Evidentemente, (2.7) converge allo zero in un solo passo quando f è lineare, cioè della forma $f(x) = a_1 x + a_0$.

Esempio 2.2 Risolviamo con il metodo di Newton lo stesso caso dell'Esempio 2.1 a partire dal dato iniziale $x^{(0)} = 0.3$. Il metodo converge allo zero cercato e dopo 6 iterazioni la differenza fra due iterate successive è minore di 10^{-12}. ■

La convergenza del metodo di Newton non è garantita per ogni scelta di $x^{(0)}$, ma soltanto per valori di $x^{(0)}$ *sufficientemente vicini* ad α, ovvero appartenenti ad un intorno $I(\alpha)$ sufficientemente piccolo di α. Questa richiesta a prima vista sembra insensata: per trovare l'incognita α abbiamo bisogno di scegliere $x^{(0)}$ sufficientemente vicino ad α, quando α è proprio il valore sconosciuto!

In pratica, un possibile valore di $x^{(0)}$ può essere ottenuto utilizzando ad esempio poche iterazioni del metodo di bisezione, oppure attraverso uno studio del grafico di f. Se $x^{(0)}$ è stato scelto opportunamente e se lo zero α è semplice, ovvero se $f'(\alpha) \neq 0$, allora il metodo di Newton converge. Inoltre, nel caso in cui f è derivabile con continuità due volte, otteniamo il seguente risultato di convergenza (si veda l'Esercizio 2.8)

$$\lim_{k\to\infty} \frac{x^{(k+1)} - \alpha}{(x^{(k)} - \alpha)^2} = \frac{f''(\alpha)}{2f'(\alpha)} \tag{2.9}$$

La (2.9) afferma che se $f'(\alpha) \neq 0$ il metodo di Newton converge almeno *quadraticamente* o con ordine 2 nel senso che, per k sufficientemente grande, l'errore al passo $(k+1)$-esimo si comporta come il quadrato dell'errore al passo k-esimo, moltiplicato per una costante indipendente da k.

Se lo zero ha invece molteplicità m maggiore di 1, ovvero $f'(\alpha) = 0, \ldots, f^{(m-1)}(\alpha) = 0$, il metodo di Newton è ancora convergente, purché $x^{(0)}$ sia scelto opportunamente e $f'(x) \neq 0\ \forall x \in I(\alpha) \setminus \{\alpha\}$. Tuttavia in questo caso l'ordine di convergenza è pari a 1 (si veda l'Esercizio 2.17). In tal caso, l'ordine 2 può essere ancora recuperato usando anziché (2.7) la relazione

$$x^{(k+1)} = x^{(k)} - m\frac{f(x^{(k)})}{f'(x^{(k)})}, \quad k \geq 0 \tag{2.10}$$

purché $f'(x^{(k)}) \neq 0$. Naturalmente, questo *metodo di Newton modificato* richiede una conoscenza a priori di m. In mancanza di tale informazione si può formulare un *metodo di Newton adattivo*, ancora di ordine 2, come riportato in [QSSG14, Sez. 6.6.2].

Esempio 2.3 La funzione $f(x) = (x-1)\log(x)$ ha un solo zero, $\alpha = 1$, di molteplicità $m = 2$. Calcoliamolo con il metodo di Newton (2.7) e con la sua versione modificata (2.10). Nel grafico di Figura 2.4 viene riportato l'errore ottenuto con i due metodi in funzione del numero di iterazioni. Come si vede, nel caso del metodo classico (2.7) l'errore decresce solo linearmente. ■

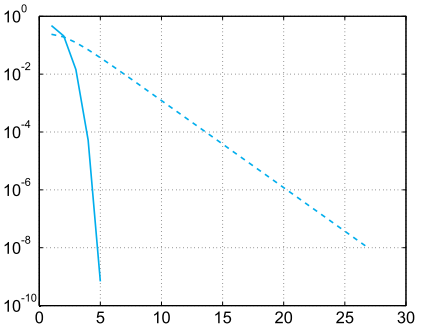

Figura 2.4. Errore in scala semi-logaritmica in funzione del numero di iterazioni per la funzione dell'Esempio 2.3. La curva tratteggiata corrisponde al metodo di Newton (2.7), quella continua al metodo di Newton modificato (2.10) (con $m = 2$)

2.3.1 Come arrestare il metodo di Newton

Il metodo di Newton, quando converge, restituisce il valore esatto di α solo dopo un numero infinito di iterazioni. D'altra parte in generale ci si accontenta di ottenere α a meno di una tolleranza fissata ε: è quindi sufficiente arrestarsi alla prima iterata $k_{\min}$ in corrispondenza della quale si abbia

$$|e^{(k_{\min})}| = |\alpha - x^{(k_{\min})}| < \varepsilon.$$

Si tratta di un test sull'errore. Sfortunatamente essendo l'errore incognito, è necessario impiegare in sua vece degli *stimatori dell'errore* vale a dire delle quantità facilmente calcolabili grazie alle quali sia possibile maggiorare l'errore stesso. Come vedremo al termine della Sezione 2.6, come stimatore dell'errore per il metodo di Newton possiamo prendere la *differenza fra due iterate consecutive* e ci si arresta cioè in corrispondenza del più piccolo intero $k_{\min}$ per il quale

$$|x^{(k_{\min})} - x^{(k_{\min}-1)}| < \varepsilon \tag{2.11}$$

Si tratta di un test sull'incremento. Come vedremo nella Sezione 2.6.1, questo è un buon criterio quando lo zero cercato è semplice. Uno stimatore alternativo, anche per metodi iterativi diversi da quello di Newton vòlti a trovare gli zeri di una funzione f, è dato dal *residuo* al passo k definito come $r^{(k)} = f(x^{(k)})$ che è nullo quando $x^{(k)}$ è uno zero di f.

Il metodo viene in tal caso arrestato alla prima iterata $k_{\min}$ per cui

$$|r^{(k_{\min})}| = |f(x^{(k_{\min})})| < \varepsilon \tag{2.12}$$

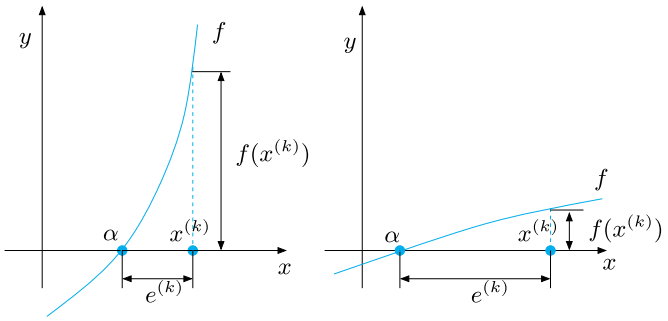

Figura 2.5. Le due possibili situazioni nelle quali il residuo non è un buon stimatore dell'errore: $|f'(x)| \gg 1$ (*a sinistra*), $|f'(x)| \ll 1$ (*a destra*), con x appartenente ad un intervallo contenente α

Il residuo fornisce una stima accurata dell'errore solo quando $|f'(x)|$ è circa pari a 1 in un intorno I_α dello zero α cercato (si veda la Figura 2.5). In caso contrario, porterà ad una sovrastima dell'errore se $|f'(x)| \gg 1$ per $x \in I_\alpha$ o ad una sottostima se $|f'(x)| \ll 1$ (si vedano gli Esercizi 2.6 e 2.20).

Nel Programma 2.2 viene riportata una implementazione del metodo di Newton nella sua forma (2.7) (per utilizzare la forma modificata è sufficiente inserire, invece di f', la funzione f'/m). I parametri `fun` e `dfun` sono *function handle* associati alla funzione f ed alla sua derivata prima, mentre `x0` è il dato iniziale. Il metodo viene arrestato se il valore assoluto della differenza fra due iterate consecutive è minore della tolleranza `tol` o se è stato oltrepassato il massimo numero di iterazioni consentito, `kmax`. I parametri di *output* contengono: `zero` l'approssimazione calcolata $\widehat{\alpha}$ della radice α, `res` il valore $f(\widehat{\alpha})$ del residuo, `niter` il numero minimo di iterazioni necessarie a soddisfare il test d'arresto e `difv` il vettore dei valori assoluti $|x^{(k+1)} - x^{(k)}|$ delle differenze tra due iterate successive (utilizzati per il test d'arresto).

Programma 2.2. newton: il metodo di Newton

```
function [zero,res,niter,difv]=newton(fun,dfun,x0,...
                                tol,kmax,varargin)
%NEWTON Trova uno zero di una funzione.
%   ZERO=NEWTON(FUN,DFUN,X0,TOL,KMAX) approssima lo
%   zero ZERO della funzione definita nella function
%   FUN, continua e derivabile, usando il metodo di
%   Newton e partendo da X0. Se la ricerca
%   dello zero fallisce in KMAX iterazioni, il pro-
%   gramma restituisce un messaggio d'errore.
%   FUN e DFUN possono essere anonymous
%   function o function definite in M-file.
%   ZERO=NEWTON(FUN,DFUN,X0,TOL,KMAX,P1,P2,...) passa
```

```
%   i parametri P1,P2,... alle funzioni
%   FUN(X,P1,P2,...) e DFUN(X,P1,P2,...).
%   [ZERO,RES,NITER,DIFV]= NEWTON(FUN,...) restituisce
%   il valore del residuo RES in ZERO, il numero di
%   iterazioni NITER necessario per calcolare ZERO  ed
%   il vettore DIFV degli incrementi |x^(k+1)-x^k|

x = x0;
fx = fun(x,varargin{:});
dfx = dfun(x,varargin{:});
k = 0; diff = tol+1; difv=[ ];
while diff >= tol && k < kmax
   k = k + 1;
   diff = - fx/dfx;
   x = x + diff;
   diff = abs(diff); difv=[difv; diff];
   fx = fun(x,varargin{:});
   dfx = dfun(x,varargin{:});
end
if (k==kmax && diff > tol)
  fprintf(['Newton si e'' arrestato senza aver ',...
   'soddisfatto l''accuratezza richiesta, avendo\n',...
   'raggiunto il massimo numero di iterazioni\n']);
end
zero = x; res = fx; niter=k;
```

2.4 Il metodo delle secanti

In molte applicazioni è possibile che la funzione f di cui vogliamo calcolare gli zeri non sia nota in forma esatta, ma che sia solo valutabile in un punto x assegnato attraverso un programma. Di conseguenza, ci risulta impossibile poter valutare la sua derivata in maniera esatta ed applicare il metodo di Newton.

Per ovviare a questo inconveniente, la valutazione di $f'(x^{(k)})$ può essere sostituita da un rapporto incrementale calcolato su valori di f già noti. Una possibile implementazione di questa strategia è quella del metodo delle secanti: assegnati due punti $x^{(0)}$ e $x^{(1)}$, per $k \geq 1$ si calcola

$$x^{(k+1)} = x^{(k)} - \left(\frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}} \right)^{-1} f(x^{(k)}) \tag{2.13}$$

Rinunciando alla conoscenza esatta della derivata prima, in caso di convergenza, la velocità sarà inferiore a quella del metodo di Newton. In effetti si può dimostrare che, se α è radice semplice e $I(\alpha)$ un suo opportuno intorno, se $x^{(0)}$ e $x^{(1)}$ sono sufficientemente vicini ad α e $f'(x) \neq 0 \ \forall x \in I(\alpha) \setminus \{\alpha\}$, allora il metodo delle secanti (2.13) converge ad α. Inoltre, se $f \in C^2(I(\alpha))$ e $f'(\alpha) \neq 0$, allora esiste una costante

$c > 0$ tale che

$$|x^{(k+1)} - \alpha| \leq c|x^{(k)} - \alpha|^p, \quad \text{con } p = \frac{1+\sqrt{5}}{2} \simeq 1.618\ldots \tag{2.14}$$

cioè il metodo delle secanti converge con ordine p *super-lineare*. Se invece la radice α è multipla, allora la convergenza è soltanto lineare come succederebbe usando il metodo di Newton.

Esempio 2.4 Risolviamo con il metodo delle secanti lo stesso caso dell'Esempio 2.1 a partire dai dati iniziali $x^{(0)} = 0.3$ e $x^{(1)} = -0.3$. Il metodo converge allo zero cercato in 8 iterazioni, contro le 6 iterazioni necessarie al metodo di Newton partendo da $x^{(0)} = 0.3$.

Scegliendo $x^{(0)} = 0.3$ e $x^{(1)} = 0.1$ il metodo delle secanti convergerebbe in 6 iterazioni, al pari di Newton. ■

2.5 I sistemi di equazioni non lineari

Consideriamo il seguente sistema di equazioni non lineari

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) = 0, \\ f_2(x_1, x_2, \ldots, x_n) = 0, \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) = 0, \end{cases} \tag{2.15}$$

dove $f_1, \ldots, f_n$ sono funzioni non lineari. Se poniamo $\mathbf{f} \equiv (f_1, \ldots, f_n)^T$ e $\mathbf{x} \equiv (x_1, \ldots, x_n)^T$, possiamo riscrivere il sistema (2.15) nella forma

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}. \tag{2.16}$$

Un semplice esempio di sistema non lineare è il seguente

$$\begin{cases} f_1(x_1, x_2) = x_1^2 + x_2^2 - 1 = 0, \\ f_2(x_1, x_2) = \sin(\pi x_1/2) + x_2^3 = 0. \end{cases} \tag{2.17}$$

Al fine di estendere il metodo di Newton al caso di un sistema, sostituiamo alla derivata prima della funzione scalare f la *matrice Jacobiana* $\mathrm{J}_\mathbf{f}$ della funzione vettoriale $\mathbf{f}$, le cui componenti sono

$$(\mathrm{J}_\mathbf{f})_{ij} \equiv \frac{\partial f_i}{\partial x_j}, \quad i, j = 1, \ldots, n.$$

Il simbolo $\partial f_i / \partial x_j$ rappresenta la derivata parziale di f_i rispetto a x_j (si veda la definizione (1.11)). Con questa notazione, il metodo di Newton

per (2.16) diventa: dato $\mathbf{x}^{(0)} \in \mathbb{R}^n$, per $k = 0, 1, \ldots$, fino a convergenza

$$\begin{array}{ll} \text{risolvere} & \mathrm{J}_{\mathbf{f}}(\mathbf{x}^{(k)})\boldsymbol{\delta}\mathbf{x}^{(k)} = -\mathbf{f}(\mathbf{x}^{(k)}); \\ \text{porre} & \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}\mathbf{x}^{(k)} \end{array} \tag{2.18}$$

Di conseguenza, esso richiede ad ogni passo la soluzione di un sistema lineare di matrice $\mathrm{J}_{\mathbf{f}}(\mathbf{x}^{(k)})$.

Il Programma 2.3 implementa il metodo di Newton per un sistema non lineare usando il comando \ di MATLAB (si veda la Sezione 5.8) per risolvere il sistema lineare sulla matrice Jacobiana. In ingresso, è necessario definire un vettore che rappresenta il dato iniziale e due *function*, `Ffun` e `Jfun`, che calcolano, rispettivamente, il vettore colonna `F`, contenente la valutazione di $\mathbf{f}$ su un generico vettore `x`, e la matrice Jacobiana $\mathrm{J}_{\mathbf{f}}$, anch'essa valutata su un vettore `x`. `Ffun` e `Jfun` possono essere *function handles* o *user-defined functions*. Il metodo si arresta quando la differenza in norma euclidea fra due iterate consecutive è minore di `tol` o quando viene raggiunto il massimo numero di iterazioni consentito `kmax`. Le variabili in output assumono lo stesso significato che hanno nel Programma `newton` 2.2, ad eccezione di `difv` che ora contiene i valori $\|\mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}\|$ al variare di $k = 0, \ldots$, fino a convergenza.

Programma 2.3. newtonsys: il metodo di Newton per un sistema non lineare

```
function [x,res,niter,difv] = newtonsys(Ffun,Jfun,...
                                  x0,tol, kmax, varargin)
%NEWTONSYS calcola una radice di un sistema non lineare
%  [ZERO,RES,NITER,DIFV]=NEWTONSYS(FFUN,JFUN,X0,...
%                                  TOL, KMAX)
%  calcola il vettore ZERO, radice di un sistema non
%  lineare definito nella function FFUN con matrice
%  Jacobiana definita nella function JFUN a partire
%  dal vettore X0. RES contiene il valore del residuo
%  in ZERO e NITER il numero di iterazioni necessarie
%  per calcolare ZERO. Il vettore DIFV contiene le
%  norme ||x^(k+1)-x^(k)||. FFUN e JFUN possono essere
%  anonymous functions o user-defined functions.

k = 0;
err = tol + 1; difv=[ ];
x = x0;
while err >= tol && k < kmax
    J = Jfun(x,varargin{:});
    F = Ffun(x,varargin{:});
    delta = - J\F;
    x = x + delta;
    err = norm(delta); difv=[difv; err];
    k = k + 1;
end
res = norm(Ffun(x,varargin{:}));
if (k==kmax && err> tol)
```

```
    fprintf(['Il metodo non converge nel massimo ',...
       'numero di iterazioni. L''ultima iterata\n',...
       'calcolata ha residuo relativo pari a %e\n'],F);
end
niter=k;
```

Esempio 2.5 Consideriamo il sistema non lineare (2.17) che ammette le due soluzioni (individuabili, ad esempio, per via grafica) $(0.4761, -0.8794)$ e $(-0.4761, 0.8794)$ (riportiamo le sole prime 4 cifre significative).

Partendo dal dato iniziale `x0=[1;1]` e usando le seguenti istruzioni:

```
Ffun=@(x)[x(1)^2+x(2)^2-1;
          sin(pi*x(1)/2)+x(2)^3];
pi2 = 0.5*pi;
Jfun=@(x)[2*x(1), 2*x(2);
          pi2*cos(pi2*x(1)), 3*x(2)^2];
x0=[1;1]; tol=1e-5; kmax=10;
[x,F,niter,difv] = newtonsys(Ffun,Jfun,x0,tol,kmax);
```

il metodo di Newton converge in 8 iterazioni al vettore

```
4.760958225338114e-01
-8.793934089897496e-01
```

Si noti che per far convergere il metodo all'altra radice basta scegliere come dato iniziale `x0=[-1;-1]`. In generale, esattamente come nel caso scalare, la convergenza del metodo di Newton dipende dalla scelta del dato iniziale $\mathbf{x}^{(0)}$ ed in particolare bisogna garantire che $\det(\mathrm{J}_{\mathbf{f}}(\mathbf{x}^{(0)})) \neq 0$. ■

Il metodo delle secanti può essere adattato alla risoluzione di sistemi di equazioni non lineari, mantenendo l'ordine di convergenza superlineare. L'idea di base è quella di sostituire le matrici Jacobiane $\mathrm{J}_{\mathbf{f}}(\mathbf{x}^{(k)})$ (per $k \geq 0$) con delle matrici B_k definite ricorsivamente a partire da una matrice B_0 che sia una approssimazione di $\mathrm{J}_{\mathbf{f}}(\mathbf{x}^{(0)})$. (Approssimazioni alternative verranno considerate nella Sezione 4.2 e nel Capitolo 9.) Il metodo più noto che si basa su questa idea è quello di Broyden. Utilizzando le stesse notazioni della sezione precedente, esso si formula così: dato $\mathbf{x}^{(0)} \in \mathbb{R}^n$, data $\mathrm{B}_0 \in \mathbb{R}^{n\times n}$, per $k = 0, 1, \ldots$, fino a convergenza

$$
\begin{aligned}
&\text{risolvere} && \mathrm{B}_k \boldsymbol{\delta}\mathbf{x}^{(k)} = -\mathbf{f}(\mathbf{x}^{(k)}) \\
&\text{porre} && \mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \boldsymbol{\delta}\mathbf{x}^{(k)} \\
&\text{porre} && \boldsymbol{\delta}\mathbf{f}^{(k)} = \mathbf{f}(\mathbf{x}^{(k+1)}) - \mathbf{f}(\mathbf{x}^{(k)}) \\
&\text{calcolare} && \mathrm{B}_{k+1} = \mathrm{B}_k + \frac{(\boldsymbol{\delta}\mathbf{f}^{(k)} - \mathrm{B}_k \boldsymbol{\delta}\mathbf{x}^{(k)}) {\boldsymbol{\delta}\mathbf{x}^{(k)}}^T}{{\boldsymbol{\delta}\mathbf{x}^{(k)}}^T \boldsymbol{\delta}\mathbf{x}^{(k)}}
\end{aligned}
\tag{2.19}
$$

Facciamo osservare che non si chiede alla successione $\{\mathrm{B}_k\}$ così costruita di convergere alla vera matrice Jacobiana $\mathrm{J}_{\mathbf{f}}(\boldsymbol{\alpha})$ ($\boldsymbol{\alpha}$ è la radice del sistema); questo risultato in effetti non è nemmeno garantito. Piuttosto, si ha

$$\lim_{k\to\infty} \frac{\|(\mathrm{B}_k - \mathrm{J}_{\mathbf{f}}(\boldsymbol{\alpha}))(\mathbf{x}^{(k)} - \boldsymbol{\alpha})\|}{\|\mathbf{x}^{(k)} - \boldsymbol{\alpha}\|} = 0.$$

Ciò esprime il fatto che B_k approssima bene $\mathrm{J}_{\mathbf{f}}(\boldsymbol{\alpha})$ lungo la direzione dell'errore $\mathbf{x}^{(k)} - \boldsymbol{\alpha}$, garantendo una convergenza super-lineare.

Ad ogni passo, il costo $\mathcal{O}(n^3)$ per il calcolo di $\boldsymbol{\delta}\mathbf{x}^{(k)}$ può essere ridotto ad $\mathcal{O}(n^2)$, utilizzando ricorsivamente fattorizzazioni QR sulle matrici B_k (si veda, ad esempio, [GM72]) e, grazie all'uguaglianza $(\boldsymbol{\delta}\mathbf{f}^{(k)} - \mathrm{B}_k\boldsymbol{\delta}\mathbf{x}^{(k)}) = \mathbf{f}(\mathbf{x}^{(k+1)})$, non serve implementare prodotti matrice–vettore nel calcolo di B_{k+1}.

Per una descrizione più completa del metodo di Broyden e di altri metodi di tipo secanti (detti anche metodi *quasi-Newton*) rimandiamo a [DS96], [Deu04], [SM03] e [QSSG14, Cap. 6].

Il metodo di Broyden (2.19) è implementato nel Programma 2.4. Il parametro di *input* `B0` contiene la matrice B_0, tutti gli altri parametri sono definiti come per il Programma 2.3.

Programma 2.4. broyden: il metodo di Broyden

```
function [zero,res,niter,difv]=broyden(fun,B0,x0,...
                                  tol, kmax,varargin)
%BROYDEN Trova uno zero di un sistema di funzioni.
%   ZERO=BROYDEN(FUN,B0,X0,TOL,KMAX) approssima lo
%   zero ZERO del sistema di funzioni definite nella
%   function FUN, usando il metodo di Broyden
%   partendo da X0, dove B0 e' l'approssimazione
%   dello Jacobiano al passo 0. FUN accetta in ingresso
%   un vettore x e restituisce un vettore della stessa
%   dimensione. Se la ricerca dello zero fallisce in
%   KMAX iterazioni, il programma restituisce un mes-
%   saggio d'errore. FUN puo' essere una anonymous
%   function o una function definita in M-file.
%   ZERO=BROYDEN(FUN,B0,X0,TOL,KMAX,P1,P2,...)
%   passa i parametri P1,P2,... alla funzione
%   FUN(X,P1,P2,...).
%   [ZERO,RES,NITER,DIFV]= BROYDEN(FUN,...) restituisce
%   il valore del residuo RES in ZERO, il numero di
%   iterazioni NITER necessario per calcolare ZERO ed
%   il vettore DIFV delle norme ||x^(k+1)-x^(k)||

fx0 = fun(x0,varargin{:});
k = 0;
diff = tol+1; difv= [ ];
while diff >= tol && k < kmax
  k = k + 1;
  deltax=-B0\fx0;
  x1=x0+deltax;
  fx1=fun(x1,varargin{:});
  B0=B0+(fx1*deltax')/(deltax'*deltax);
  diff = norm(deltax);
  difv=[difv;diff];
  x0=x1; fx0=fx1;
end
zero = x1; res = norm(fx1);
if (k==kmax && diff > tol)
```

```
  fprintf(['Broyden si e'' arrestato senza aver ',...
   'soddisfatto l''accuratezza richiesta, avendo\n',...
   'raggiunto il massimo numero di iterazioni\n']);
end
niter=k;
```

Esempio 2.6 Consideriamo il problema dell'Esempio 2.5 e risolviamolo con il metodo di Broyden (2.19). Prendendo $B_0 = I$, tolleranza $\varepsilon = 10^{-5}$ per il test d'arresto sull'incremento e $\mathbf{x}^{(0)} = (1,1)^T$ otteniamo convergenza in 10 iterazioni al punto $(0.476095825652119, -0.879393405072448)^T$ con un residuo in norma pari a 1.324932e−08, contro le 8 iterazioni del metodo di Newton ed un residuo in norma pari a 2.235421e−11. Scegliendo $\mathbf{x}^{(0)} = (-1,-1)^T$, sempre con $B_0 = I$, otteniamo convergenza alla seconda radice in 17 iterazioni con residuo in norma uguale a 5.744382e−08 contro 8 iterazioni di Newton e residuo in norma uguale a 2.235421e−11. Scegliendo invece $B_0 = 2I$, il numero delle iterazioni di Broyden si riduce a 12, evidenziando quanto sia importante scegliere bene la matrice iniziale al fine di velocizzare la convergenza.

Per quanto riguarda l'accuratezza delle soluzioni calcolate, osserviamo che il residuo "di Newton" è di 3 ordini di grandezza inferiore al residuo "di Broyden" inducendoci a concludere che le soluzioni ottenute con il metodo di Newton siano comunque più accurate di quelle ottenute con quello di Broyden. ■

Si vedano gli Esercizi 2.6–2.16.

Riassumendo

1. Il calcolo degli zeri di una funzione f viene condotto attraverso metodi iterativi;
2. il metodo di bisezione consente di approssimare uno zero di una funzione "incapsulandolo" in intervalli la cui ampiezza viene dimezzata ad ogni iterazione. Esso converge sempre purché f sia continua nell'intervallo di partenza e cambi di segno agli estremi;
3. il metodo di Newton è un metodo nel quale l'approssimazione dello zero α di f viene condotta utilizzando i valori assunti da f e dalla sua derivata prima. Esso generalmente converge solo per valori del dato iniziale sufficientemente vicini ad α;
4. quando converge, il metodo di Newton converge quadraticamente se α è uno zero semplice, linearmente altrimenti;
5. il metodo di Newton può essere esteso al caso del calcolo degli zeri di un sistema di equazioni non lineari;
6. il metodo delle secanti è una approssimazione di quello di Newton in cui la derivata prima sia sostituita da un rapporto incrementale. Se α è semplice, esso converge più che linearmente, ma meno che quadraticamente; se α è multipla esso converge linearmente. Come per il metodo di Newton, i punti iniziali devono essere scelti in prossimità della radice.

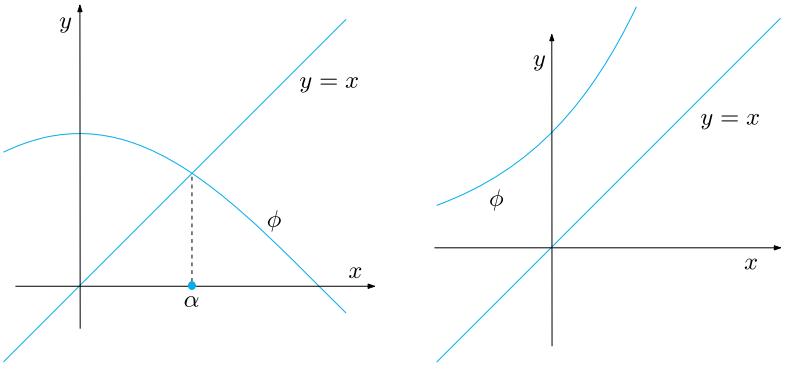

Figura 2.6. La funzione $\phi(x) = \cos x$ (*a sinistra*) ammette un solo punto fisso, mentre la funzione $\phi(x) = e^x$ (*a destra*) non ne ammette alcuno

2.6 Iterazioni di punto fisso

Con una calcolatrice si può facilmente verificare che applicando ripetutamente la funzione coseno partendo dal numero 1 si genera la seguente successione di numeri reali

$$\begin{aligned} x^{(1)} &= \cos(1) = 0.54030230586814, \\ x^{(2)} &= \cos(x^{(1)}) = 0.85755321584639, \\ &\vdots \\ x^{(10)} &= \cos(x^{(9)}) = 0.74423735490056, \\ &\vdots \\ x^{(20)} &= \cos(x^{(19)}) = 0.73918439977149, \end{aligned}$$

che tende al valore $\alpha = 0.73908513\ldots$. Essendo per costruzione $x^{(k+1)} = \cos(x^{(k)})$ per $k = 0, 1, \ldots$ (con $x^{(0)} = 1$), α è tale che $\cos(\alpha) = \alpha$: per questa ragione esso viene detto un *punto fisso* della funzione coseno. L'interesse per un metodo che sfrutti iterazioni di questo tipo è evidente: se α è punto fisso per il coseno, allora esso è uno zero della funzione $f(x) = x - \cos(x)$ ed il metodo appena proposto potrebbe essere usato per il calcolo degli zeri di f (uno solo, in questo caso). D'altra parte non tutte le funzioni ammettono punti fissi; se ad esempio si ripete l'esperimento precedente con la funzione esponenziale a partire da $x^{(0)} = 1$, dopo soli 4 passi si giunge ad una situazione di *overflow* (si veda la Figura 2.6). Precisiamo meglio questa idea intuitiva. Consideriamo pertanto il seguente problema: data una funzione $\phi : [a, b] \to \mathbb{R}$, trovare $\alpha \in [a, b]$ tale che

$$\alpha = \phi(\alpha).$$

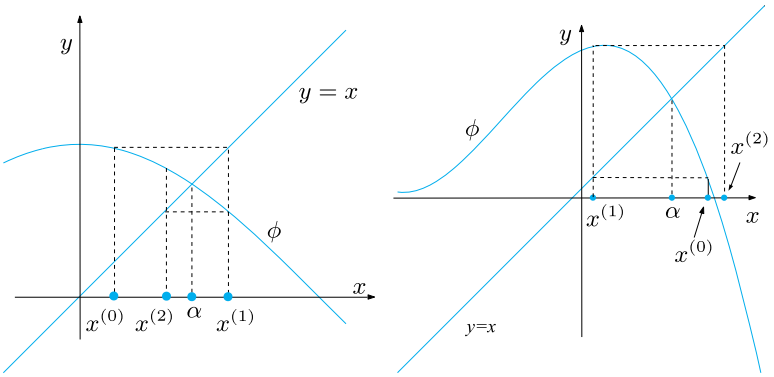

Figura 2.7. Rappresentazione delle prime iterazioni di punto fisso per due funzioni di iterazione. Le iterazioni convergono verso il punto fisso α (*a sinistra*), mentre si allontanano da α (*a destra*)

Se un tale α esiste, viene detto un punto fisso di ϕ e lo si può determinare come limite della seguente successione

$$x^{(k+1)} = \phi(x^{(k)}), \quad k \geq 0 \tag{2.20}$$

dove $x^{(0)}$ è un dato iniziale. Questo algoritmo è detto delle *iterazioni di punto fisso* e ϕ ne è detta la *funzione di iterazione*. L'esempio introduttivo è dunque un algoritmo di iterazioni di punto fisso per la funzione $\phi(x) = \cos(x)$.

Un'interpretazione geometrica della (2.20) viene riportata nel grafico di sinistra di Figura 2.7. Si intuisce che, se ϕ è una funzione continua e se esiste il limite della successione $\{x^{(k)}\}$, allora tale limite è un punto fisso di ϕ. Preciseremo bene questo risultato nelle Proposizioni 2.1 e 2.2.

Esempio 2.7 Il metodo di Newton (2.7) può essere riletto come un algoritmo di iterazioni di punto fisso per la funzione

$$\phi(x) = x - \frac{f(x)}{f'(x)}. \tag{2.21}$$

Tale funzione verrà d'ora in poi indicata con il simbolo ϕ_N, dove N sta per Newton. I metodi di bisezione e di secanti non sono invece iterazioni di punto fisso, in quanto la generica iterata $x^{(k+1)}$ può non dipendere dalla sola $x^{(k)}$, ma anche da $x^{(k-1)}$. ■

Come mostrato dalla Figura 2.7 (a destra), non tutte le funzioni di iterazione garantiscono che le iterazioni di punto fisso convergano. Vale infatti il seguente risultato:

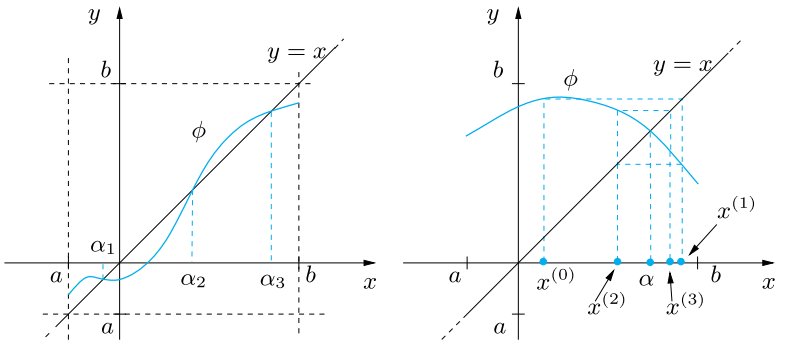

Figura 2.8. Una funzione di punto fisso che ammette 3 punti fissi (*a sinistra*), una funzione di punto fisso che soddisfa l'ipotesi (2.22) ed i primi elementi della successione (2.24) convergente all'unico punto fisso α (*a destra*)

Proposizione 2.1 *Consideriamo la successione* (2.20).

1. *Supponiamo che* $\phi(x)$ *sia continua in* $[a,b]$ *e che* $\phi(x) \in [a,b]$ *per ogni* $x \in [a,b]$*; allora esiste almeno un punto fisso* $\alpha \in [a,b]$.
2. *Se supponiamo inoltre che*

$$\exists L < 1 \text{ t.c. } |\phi(x_1) - \phi(x_2)| \leq L|x_1 - x_2| \quad \forall x_1, x_2 \in [a,b], \tag{2.22}$$

allora ϕ *ha un unico punto fisso* $\alpha \in [a,b]$ *e la successione definita nella* (2.20) *converge a* α*, qualunque sia il dato iniziale* $x^{(0)}$ *in* $[a,b]$.

Dimostrazione. *1.* Dimostriamo dapprima l'esistenza di punti fissi per ϕ. Definiamo la funzione $g(x) = \phi(x) - x$, essa è continua per costruzione su $[a,b]$ e, per l'ipotesi sull'immagine di ϕ, si ha $g(a) = \phi(a) - a \geq 0$ e $g(b) = \phi(b) - b \leq 0$. Applicando il teorema degli zeri di una funzione continua, concludiamo che g ammette almeno uno zero in $[a,b]$, ovvero ϕ ammette almeno un punto fisso in $[a,b]$. (Per un esempio si veda la Figura 2.8.)

2. Supponiamo ora che valga l'ipotesi (2.22).

Se esistessero due punti fissi distinti α_1 e α_2 avremmo

$$|\alpha_1 - \alpha_2| = |\phi(\alpha_1) - \phi(\alpha_2)| \leq L|\alpha_1 - \alpha_2| < |\alpha_1 - \alpha_2|,$$

il che è assurdo.

Dimostriamo ora che la successione $x^{(k)}$ definita in (2.20) converge per $k \to \infty$ all'unico punto fisso α, per ogni scelta del dato iniziale $x^{(0)} \in [a,b]$. Abbiamo

$$\begin{aligned} 0 \leq |x^{(k+1)} - \alpha| &= |\phi(x^{(k)}) - \phi(\alpha)| \\ &\leq L|x^{(k)} - \alpha| \leq \ldots \leq L^{k+1}|x^{(0)} - \alpha|, \end{aligned}$$

ovvero, $\forall k \geq 0$,

$$\frac{|x^{(k)} - \alpha|}{|x^{(0)} - \alpha|} \leq L^k. \tag{2.23}$$

Passando al limite per $k \to \infty$, otteniamo $\lim_{k\to\infty} |x^{(k)} - \alpha| = 0$, che è il risultato cercato. ■

Nella pratica è però spesso difficile delimitare *a priori* l'ampiezza dell'intervallo $[a, b]$; in tal caso è utile il seguente risultato di convergenza *locale*, per la cui dimostrazione si rimanda a [OR70].

Teorema 2.1 (di Ostrowski) *Sia α un punto fisso di una funzione ϕ continua e derivabile con continuità in un opportuno intorno $\mathcal{J}$ di α. Se risulta $|\phi'(\alpha)| < 1$, allora esiste $\delta > 0$ in corrispondenza del quale la successione $\{x^{(k)}\}$ converge ad α, per ogni $x^{(0)}$ tale che $|x^{(0)} - \alpha| < \delta$. Inoltre si ha*

$$\lim_{k\to\infty} \frac{x^{(k+1)} - \alpha}{x^{(k)} - \alpha} = \phi'(\alpha) \tag{2.24}$$

Dimostrazione. Limitiamoci a verificare la proprietà (2.24). Per il teorema di Lagrange, per ogni $k \geq 0$, esiste un punto ξ_k compreso tra $x^{(k)}$ e α tale che $x^{(k+1)} - \alpha = \phi(x^{(k)}) - \phi(\alpha) = \phi'(\xi_k)(x^{(k)} - \alpha)$, ovvero

$$(x^{(k+1)} - \alpha)/(x^{(k)} - \alpha) = \phi'(\xi_k). \tag{2.25}$$

Poiché ξ_k è compreso tra $x^{(k)}$ ed α, si ha $\lim_{k\to\infty} \xi_k = \alpha$ e, passando al limite in entrambi i termini di (2.25) e ricordando che ϕ' è continua in un intorno di α, si ottiene (2.24). ■

Dalla (2.23) e dalla (2.24) si deduce che le iterazioni di punto fisso convergono almeno *linearmente* cioè che, per k sufficientemente grande, l'errore al passo $k + 1$ si comporta come l'errore al passo k moltiplicato per una costante (L in (2.23), $\phi'(\alpha)$ in (2.24)) indipendente da k ed il cui valore assoluto è minore di 1.

Per questo motivo tale costante viene detta *fattore di convergenza asintotico*. Va infine osservato che la convergenza sarà tanto più rapida quanto più piccola è tale costante.

Osservazione 2.1 Nel caso in cui $|\phi'(\alpha)| > 1$, dalla (2.25) segue che se $x^{(k)}$ è sufficientemente vicino ad α, in modo tale che $|\phi'(x^{(k)})| > 1$, allora $|\alpha - x^{(k+1)}| > |\alpha - x^{(k)}|$, e non è possibile che la successione converga al punto fisso. Quando invece $|\phi'(\alpha)| = 1$ non si può trarre alcuna conclusione poiché potrebbero verificarsi sia la convergenza sia la divergenza, a seconda delle caratteristiche della funzione di punto fisso. ■

Esempio 2.8 La funzione $\phi(x) = \cos(x)$ soddisfa le ipotesi del Teorema 2.1 in quanto $|\phi'(\alpha)| = |\sin(\alpha)| \simeq 0.67 < 1$ e, di conseguenza per continuità, esiste un

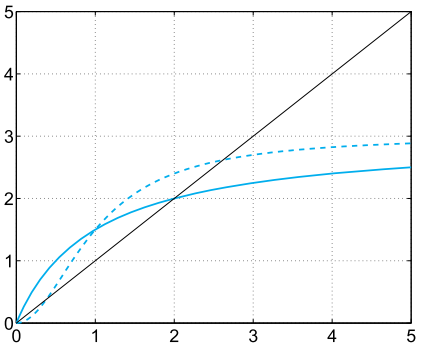

Figura 2.9. I punti fissi per due diversi modelli di dinamica delle popolazioni: il modello discreto di Verhulst (*in linea continua*) e quello predatore/preda (*in linea tratteggiata*)

intorno I_α di α nel quale $|\phi'(x)| < 1$ per ogni $x \in I_\alpha$. La funzione $\phi(x) = x^2 - 1$, pur possedendo due punti fissi $\alpha_\pm = (1 \pm \sqrt{5})/2$, non verifica le ipotesi per nessuno dei due in quanto $|\phi'(\alpha_\pm)| = |1 \pm \sqrt{5}| > 1$. La corrispondente iterazione di punto fisso non sarà pertanto convergente. ■

Esempio 2.9 (Dinamica delle popolazioni) Applichiamo le iterazioni di punto fisso alla funzione $\phi_V(x) = rx/(1+xK)$ del modello discreto di Verhulst (2.3) ed alla funzione $\phi_P(x) = rx^2/(1+(x/K)^2)$ del modello predatore/preda (2.4) scegliendo $r = 3$ e $K = 1$. Se partiamo dal dato iniziale $x^{(0)} = 1$ troviamo il punto fisso $\alpha = 2$ nel primo caso e $\alpha = 2.6180$ nel secondo (si veda la Figura 2.9). Il punto fisso $\alpha = 0$ comune a ϕ_V e ϕ_P può essere calcolato solo come punto fisso di ϕ_P, ma non di ϕ_V. Infatti $\phi_P'(\alpha) = 0$, mentre $|\phi_V'(\alpha)| = r > 1$. Analogamente il punto fisso $\alpha = 0.3820\ldots$ di ϕ_P non può essere calcolato in quanto $|\phi_P'(\alpha)| > 1$. Per lo svolgimento di questo esercizio abbiamo utilizzato il Programma 10.1 `fixedpoint.m` (si veda il Capitolo 10). ■

La convergenza quadratica non è prerogativa del solo metodo di Newton. In generale, vale infatti la seguente proprietà:

Proposizione 2.2 *Si suppongano valide le ipotesi del Teorema 2.1. Se, inoltre, ϕ è derivabile con continuità due volte e se*

$$\phi'(\alpha) = 0, \qquad \phi''(\alpha) \neq 0,$$

allora il metodo di punto fisso (2.20) è convergente di ordine 2 e si ha

$$\lim_{k\to\infty} \frac{x^{(k+1)} - \alpha}{(x^{(k)} - \alpha)^2} = \frac{1}{2}\phi''(\alpha) \tag{2.26}$$

Dimostrazione. Basta osservare che, in questo caso,

$$x^{(k+1)} - \alpha = \phi(x^{(k)}) - \phi(\alpha) = \phi'(\alpha)(x^{(k)} - \alpha) + \frac{\phi''(\eta^{(k)})}{2}(x^{(k)} - \alpha)^2$$

per un opportuno $\eta^{(k)}$ compreso tra $x^{(k)}$ e α. ■

L'Esempio 2.7 mostra che le iterazioni di punto fisso (2.20) possono servire anche per il calcolo degli zeri di funzioni. Naturalmente, data una funzione f, la ϕ definita in (2.21) non è l'unica funzione di iterazione possibile. Ad esempio, per la soluzione dell'equazione $\log(x) = \gamma$, posto $f(x) = \log(x) - \gamma$, la scelta (2.21) condurrebbe alla funzione di iterazione

$$\phi_N(x) = x(1 - \log(x) + \gamma).$$

Un altro metodo di punto fisso si trova sommando x ad ambo i membri dell'equazione $f(x) = 0$. La funzione di iterazione associata è ora $\phi_1(x) = x + \log(x) - \gamma$. Un terzo metodo può essere infine ricavato moltiplicando per x l'equazione e scegliendo $\phi_2(x) = x\log(x)/\gamma$.

Non tutti questi metodi sono convergenti; ad esempio, se $\gamma = -2$, i metodi con funzione di iterazione ϕ_N e ϕ_2 sono entrambi convergenti, mentre quello con funzione ϕ_1 non lo è in quanto $|\phi_1'(x)| > 1$ in un intorno del punto fisso α.

2.6.1 Come arrestare un'iterazione di punto fisso

In generale, le iterazioni di punto fisso verranno arrestate quando il valore assoluto della *differenza fra due iterate* è minore di una tolleranza ε fissata. Essendo $\alpha = \phi(\alpha)$ e $x^{(k+1)} = \phi(x^{(k)})$, usando il teorema del valor medio (introdotto nella Sezione 1.6.3) troviamo

$$\alpha - x^{(k+1)} = \phi(\alpha) - \phi(x^{(k)}) = \phi'(\xi^{(k)})(\alpha - x^{(k)}) \text{ con } \xi^{(k)} \in I_{\alpha,x^{(k)}},$$

essendo $I_{\alpha,x^{(k)}}$ l'intervallo di estremi α e $x^{(k)}$.

Usando l'identità $\alpha - x^{(k)} = (\alpha - x^{(k+1)}) + (x^{(k+1)} - x^{(k)})$, concludiamo che

$$\alpha - x^{(k)} = \frac{1}{1 - \phi'(\xi^{(k)})}(x^{(k+1)} - x^{(k)}). \tag{2.27}$$

Di conseguenza, se $\phi'(x) \simeq 0$ in un intorno di α, l'errore viene stimato accuratamente dalla differenza fra due iterate consecutive. Questo accade per tutti i metodi di ordine 2 e quindi, in particolare, per il metodo di Newton. In caso contrario, tanto più ϕ' è prossimo a 1, tanto più stimare l'errore con la differenza fra le iterate sarà insoddisfacente.

Esempio 2.10 Calcoliamo con il metodo di Newton lo zero $\alpha = 1$ della funzione $f(x) = (x-1)^{m-1}\log(x)$ per $m = 11$ e $m = 21$. Questo zero ha molteplicità

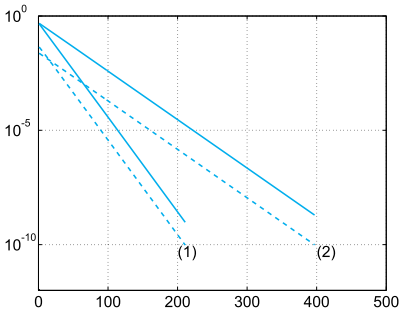

Figura 2.10. Valori assoluti degli errori (*in linea continua*) e delle differenze fra iterate (*in linea tratteggiata*) al variare delle iterazioni per il caso dell'Esempio 2.10: le curve (1) si riferiscono a $m = 11$, mentre le (2) a $m = 21$

pari a m. In tal caso l'ordine di convergenza del metodo di Newton decade a 1; inoltre, si può provare (si veda l'Esercizio 2.17) che $\phi'_N(\alpha) = 1 - 1/m$, essendo ϕ_N la funzione di iterazione del metodo stesso, visto come iterazione di punto fisso. Quindi, al crescere di m, la stima dell'errore fornita dalla differenza fra le iterate diventa sempre meno affidabile. È quello che si verifica sperimentalmente: nei grafici della Figura 2.10 vengono paragonati gli errori e la differenza fra le iterate in valore assoluto per $m = 11$ e $m = 21$. Come si vede lo scarto fra le due quantità è maggiore per $m = 21$. ■

2.7 Accelerazione con il metodo di Aitken

In questa Sezione illustriamo una tecnica che consente di accelerare la convergenza di una successione ottenuta a partire da iterazioni di punto fisso. Supponiamo pertanto che $x^{(k)} = \phi(x^{(k-1)})$, $k \geq 1$. Se la successione $\{x^{(k)}\}$ converge *linearmente* ad un punto fisso α di ϕ, dalla (2.24) si ricava che, per k fissato, dovrà esistere un valore λ (da determinare) tale che

$$\phi(x^{(k)}) - \alpha = \lambda(x^{(k)} - \alpha), \tag{2.28}$$

dove volutamente non abbiamo identificato $\phi(x^{(k)})$ con $x^{(k+1)}$. L'idea del metodo di Aitken consiste infatti nel definire un nuovo valore per $x^{(k+1)}$ (e, di conseguenza, una nuova successione) che sia un'approssimazione di α migliore di quella data da $\phi(x^{(k)})$. In effetti, dalla (2.28) ricaviamo

$$\alpha = \frac{\phi(x^{(k)}) - \lambda x^{(k)}}{1 - \lambda} = \frac{\phi(x^{(k)}) - \lambda x^{(k)} + x^{(k)} - x^{(k)}}{1 - \lambda},$$

ovvero

$$\boxed{\alpha = x^{(k)} + (\phi(x^{(k)}) - x^{(k)})/(1 - \lambda)} \tag{2.29}$$

Si tratta a questo punto di calcolare λ. Per fare questo introduciamo la seguente successione

$$\lambda^{(k)} = \frac{\phi(\phi(x^{(k)})) - \phi(x^{(k)})}{\phi(x^{(k)}) - x^{(k)}} \tag{2.30}$$

e verifichiamo che vale la seguente proprietà:

Lemma 2.1 *Se la successione di elementi $x^{(k+1)} = \phi(x^{(k)})$ converge a α, allora $\lim_{k\to\infty} \lambda^{(k)} = \phi'(\alpha)$.*

Dimostrazione. Se $x^{(k+1)} = \phi(x^{(k)})$, allora $x^{(k+2)} = \phi(\phi(x^{(k)}))$ e quindi, dalla (2.30), si ricava $\lambda^{(k)} = (x^{(k+2)} - x^{(k+1)})/(x^{(k+1)} - x^{(k)})$, cioè

$$\lambda^{(k)} = \frac{x^{(k+2)} - \alpha - (x^{(k+1)} - \alpha)}{x^{(k+1)} - \alpha - (x^{(k)} - \alpha)} = \frac{\frac{x^{(k+2)}-\alpha}{x^{(k+1)}-\alpha} - 1}{1 - \frac{x^{(k)}-\alpha}{x^{(k+1)}-\alpha}},$$

da cui, passando al limite e ricordando la (2.24), si perviene alla tesi

$$\lim_{k\to\infty} \lambda^{(k)} = \frac{\phi'(\alpha) - 1}{1 - 1/\phi'(\alpha)} = \phi'(\alpha).$$

■

Grazie al Lemma 2.1 possiamo concludere che, per k fissato, $\lambda^{(k)}$ può essere considerato come un'approssimazione del valore incognito λ, introdotto in precedenza. Utilizziamo allora la (2.30) nella (2.29) e definiamo un nuovo $x^{(k+1)}$ nel modo seguente

$$x^{(k+1)} = x^{(k)} - \frac{(\phi(x^{(k)}) - x^{(k)})^2}{\phi(\phi(x^{(k)})) - 2\phi(x^{(k)}) + x^{(k)}}, \quad k \geq 0 \tag{2.31}$$

Questa espressione è nota come *formula di estrapolazione di Aitken* e può essere considerata come *nuova* iterazione di punto fisso in cui si ponga come funzione di iterazione

$$\phi_\Delta(x) = \frac{x\phi(\phi(x)) - [\phi(x)]^2}{\phi(\phi(x)) - 2\phi(x) + x}$$

(tale metodo è noto talvolta anche con il nome di *metodo di Steffensen*).

Evidentemente la funzione ϕ_Δ è indeterminata per $x = \alpha$ in quanto tanto il numeratore che il denominatore si annullano. Tuttavia, assumendo che ϕ sia derivabile con $\phi'(\alpha) \neq 1$ ed applicando la formula di de l'Hôpital si trova

$$\begin{aligned}\lim_{x\to\alpha} \phi_\Delta(x) &= \frac{\phi(\phi(\alpha)) + \alpha\phi'(\phi(\alpha))\phi'(\alpha) - 2\phi(\alpha)\phi'(\alpha)}{\phi'(\phi(\alpha))\phi'(\alpha) - 2\phi'(\alpha) + 1} \\ &= \frac{\alpha + \alpha[\phi'(\alpha)]^2 - 2\alpha\phi'(\alpha)}{[\phi'(\alpha)]^2 - 2\phi'(\alpha) + 1} = \alpha.\end{aligned}$$

Di conseguenza, $\phi_\Delta(x)$ può essere estesa per continuità in $x = \alpha$ con $\phi_\Delta(\alpha) = \alpha$. Quando $\phi(x) = x - f(x)$ il caso $\phi'(\alpha) = 1$ corrisponde ad una radice di molteplicità almeno 2 per f (in quanto $\phi'(\alpha) = 1 - f'(\alpha)$). Anche in questa situazione si può però dimostrare, passando al limite, che $\phi_\Delta(\alpha) = \alpha$. Infine, si può anche verificare che i punti fissi di ϕ_Δ sono tutti e soli i punti fissi di ϕ.

Il metodo di Aitken può essere quindi applicato ad un metodo di punto fisso qualsiasi. Vale infatti il seguente teorema:

Teorema 2.2 *Siano $x^{(k+1)} = \phi(x^{(k)})$ le iterazioni di punto fisso (2.20), con $\phi(x) = x - f(x)$, per l'approssimazione delle radici di f. Allora, se f è sufficientemente regolare abbiamo che:*

- *se le iterazioni di punto fisso convergono linearmente ad una radice semplice di f, allora il metodo di Aitken converge quadraticamente alla stessa radice;*
- *se le iterazioni di punto fisso convergono con ordine $p \geq 2$ ad una radice semplice di f, allora il metodo di Aitken converge alla stessa radice con ordine $2p - 1$;*
- *se le iterazioni di punto fisso convergono linearmente ad una radice di molteplicità $m \geq 2$ di f, allora il metodo di Aitken converge linearmente alla stessa radice con un fattore di convergenza asintotico $C = 1 - 1/m$.*

In particolare, se $p = 1$ e la radice di f è semplice il metodo di estrapolazione di Aitken converge anche se le corrispondenti iterazioni di punto fisso divergono.

Nel Programma 2.5 riportiamo un'implementazione del metodo di Aitken. In esso `phi` è un *function handle* associato alla funzione ϕ di iterazione del metodo di punto fisso cui viene applicata la tecnica di estrapolazione di Aitken. Il dato iniziale viene precisato nella variabile `x0`, mentre `tol` e `kmax` sono rispettivamente la tolleranza sul criterio d'arresto (sul valore assoluto della differenza fra due iterate consecutive) ed il numero massimo di iterazioni consentite. Se non precisati, vengono assunti i valori di *default* pari a `kmax=100` e `tol=1.e-04`.

Programma 2.5. aitken: il metodo di Aitken

```
function [x,niter]=aitken(phi,x0,tol,kmax,varargin)
%AITKEN Estrapolazione di Aitken
%   [ALPHA,NITER]=AITKEN(PHI,X0) calcola un'appros-
%   simazione di un punto fisso ALPHA della funzione
%   PHI a partire dal dato iniziale X0 con il metodo
%   di estrapolazione di Aitken. Il metodo si arresta
%   dopo 100 iterazioni o dopo che il valore assoluto
%   della differenza fra due iterate consecutive e'
```

```
%    minore di 1.e-04. PHI puo' essere una anonymous
%    function, o una function definita in un M-file.
%    [ALPHA,NITER]=AITKEN(PHI,X0,TOL,KMAX) consente di
%    definire la tolleranza sul criterio d'arresto ed
%    il numero massimo di iterazioni.

if nargin == 2
    tol = 1.e-04;
    kmax = 100;
elseif nargin == 3
    kmax = 100;
end
x = x0;
diff = tol + 1;
k = 0;
while k < kmax && diff >= tol
    gx = phi(x,varargin{:});
    ggx = phi(gx,varargin{:});
    xnew = (x*ggx-gx^2)/(ggx-2*gx+x);
    diff = abs(x-xnew);
    x = xnew;
    k = k  + 1;
end
niter=k;
if (k==kmax && diff>tol)
    fprintf([' Il metodo non converge nel numero',...
             ' massimo di iterazioni\n ']);
end
```

Esempio 2.11 Per il calcolo della radice semplice $\alpha = 1$ della funzione $f(x) = e^x(x-1)$ applichiamo il metodo di Aitken a partire dalle due seguenti funzioni di iterazione

$$\phi_0(x) = \log(xe^x), \qquad \phi_1(x) = \frac{e^x + x}{e^x + 1}.$$

Utilizziamo il Programma 2.5 con `tol=1.e-10`, `kmax=100`, `x0=2` e definiamo le due funzioni di iterazione come segue

```
phi0 = @(x)log(x*exp(x));
phi1 = @(x)(exp(x)+x)/(exp(x)+1);
```

A questo punto eseguiamo il Programma 2.5 nel modo seguente

```
[alpha,niter]=aitken(phi0,x0,tol,kmax)

alpha =
   1.0000 + 0.0000i
niter =
    10

[alpha,niter]=aitken(phi1,x0,tol,kmax)

alpha =
     1
niter =
     4
```

Come si vede la convergenza del metodo è estremamente rapida (per confronto il metodo di punto fisso con funzione di iterazione ϕ_1 e con lo stesso criterio d'arresto avrebbe richiesto 18 iterazioni, mentre il metodo corrispondente a ϕ_0 non sarebbe stato convergente in quanto $|\phi_0'(1)| = 2$). ■

Si vedano gli Esercizi 2.17–2.21.

Riassumendo

1. Un valore α tale che $\phi(\alpha) = \alpha$ si dice punto fisso della funzione ϕ. Per il suo calcolo si usano metodi iterativi della forma $x^{(k+1)} = \phi(x^{(k)})$, che vengono detti iterazioni di punto fisso;
2. le iterazioni di punto fisso convergono sotto precise condizioni su ϕ e sulla sua derivata prima. Tipicamente la convergenza è lineare, diventa quadratica qualora $\phi'(\alpha) = 0$;
3. è possibile utilizzare le iterazioni di punto fisso anche per il calcolo degli zeri di una funzione f;
4. data un'iterazione di punto fisso $x^{(k+1)} = \phi(x^{(k)})$, anche non convergente, è sempre possibile costruire una nuova iterazione di punto fisso convergente tramite il metodo di Aitken.

2.8 Polinomi algebrici

In questa Sezione consideriamo il caso in cui f sia un polinomio di grado $n \geq 0$ della forma (1.9). Come già osservato, lo spazio di tutti i polinomi (1.9) viene indicato con il simbolo $\mathbb{P}_n$. Si ricorda che se $p_n \in \mathbb{P}_n$, per $n \geq 2$, e $a_k \in \mathbb{R}$, se $\alpha \in \mathbb{C}$ con $\mathrm{Im}(\alpha) \neq 0$ è una radice di p_n, allora lo è anche la sua complessa coniugata $\bar{\alpha}$.

Il teorema di Abel assicura che per ogni $n \geq 5$ non esiste una forma esplicita per calcolare tutti gli zeri di un generico polinomio p_n. Questo fatto motiva ulteriormente l'uso di metodi numerici per il calcolo delle radici di p_n.

Come abbiamo visto in precedenza per tali metodi è importante la scelta di un buon dato iniziale $x^{(0)}$ o di un opportuno intervallo di ricerca $[a, b]$ per la radice. Nel caso dei polinomi ciò è talvolta possibile sulla base dei seguenti risultati.

Teorema 2.3 (Regola dei segni di Cartesio) *Sia $p_n \in \mathbb{P}_n$. Indichiamo con ν il numero di variazioni di segno nell'insieme dei coefficienti $\{a_j\}$ e con k il numero di radici reali positive di p_n ciascuna contata con la propria molteplicità. Si ha allora che $k \leq \nu$ e $\nu - k$ è pari.*

Esempio 2.12 Il polinomio $p_6(x) = x^6 - 2x^5 + 5x^4 - 6x^3 + 2x^2 + 8x - 8$ ha come zeri $\{\pm 1, \pm 2i, 1 \pm i\}$ e quindi ammette una radice reale positiva ($k = 1$). In effetti, il numero di variazioni di segno ν dei coefficienti è 5 e quindi $k \leq \nu$ e $\nu - k = 4$ è pari. ■

Teorema 2.4 (di Cauchy) *Tutti gli zeri di p_n sono inclusi nel cerchio Γ del piano complesso*

$$\Gamma = \{z \in \mathbb{C} : \ |z| \leq 1 + \eta\}, \quad \textit{dove } \eta = \max_{0 \leq k \leq n-1} |a_k/a_n|. \tag{2.32}$$

Questa proprietà è di scarsa utilità quando $\eta \gg 1$ (per il polinomio p_6 dell'Esempio 2.12 si ha ad esempio $\eta = 8$, mentre le radici stanno tutte all'interno di cerchi di raggio decisamente minore).

2.8.1 Il metodo di Hörner

Illustriamo in questa Sezione un metodo per la valutazione efficiente di un polinomio (e della sua derivata) in un punto assegnato z. Tale algoritmo consente di generare un procedimento automatico, detto metodo di *deflazione*, per l'approssimazione progressiva di *tutte* le radici di un polinomio. Da un punto di vista algebrico la (1.9) è equivalente alla seguente rappresentazione

$$p_n(x) = a_0 + x(a_1 + x(a_2 + \ldots + x(a_{n-1} + a_n x)\ldots)). \tag{2.33}$$

Tuttavia, mentre la (1.9) richiede n addizioni e $2n - 1$ moltiplicazioni per valutare $p_n(x)$ (per x fissato), la (2.33) richiede solo n addizioni più n moltiplicazioni. L'espressione (2.33), nota anche come algoritmo delle moltiplicazioni annidate, sta alla base del metodo di Hörner. Quest'ultimo consente la valutazione efficiente del polinomio p_n in un punto z mediante il seguente algoritmo di *divisione sintetica*:

$$\begin{aligned} &b_n = a_n, \\ &b_k = a_k + b_{k+1} z, \quad k = n-1, n-2, \ldots, 0 \end{aligned} \tag{2.34}$$

Nella (2.34) tutti i coefficienti b_k con $k \leq n-1$ dipendono da z e possiamo verificare che $b_0 = p_n(z)$. Il polinomio

$$q_{n-1}(x; z) = b_1 + b_2 x + \ldots + b_n x^{n-1} = \sum_{k=1}^{n} b_k x^{k-1}, \tag{2.35}$$

di grado pari a $n-1$ nella variabile x, dipende dal parametro z (attraverso i coefficienti b_k) e si dice il *polinomio associato* a p_n. L'algoritmo (2.34)

è stato implementato nel Programma 2.6. I coefficienti a_j del polinomio da valutare sono memorizzati nel vettore `a` a partire da a_n fino ad a_0.

Programma 2.6. horner: il metodo di divisione sintetica

```
function [y,b] = horner(a,z)
%HORNER Metodo di Horner
%   Y=HORNER(A,Z) calcola
%   Y = A(1)*Z^N + A(2)*Z^(N-1) + ... + A(N)*Z + A(N+1)
%   con il metodo di divisione sintetica di Horner.
n = length(a)-1; b = zeros(n+1,1);
b(1) = a(1);
for j=2:n+1
   b(j) = a(j)+b(j-1)*z;
end
y = b(n+1); b = b(1:end-1);
```

Vogliamo a questo punto introdurre un algoritmo efficiente che, nota una radice di un polinomo (od una sua approssimazione), sia in grado di eliminarla e consentire quindi il calcolo della successiva fino all'esaurimento di tutte le radici.

A questo proposito conviene ricordare la seguente proprietà sulla *divisione tra polinomi*:

Proposizione 2.3 *Dati due polinomi $h_n \in \mathbb{P}_n$ e $g_m \in \mathbb{P}_m$ con $m \leq n$, esistono un unico polinomio $\delta \in \mathbb{P}_{n-m}$ ed un unico polinomio $\rho \in \mathbb{P}_{m-1}$ tali che*

$$h_n(x) = g_m(x)\delta(x) + \rho(x). \tag{2.36}$$

Dividendo allora un polinomio $p_n \in \mathbb{P}_n$ per $x - z$, grazie alla (2.36) si deduce che

$$p_n(x) = b_0 + (x - z)q_{n-1}(x; z),$$

avendo indicato con q_{n-1} il quoziente e con b_0 il resto della divisione. Se z è una radice di p_n, allora si ha $b_0 = p_n(z) = 0$ e quindi $p_n(x) = (x - z)q_{n-1}(x; z)$. In tal caso l'equazione algebrica $q_{n-1}(x; z) = 0$ fornisce le $n-1$ radici restanti di $p_n(x)$. Questa osservazione suggerisce di adottare il seguente procedimento, detto di *deflazione*, per il calcolo di *tutte* le radici di p_n.

Per $m = n, n-1, \ldots, 1$:

1. si trova una radice r_m di p_m con un opportuno metodo di approssimazione;
2. si calcola $q_{m-1}(x; r_m)$ tramite le (2.34)–(2.35) (posto $z = r_m$);
3. si pone $p_{m-1} = q_{m-1}$.

Nella Sezione che segue proponiamo il metodo più noto di questa famiglia, che utilizza per l'approssimazione delle radici il metodo di Newton.

2.8.2 Il metodo di Newton-Hörner

Come suggerisce il nome, il *metodo di Newton-Hörner* implementa il procedimento di deflazione appena descritto facendo uso del metodo di Newton per il calcolo delle radici r_m. Il vantaggio risiede nel fatto che l'implementazione del metodo di Newton sfrutta convenientemente l'algoritmo di Hörner (2.34). Infatti, se q_{n-1} è il polinomio associato a p_n definito nella (2.35), poiché

$$p'_n(x) = q_{n-1}(x; z) + (x - z)q'_{n-1}(x; z),$$

si ha

$$p'_n(z) = q_{n-1}(z; z).$$

Grazie a questa identità il metodo di Newton-Hörner per l'approssimazione di una radice (reale o complessa) r_j di p_n $(j = 1, \ldots, n)$ prende la forma seguente.

Data una stima iniziale $r_j^{(0)}$ della radice, calcolare per ogni $k \geq 0$ fino a convergenza

$$r_j^{(k+1)} = r_j^{(k)} - \frac{p_n(r_j^{(k)})}{p'_n(r_j^{(k)})} = r_j^{(k)} - \frac{p_n(r_j^{(k)})}{q_{n-1}(r_j^{(k)}; r_j^{(k)})} \tag{2.37}$$

A questo punto si utilizza la tecnica di deflazione, sfruttando il fatto che $p_n(x) = (x - r_j)p_{n-1}(x)$. Si può quindi passare all'approssimazione di uno zero di p_{n-1} e così via sino all'esaurimento di tutte le radici di p_n.

Si tenga conto che quando $r_j \in \mathbb{C}$ è necessario condurre i calcoli in aritmetica complessa, prendendo $r_j^{(0)}$ con parte immaginaria non nulla. In caso contrario, infatti, il metodo di Newton-Hörner genererebbe una successione $\{r_j^{(k)}\}$ di numeri reali.

Il metodo di Newton-Hörner è stato implementato nel Programma 2.7. I coefficienti a_j del polinomio del quale si intendono calcolare le radici sono memorizzati nel vettore `a` a partire da a_n fino ad a_0. Gli altri parametri di input, `tol` e `kmax`, sono rispettivamente la tolleranza sul criterio d'arresto (sul valore assoluto della differenza fra due iterate consecutive) ed il numero massimo di iterazioni consentite. Se non diversamente precisati, vengono assunti i valori di *default* pari a `kmax=100` e `tol=1.e-04`. In output, il programma restituisce nei vettori `radici` e `iter` le radici calcolate ed il numero di iterazioni che è stato effettuato per calcolare ciascun valore.

Programma 2.7. newtonhorner: il metodo di Newton-Hörner

```
function [radici,iter]=newtonhorner(a,x0,tol,kmax)
%NEWTONHORNER Metodo di Newton-Horner
%   [RADICI,ITER]=NEWTONHORNER(A,X0) calcola le
%   radici del polinomio
%    P(X) = A(1)*X^N + A(2)*X^(N-1) + ...
%                                   + A(N)*X + A(N+1)
%   con il metodo di Newton-Horner a partire dal dato
%   iniziale X0. Il metodo si arresta per ogni radice
%   al massimo dopo 100 iterazioni o dopo che il valore
%   assoluto della differenza fra due iterate conse-
%   cutive e' minore di 1.e-04.
%   [RADICI,ITER]=NEWTONHORNER(A,X0,TOL,KMAX) consente
%   di definire la tolleranza sul criterio d'arresto
%   ed il numero massimo di iterazioni.

if nargin == 2
    tol = 1.e-04;
    kmax = 100;
elseif nargin == 3
    kmax = 100;
end
n=length(a)-1;
radici = zeros(n,1);
iter = zeros(n,1);

for k = 1:n
    % Iterazioni di Newton
    niter = 0;
    x = x0;
    diff = tol + 1;
    while niter < kmax && diff >= tol
        [pz,b] = horner(a,x);
        [dpz,b] = horner(b,x);
        xnew = x - pz/dpz;
        diff = abs(xnew-x);
        niter = niter + 1;
        x = xnew;
    end

    if (niter==kmax && diff> tol)
      fprintf([' Il metodo non converge nel numero',...
               ' massimo di iterazioni\n ']);
    end
    % Deflazione
    [pz,a] = horner(a,x);
    radici(k) = x;
    iter(k) = niter;
end
```

Osservazione 2.2 Onde minimizzare la propagazione degli errori di arrotondamento, nel processo di deflazione conviene approssimare per prima la radice r_1 di modulo minimo e procedere quindi al calcolo delle successive radici $r_2, r_3, \ldots$, sino a quella di modulo massimo (per approfondimenti si veda ad esempio [QSSG14]). ■

Esempio 2.13 Richiamiamo il Programma 2.7 per calcolare le radici $\{1, 2, 3\}$ del polinomio $p_3(x) = x^3 - 6x^2 + 11x - 6$. Usiamo le seguenti istruzioni

```
a=[1 -6 11 -6];
[x,niter]=newtonhorner(a,0,1.e-15,100)

x =
      1
      2
      3

niter =
      8
      8
      2
```

Come si vede il metodo calcola accuratamente ed in poche iterazioni tutte e tre le radici. Come notato nell'Osservazione 2.2 non sempre il metodo è però così efficiente.

Ad esempio, per il calcolo delle radici del polinomio $p_4(x) = x^4 - 7x^3 + 15x^2 - 13x + 4$ (che presenta una radice pari a 1 con molteplicità 3 ed una radice semplice pari a 4) si trovano i seguenti risultati

```
a=[1 -7 15 -13 4]; format long;
[x,niter]=newtonhorner(a,0,1.e-15,100)

x =
   1.000006935337374
   0.999972452635761
   1.000020612232168
   3.999999999794697

niter =
    61
   100
     6
     2
```

dai quali risulta un evidente deterioramento nell'accuratezza del calcolo della radice multipla. In effetti si può dimostrare che questa perdita di accuratezza è tanto maggiore quanto più grande è la molteplicità della radice. Più in generale, si può dimostrare (si veda [QSSG14, Cap. 6]) che il problema del calcolo di radici di una funzione f diventa mal condizionato (ovvero, molto sensibile a perturbazioni sui dati) qualora la derivata f' sia prossima a zero nelle radici. Per un esempio, si veda l'Esercizio 2.6. ■

2.9 Cosa non vi abbiamo detto

I metodi più complessi per il calcolo accurato degli zeri di una generica funzione si ottengono combinando fra loro diversi algoritmi. Segnaliamo a questo proposito il comando `fzero` (già introdotto nella Sezione 1.6.1) che utilizza il metodo di Dekker-Brent (si veda [QSS07, Cap. 6]). Nella sua forma più semplice a partire da un dato `x0`, `fzero(fun,x0)` calcola lo zero di una funzione `fun` più vicino a `x0`.

Ad esempio, risolviamo il problema dell'Esempio 2.1 anche con `fzero`, prendendo come valore iniziale `x0=0.3` (lo stesso che abbiamo scelto quando abbiamo usato il metodo di Newton). È sufficiente dare le seguenti istruzioni

```
M=6000; v=1000; f=@(r) M-v*(1+r)/r*((1+r)^5-1);
x0=0.3;
[alpha,res,flag,info]=fzero(f,x0);
```

per trovare la radice `alpha=0.06140241153653` ed un residuo pari a `res=-1.8190e-12` dopo 7 iterazioni e con 29 valutazioni funzionali. La variabile `info` è una cosiddetta struttura, che si compone di 5 sottocampi. In particolare nei campi `iterations` e `funcCount` della struttura `info` (ovvero in `info.iterations` ed in `info.funcCount`) vengono riportati rispettivamente il numero delle iterazioni svolte ed il numero delle valutazioni funzionali effettuate. Si noti che quando il parametro di uscita `flag` assume un valore negativo significa che `fzero` ha fallito nella ricerca dello zero. Per confronto, osserviamo che il metodo di Newton converge in 6 iterazioni al valore `alpha=0.06140241153653` con un residuo pari a `res=9.0949e-13` richiedendo tuttavia anche la conoscenza della derivata prima di f per un totale di 12 valutazioni funzionali.

Per il calcolo degli zeri di un polinomio oltre al metodo di Newton-Hörner citiamo i metodi basati sulle successioni di Sturm, il metodo di Müller (si vedano [Atk89], [Com95] o [QSSG14]), ed il metodo di Bairstow ([RR01], pag. 371 e seguenti). Un altro approccio consiste nel caratterizzare gli zeri di un polinomio come gli autovalori di una particolare matrice, detta *companion matrix*, e nel calcolare quest'ultimi con tecniche opportune. Questo è l'approccio adottato dalla funzione `roots` di MATLAB, introdotta nella Sezione 1.6.2.

Nella Sezione 2.5 abbiamo mostrato come si possa adattare il metodo di Newton al caso di sistemi di equazioni non lineari. In generale, ogni iterazione di punto fisso può essere facilmente estesa al calcolo delle radici di un sistema di equazioni non lineari.

`fsolve` L'istruzione MATLAB

```
zero=fsolve(@fun,x0)
```

permette di calcolare uno zero di un sistema non lineare definito attraverso la *user-defined function* `fun` (costruita dall'utilizzatore) e partendo da un vettore iniziale `x0`. La *user-defined function* `fun` restituisce i valori $f_i(\bar{x}_1, \ldots, \bar{x}_n)$, $i = 1, \ldots, n$, per ogni vettore in ingresso $(\bar{x}_1, \ldots, \bar{x}_n)^T$.

Ad esempio, per il sistema non lineare (2.17) la corrispondente *function* MATLAB, che noi chiamiamo `systemnl`, è

```
function fx=systemnl(x)
fx(1) = x(1)^2+x(2)^2-1;
fx(2) = sin(pi*0.5*x(1))+x(2)^3;
```

Le istruzioni MATLAB per risolvere il sistema dato sono allora

```
x0 = [1 1];
alpha=fsolve(@systemnl,x0)

alpha =
    0.4761   -0.8794
```

Usando questa procedura abbiamo trovato solo una delle due radici. L'altra può essere calcolata usando come dato iniziale `-x0`.

Osservazione 2.3 I comandi `fzero` e `fsolve` hanno la stessa funzionalità in MATLAB e in Octave, tuttavia le loro interfacce differiscono leggermente per quanto concerne gli input opzionali. Consigliamo al lettore di consultare la documentazione attraverso l'`help` per entrambi i comandi sia in ambiente MATLAB sia in Octave. ■

MAT‖OCT

2.10 Esercizi

Esercizio 2.1 Data la funzione $f(x) = \cosh x + \cos x - \gamma$, per $\gamma = 1, 2, 3$ si individui un intervallo contenente uno zero di f e lo si calcoli con il metodo di bisezione con una accuratezza pari a 10^{-10}.

Esercizio 2.2 (Equazione di stato di un gas) Per l'anidride carbonica (CO_2) i coefficienti a e b della (2.1) valgono rispettivamente $a = 0.401$ Pa m^6 e $b = 42.7 \cdot 10^{-6}$ m^3 (Pa sta per Pascal). Si trovi il volume occupato da 1000 molecole di anidride carbonica poste ad una temperatura $T = 300$ K e ad una pressione $p = 3.5 \cdot 10^7$ Pa utilizzando il metodo di bisezione con una accuratezza pari a 10^{-12} (la costante di Boltzmann è pari a $k = 1.3806503 \cdot 10^{-23}$ Joule K^{-1}).

Esercizio 2.3 Si consideri un piano la cui inclinazione varia con velocità costante ω. Su di esso si trova un oggetto che all'istante iniziale è fermo; dopo t secondi la sua posizione è

$$s(t, \omega) = \frac{g}{2\omega^2}[\sinh(\omega t) - \sin(\omega t)],$$

dove $g = 9.8$ m/s^2 è l'accelerazione di gravità. Supponiamo che il corpo si sia mosso di un metro in un secondo; si ricavi il corrispondente valore di ω con una accuratezza pari a 10^{-5}.

Esercizio 2.4 Si dimostri la disuguaglianza (2.6).

Esercizio 2.5 Nel Programma 2.1 per calcolare il punto medio è stata utilizzata la seguente istruzione: `x(2) = x(1)+(x(3)-x(1))*0.5`, invece della più naturale: `x(2) = (x(1)+x(3))*0.5`. Per quale motivo?

Esercizio 2.6 Si ripeta per il metodo di Newton l'Esercizio 2.1. Perché per $\gamma = 2$ il metodo risulta inaccurato?

Esercizio 2.7 Utilizzando il metodo di Newton si costruisca un algoritmo per il calcolo della radice quadrata di un numero positivo a. Si proceda in modo analogo per il calcolo della radice cubica di a.

Esercizio 2.8 Supponendo il metodo di Newton convergente, si dimostri la (2.9) con α radice semplice di $f(x) = 0$ e f derivabile due volte con continuità in un intorno di α.

Esercizio 2.9 (Statica) Si risolva il Problema 2.3, al variare di $\beta \in [0, 2\pi/3]$ e con una tolleranza pari a 10^{-5}, supponendo che le aste abbiano le seguenti lunghezze $a_1 = 10$ cm, $a_2 = 13$ cm, $a_3 = 8$ cm, $a_4 = 10$ cm, usando il metodo di Newton e richiedendo una tolleranza pari a 10^{-5}. Per ogni valore di β si considerino due possibili valori iniziali pari a $x^{(0)} = -0.1$ e a $x^{(0)} = 2\pi/3$.

Esercizio 2.10 Si osservi che la funzione $f(x) = e^x - 2x^2$ ha 3 zeri, $\alpha_1 < 0$ e α_2 e α_3 positivi. Per quali valori di $x^{(0)}$ il metodo di Newton converge a α_1?

Esercizio 2.11 Si applichi il metodo di Newton per il calcolo dello zero di $f(x) = x^3 - 3x^2 2^{-x} + 3x 4^{-x} - 8^{-x}$ in $[0, 1]$ e si analizzi sperimentalmente l'ordine di convergenza. La convergenza non risulta di ordine 2. Perché?

Esercizio 2.12 Un proiettile che viene lanciato ad una velocità v_0 con una inclinazione α in un tunnel di altezza h, raggiunge la massima gittata quando α è tale che $\sin(\alpha) = \sqrt{2gh/v_0^2}$, dove $g = 9.8$ m/s^2 è l'accelerazione di gravità. Si calcoli α con il metodo di Newton, quando $v_0 = 10$ m/s e $h = 1$ m.

Esercizio 2.13 (Piano di investimento) Si risolva, a meno di una tolleranza `tol=1.e-12`, il Problema 2.1 con il metodo di Newton, supponendo che $M = 6000$ euro, $v = 1000$ euro, $n = 5$ ed utilizzando un dato iniziale pari al risultato ottenuto dopo cinque iterazioni del metodo di bisezione sull'intervallo $(0.01, 0.1)$.

Esercizio 2.14 Un corridoio ha la pianta indicata in Figura 2.11. La lunghezza massima L di un'asta che possa passare da un estremo all'altro strisciando per terra è data da

$$L = l_2/(\sin(\pi - \gamma - \alpha)) + l_1/\sin(\alpha),$$

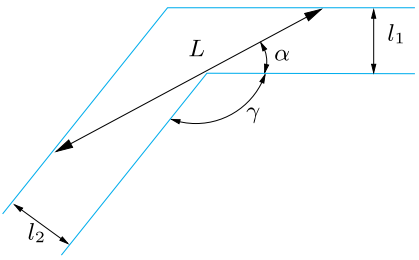

Figura 2.11. Problema dello scorrimento di un'asta in un corridoio

dove α è la soluzione della seguente equazione non lineare

$$l_2 \frac{\cos(\pi - \gamma - \alpha)}{\sin^2(\pi - \gamma - \alpha)} - l_1 \frac{\cos(\alpha)}{\sin^2(\alpha)} = 0. \tag{2.38}$$

Si determini α con il metodo di Newton quando $l_2 = 10$, $l_1 = 8$ e $\gamma = 3\pi/5$.

Esercizio 2.15 Si considerino i metodi di Newton e delle secanti per approssimare le radici dell'equazione $(1-x)/(100x^2-100x+26) = x/3$ nell'intervallo $[-2, 2]$. Localizzare le radici per via grafica e dire se esse sono semplici o multiple. Dopo aver posto la tolleranza per il test d'arresto pari a `tol=1.e-8` ed il numero massimo di iterazioni pari a `kmax=200`, si richiamino i metodi di Newton, prima con $x^{(0)} = 0.45$ e poi con $x^{(0)} = 0.55$, e quindi il metodo delle secanti, prima con $x^{(0)} = 0.45$, $x^{(1)} = 0.75$ e poi con $x^{(0)} = 0.5$, $x^{(1)} = 0.96$. I risultati numerici concordano con quanto afferma la teoria?

Esercizio 2.16 Si consideri il sistema di equazioni non lineari

$$\begin{cases} x^3 + y - 2x^2 - 2 = 0 \\ (x - 0.5)^2 - y^2 + x + \gamma = 0 \end{cases} \tag{2.39}$$

essendo γ un parametro reale assegnato. Per $\gamma = 2$ si localizzino le radici e le si classifichi in base alla molteplicità, quindi si calcolino tutte le radici del sistema dato con il metodo di Newton. Ripetere le stesse operazioni per $\gamma = 3.75$.

Esercizio 2.17 Verificare che, indicata al solito con ϕ_N la funzione di iterazione del metodo di Newton considerato come metodo di punto fisso, se α è uno zero di f di molteplicità m, allora $\phi_N'(\alpha) = 1 - 1/m$. Se ne deduca che il metodo di Newton converge quadraticamente se α è uno zero semplice di $f(x) = 0$, linearmente negli altri casi.

Esercizio 2.18 Si tracci il grafico della funzione $f(x) = x^3 + 4x^2 - 10$ e se ne deduca che essa ammette un unico zero reale α. Per il suo calcolo si usino le seguenti iterazioni di punto fisso: dato $x^{(0)}$, si definisce $x^{(k+1)}$ come

$$x^{(k+1)} = \frac{2(x^{(k)})^3 + 4(x^{(k)})^2 + 10}{3(x^{(k)})^2 + 8x^{(k)}}, \quad k \geq 0.$$

Se ne studi la convergenza a α.

Esercizio 2.19 Si studi la convergenza delle seguenti iterazioni di punto fisso

$$x^{(k+1)} = \frac{x^{(k)}[(x^{(k)})^2 + 3a]}{3(x^{(k)})^2 + a}, \quad k \geq 0,$$

per il calcolo della radice quadrata di un numero positivo a.

Esercizio 2.20 Si ripetano i calcoli effettuati nell'Esercizio 2.11 usando come criterio d'arresto quello sul residuo. È più accurato il risultato ottenuto ora o quello ricavato precedentemente?

Esercizio 2.21 Si vogliono approssimare le intersezioni tra le funzioni $f_1(x) = \frac{1}{3}\log\left(\frac{x}{\pi}\right) + x^2$ e $f_2(x) = x^2 + x - 2$. Localizzare per via grafica le intersezioni tra le due curve, quindi approssimare i punti di intersezione con il metodo di Newton, scegliendo opportunamente il dato iniziale. In un secondo momento proporre una o più funzioni di punto fisso $\phi(x)$, i cui punti fissi coincidono con le intersezioni cercate, ed analizzarne la convergenza.

http://www.springer.com/978-88-470-3952-0

Calcolo Scientifico
Esercizi e problemi risolti con MATLAB e Octave
Quarteroni, A.; Saleri, F.; Gervasio, P.
2017, XX, 523 pagg. 209 figg., 191 figg. a colori.,
Softcover
ISBN: 978-88-470-3952-0