Capitolo 2

Procedimenti iterativi

Molto spesso le leggi della natura sono non lineari. Ne segue la necessità di risolvere equazioni non lineari. Tranne che per polinomi di grado basso, per i quali è possibile ottenere la soluzione in forma esplicita, la stragrande maggioranza di questi problemi si risolve usando procedimenti iterativi di cui si comincia lo studio in questo capitolo.

2.1 Metodo delle bisezioni

Supponiamo di voler risolvere l'equazione lineare scalare:

$$f(x) = (a-1)x + b = 0. \tag{2.1}$$

A prescindere dal fatto che in questo caso, se $a \neq 1$, la soluzione $-b/(a-1)$ si ottiene facilmente, possiamo pensare di approssimare la stessa mediante un procedimento iterativo. Se per esempio $a = 0$ e $b = 1$, sappiamo che la soluzione si trova nell'intervallo $[-2,\ 0]$, infatti $f(-2) = -1$ e $f(0) = 1$, la funzione cambia di segno nell'intervallo. Questo ci assicura che lo zero appartiene all'intervallo $[-2,\ 0]$.

Teorema 2.1.1 *Teorema di Bolzano Se $f(x) \in C([a,b])$ ed $f(a)\,f(b) < 0$ allora esiste almeno uno zero reale α contenuto nell'intervallo $[a,b]$.*

Supponiamo che α sia l'unico zero contenuto nell'intervallo $[a,b]$. Posto $a_0 = a$, e $b_0 = b$, risulta $f(a_0)\,f(b_0) < 0$; si considera come prima stima di $\bar{x}$

il punto medio dell'intervallo, ovvero:

$$c_0 = \frac{a_0 + b_0}{2}.$$

Si calcola poi il valore che la funzione assume in tale punto:

1. se $f(c_0) = 0$ abbiamo trovato la soluzione,
2. se $f(a_0)\, f(c_0) < 0$ allora lo zero cade in $[a_0, c_0]$ e si definisce come nuovo intervallo $[a_1, b_1] = [a_0, c_0]$.
3. se $f(c_0)\, f(b_0) < 0$ allora lo zero cade in $[c_0, b_0]$ e si considera come nuovo intervallo $[a_1, b_1] = [c_0, b_0]$.
4. A questo punto si analizza nuovamente il comportamento della funzione nel punto medio
$$c_1 = \frac{a_1 + b_1}{2}$$
assunto come nuova approssimazione dello zero, ed il processo si ripete.

La procedura definita dal metodo delle bisezioni determina una sequenza di intervalli ciascuno dei quali è contenuto nel precedente

$$[a_n, b_n] \subseteq [a_{n-1}, b_{n-1}] \subseteq \ldots \subseteq [a_1, b_1] \subseteq [a_0, b_0];$$

il generico intervallo ha ampiezza pari alla metà di quella dell'intervallo precedentemente determinato e contiene lo zero α di $f(x)$.
Generalizzando, la procedura costruisce la successione dei punti medi

$$c_n = \frac{a_{n-1} + b_{n-1}}{2} \qquad n \geq 1$$

e se $f(c_n) \neq 0$, definisce i nuovi intervalli nel modo seguente

$$[a_n, b_n] = \begin{cases} [c_n, b_{n-1}] & \text{se } f(c_n)\, f(b_{n-1}) < 0 \\ [a_{n-1}, c_n] & \text{se } f(a_{n-1})\, f(c_n) < 0. \end{cases}$$

Il metodo delle bisezioni fornisce sempre uno zero α di f in $[a, b]$, quindi risulta essere sempre convergente. Per questo viene detto globalmente convergente. Definiamo

$$|e_n| = |c_n - \alpha|$$

allora

$$|e_n| \leq \frac{|b-a|}{2^{n+1}}$$

quindi

$$\lim_{n\to\infty} e_n = \alpha$$

e l'errore viene dimezzato ad ogni iterata. In particolare si può osservare che ad ogni iterata si guadagna una cifra binaria e quindi dopo 3.3 iterate circa si guadagnerà una cifra decimale esatta, essendo $3.3 \simeq \log_2 10$.

Pur essendo comunque convergente è necessario definire dei criteri di arresto che interrompano il succedersi delle iterazioni nel momento in cui è stata raggiunta una buona stima dello zero. Cioè quando l' errore relativo scende al di sotto di una certa tolleranza τ:

$$E_v = \left|\frac{c_n - \alpha}{\alpha}\right| < \tau.$$

È possibile calcolare un errore relativo approssimato, con la formula

$$E_a = \left|\frac{b_n - a_n}{\min(|a_n|, |b_n|)}\right|.$$

Quando E_a diventa minore di τ è possibile terminare il procedimento iterativo, infatti ogni volta che il metodo dà come approssimazione dello zero

$$c_n = \frac{a_n + b_n}{2}$$

si è certi che il valore esatto dello zero giace all'interno di un intervallo di ampiezza $|b_n - a_n|$. Il calcolo di E_a può provocare problemi quando uno degli estremi dell'intervallo è molto piccolo, in questo caso conviene utilizzare un criterio di arresto misto, che si ottiene con la formula seguente:

$$E_a = \left|\frac{a_n - b_n}{\max(1, \min(|a_n|, |b_n|))}\right|.$$

La tolleranza τ non deve essere scelta troppo piccola perchè a causa degli errori di arrotondamento le condizioni di arresto possono non essere mai soddisfatte così come accade quando τ è inferiore alla precisione di macchina. Il metodo delle bisezioni ha bisogno per poter essere implementato di due

punti in cui la funzione assume segno opposto, che definiscano l'intervallo iniziale. Una volta localizzato tale intervallo di osservazione, non importa quanto sia ampio, le iterazioni procedono fino a convergere allo zero α della funzione. La convergenza globale è sicuramente uno dei vantaggi forniti dal metodo. Tuttavia esso non può essere sempre applicato, come accade ad esempio per il calcolo di zeri di funzioni positive come $f(x) = x^2$ che non verificano le ipotesi su cui si fonda. A volte invece, pur verificandole, la funzione studiata presenta più zeri all'interno dell'intervallo iniziale. In tale situazione per ciascuno zero è necessario individuare un intervallo diverso, talvolta di non facile localizzazione perchè di ampiezza piuttosto piccola. Per risolvere questo problema intervengono i metodi cosiddetti "*localmente convergenti*" che per poter essere implementati hanno però bisogno di una stima iniziale prossima allo zero.

2.2 Iterazione Funzionale

Riscriviamo l'equazione $f(x) = (a-1)x + b = 0$ nella forma:

$$x = ax + b$$

Partendo da un punto iniziale x_0 si definisce il procedimento iterativo seguente:

$$x_{n+1} = ax_n + b. \tag{2.2}$$

Esso ammette la soluzione costante:

$$\alpha = -\frac{b}{a-1}$$

che è la soluzione da noi cercata. La successione $x_0, x_1, \ldots$ tenderà ad avvicinarsi ad α se $|a| < 1$. Infatti l'errore verifica

$$\begin{aligned} x_{n+1} - \alpha &= ax_n + b - a\alpha - b = a(x_n - \alpha) \\ &= a^2(x_{n-1} - \alpha) = \ldots = a^{n+1}(x_0 - \alpha) \end{aligned}$$

e tenderà a zero solo se $|a| < 1$. Ovviamente data una equazione come la (2.1) si possono definire dei procedimenti iterativi in infiniti modi. Ad esempio si può pensare di esprimere $a - 1$ come la differenza di due numeri a_1 ed a_2 e scrivere:

$$a_1 x = a_2 x + b,$$

da cui si ottiene il procedimento iterativo:

$$x_{n+1} = \frac{a_2}{a_1} x_n + \frac{b}{a_1}. \tag{2.3}$$

Per avere la convergenza, ovviamente alla stessa radice, bisognerà richiedere che $|a_2/a_1| < 1$.

Esempio 2.2.1 Sia da cercare la radice di $3x - 5 = 0$. Si può porre $3 = 2 + 1$ e quindi definire il procedimento:

$$x_{n+1} = -\frac{1}{2} x_n + \frac{5}{2}.$$

Partendo da $x_0 = 0$, si ha la successione $x_1 = 2.5$, $x_2 = 1.25$, $x_3 = 1.875, \ldots$ che converge alla soluzione $\alpha = 1.666...$

I procedimenti iterativi del tipo considerato possono porsi nella forma più generale:

$$x_{n+1} = \phi(x_n) \tag{2.4}$$

ove la funzione ϕ, detta funzione iteratrice, è continua. Se questo procedimento converge verso un punto α allora deve essere $\alpha = \phi(\alpha)$. Infatti:

$$\alpha = \lim_{n\to\infty} x_{n+1} = \lim_{n\to\infty} \phi(x_n)$$

e per la continuità della funzione ϕ:

$$\alpha = \lim_{n\to\infty} \phi(x_n) = \phi(\lim_{n\to\infty} x_n) = \phi(\alpha).$$

Se siamo interessati a cercare una radice α della equazione $f(x) = 0$, nella costruzione della funzione iteratrice $\phi(x)$ dobbiamo fare in modo che $\alpha = \phi(\alpha)$, cioè che α sia una soluzione costante della (2.4).

I procedimenti iterativi hanno una interessante interpretazione grafica. Si disegna in un piano cartesiano la bisettrice del primo e terzo quadrante e la curva $y = \phi(x)$. Partendo dal punto iniziale x_0 sull'asse orizzontale, si traccia la verticale fino ad incontrare la curva. Da questo punto si traccia

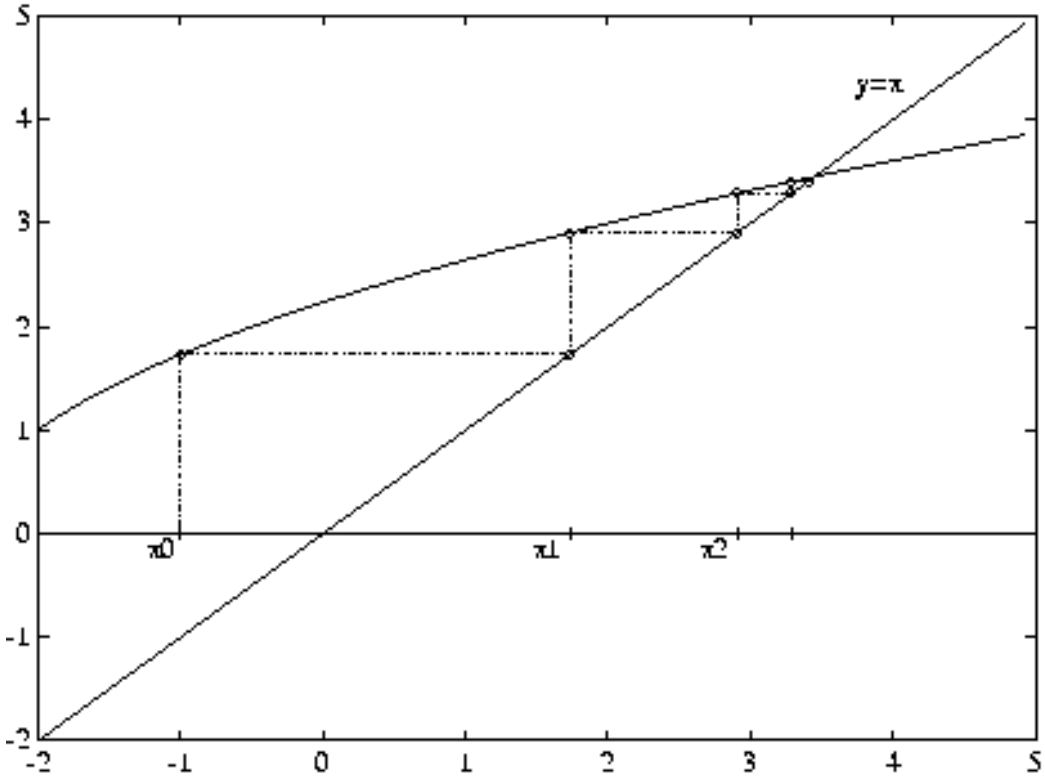

Figura 2.1: Procedimento iterativo $x_{n+1} = \phi(x_n) \equiv (2x_n + 5)^{1/3}$

la parallela all'asse x fino ad incontrare la bisettrice. L'ascissa del punto di intersezione è il nuovo punto x_1. Ripetendo la costruzione si ottengono gli altri punti (Fig. 2.1).

Se l'equazione da risolvere è non lineare, allora i procedimenti iterativi possono diventare indispensabili, sia perchè i metodi diretti, quando esistono, diventano costosi, sia perchè, molto spesso, i metodi diretti non esistono. Ovviamente anche nel caso non lineare, per ogni equazione, vi possono essere infiniti procedimenti iterativi appropriati. È necessario, quindi, avere una guida nella scelta del più opportuno procedimento iterativo. Il seguente teorema può servire allo scopo. Esso è una semplice generalizzazione del criterio seguito nel caso lineare in cui si chiedeva $|a + 1| < 1$. Infatti in tal caso la funzione iteratrice $\phi(x)$ era lineare con $\phi'(x) = a$.

Teorema 2.2.1 *Sia $x_{n+1} = \phi(x_n)$ un procedimento iterativo tale che $\alpha = \phi(\alpha)$ e con $\phi(x)$ continuamente differenziabile in un intorno I_0 di α. Se*

$|\phi'(\alpha)| < \lambda < 1$, *partendo da un opportuno intorno della radice, il procedimento converge ad* α.

Dimostrazione. Per la continuità di $\phi'(x)$ in I_0, esiste un altro intorno della radice, contenuto in I_0, che indicheremo con I, in cui si ha $|\phi'(x)| \leq \lambda < 1$. Supponendo che $x_0 \in I$, si ha

$$|x_1 - \alpha| = |\phi(x_0) - \phi(\alpha)| \leq |\phi'(\xi_0)||x_0 - \alpha|$$

con $\xi \in I$. Essendo $|\phi'(\xi_0)| \leq \lambda < 1$, si ha che $x_1 \in I$.

Allo stesso modo si dimostra che tutti i successivi punti sono in I. Inoltre si ha

$$|x_{n+1} - \alpha| = |\phi(x_n) - \phi(\alpha)| \leq \lambda|x_n - \alpha| \leq \lambda^n|x_0 - \alpha|$$

e quindi

$$\lim_{n\to\infty} |x_{n+1} - \alpha| = \lim_{n\to\infty} \lambda^n|x_0 - \alpha| = 0$$

e il metodo converge.

Per poter usare questo teorema è necessaria una conoscenza approssimata sulla localizzazione della radice. Ciò nonostante esso è spesso molto utile.

Esempio 2.2.2 Sia da calcolare la radice positiva di $f(x) \equiv x^3 - 2x - 5$. Poichè $f(1.5) < 0$, ed $f(2.5) > 0$, la radice deve trovarsi nell'intervallo (1.5, 2.5). La funzione iteratrice più immediata $\phi(x) = 0.5(x^3 - 5)$, che definisce il procedimento iterativo:

$$x_{n+1} = 0.5(x_n^3 - 5),$$

non è appropriata perché nell'intervallo (1.5, 2.5) è sempre $|\phi'(x)| > 1$ e quindi non è soddisfatta l'ipotesi del precedente teorema. La funzione iteratrice $\phi(x) = (2x_n + 5)^{1/3}$, che definisce il procedimento iterativo:

$$x_{n+1} = (2x_n + 5)^{1/3}, \tag{2.5}$$

invece soddisfa l'ipotesi del teorema essendo, nell'intervallo considerato, $1/6 < \phi'(x) < 0.15$. Il procedimento è quindi convergente.

Esercizio 2.2.1 Implementare sul calcolatore i due procedimenti iterativi precedenti e determinare la radice.

Dopo aver trovato, con l'aiuto del teorema precedente uno o più procedimenti iterativi convergenti, è necessario ancora distinguere tra questi in base ad altri criteri. Il principale di tali criteri è la rapidità con cui la successione delle iterate converge. Infatti non tutti i procedimenti iterativi convergenti raggiungono la radice con la stessa rapidità: alcuni potrebbero essere così lenti da richiedere ore di elaborazione, altri frazioni di secondo. Nel caso lineare è quasi ovvio che tutto dipende dal valore di $|a+1|$. Infatti l'errore, definito da $e_n = x_n - \alpha$, soddisfa l'equazione:

$$e_{n+1} = ae_n = a^{n-1}e_0$$

e quindi l'errore tenderà a zero tanto più velocemente quanto più piccolo sarà $|a|$. Nel caso non lineare la rapidità di convergenza dipenderà da $|\phi'(\alpha)|$.

2.3 Il Metodo di Newton

Nel caso non lineare è possibile definire dei procedimenti iterativi che convergono ancora più rapidamente alla soluzione. Ciò si ha quando $\phi'(\alpha) = 0$. In tal caso i procedimenti iterativi si dicono *superlineari*. Ad esempio se si suppone che nell'intorno della radice α, a cui siamo interessati, la funzione $f(x)$ abbia derivata non nulla, la funzione iteratrice $\phi(x) = x - f(x)/f'(x)$ definisce il procedimento iterativo:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{2.6}$$

detto *metodo di Newton*. È facile verificare che nelle radici di $f(x)$ in cui $f'(x)$ sia diversa da zero, la derivata prima di $\phi(x)$ si annulla. Infatti, $\phi'(x) = 1-(f'(x)^2-f(x)f''(x))/f'(x)^2$ e nel punto α, in cui $f(\alpha) = 0$, si ha $\phi'(\alpha) = 0$. Ciò significa che in un apportuno intorno della radice la convergenza del metodo è superlineare. Anche tra i metodi con convergenza superlineare vi sono metodi che convergono più rapidamente di altri. Cioè è possibile definire un opportuno parametro che distingue tra di loro questi metodi. Questo parametro è l'*ordine di convergenza* così definito:

Sia $x_0, x_1, \ldots,$ la successione ottenuta mediante il metodo iterativo (2.4). Essa converge con ordine $p \geq 1$ se esiste $c > 0$ tale che

$$\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|^p} = c$$

Nel caso $p = 1$ si ha convergenza lineare ed in tal caso c deve essere strettamente minore di 1.

Mostriamo che il metodo di Newton, quando la funzione $f(x)$ ha derivata prima non nulla e derivata seconda continua in un intorno della radice, ha ordine 2. Infatti si ha, essendo ξ_n un opportuno punto tra la radice ed x_n,

$$x_{n+1} - \alpha = \phi(x_n) - \phi(\alpha) = \phi'(\alpha)(x_n - \alpha) + \frac{1}{2}\phi''(\xi_n)(x_n - \alpha)^2$$

da cui, essendo $\phi'(\alpha) = 0$, si ha:

$$|e_{n+1}| = \frac{1}{2}|\phi''(\xi_n)||e_n|^2.$$

Se si pone $c = (1/2)\max_\xi |\phi''(\xi)| \equiv (1/2)\max_\xi |f''(\xi)/f'(\xi)|$, si ottiene facilmente che $p = 2$.

Esempio 2.3.1 Consideriamo la funzione dell'esempio 2.2.2. Nella tabella seguente sono riportati i risultati ottenuti con il procedimento iterativo (2.5) e (2.6).

n	x_n (2.5)	x_n (2.6)
0	2.5	2.5
1	2.1544	2.1642
2	2.1036	2.9710
3	2.0959	2.0946
4	2.0948	2.0946
5	2.0946	2.0946

Il metodo di Newton ha una interessante interpretazione geometrica. Sia $y = f(x)$ la curva definita dalla funzione di cui si vuol trovare una radice. Ovviamente le radici sono le intersezioni della curva con l'asse x. Sia x_0 il punto iniziale, a cui corrisponde sulla curva il punto $(x_0, f(x_0))$. La retta tangente alla curva passante per tale punto ha equazione:

$$y = f(x_0) + f'(x_0)(x - x_0).$$

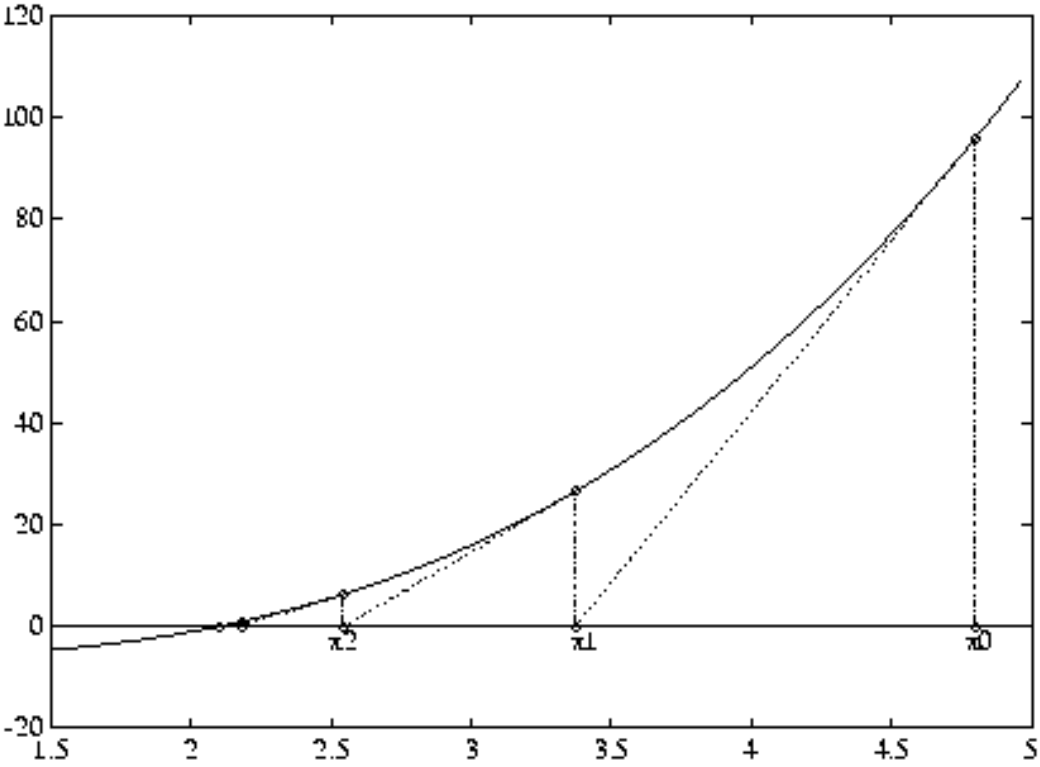

Figura 2.2: Metodo di Newton relativo all'esempio 2.2.2

Essa interseca l'asse x nel punto x_1 dato da:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

che è proprio il primo punto ottenuto dal metodo di Newton (Fig. 2.2). Per questo motivo il metodo di Newton è anche chiamato metodo delle tangenti.

Esercizio 2.3.1 Mediante la funzione plot, disegnare per punti la funzione $f(x) \equiv x^3 - 2x - 5$ e localizzarene le radici.

Come risulta dall'esercizio precedente la funzione plot può essere molto utile per localizzare una radice e per scegliere il punto iniziale per innescare un procedimento iterativo.

2.4 Caso di radici multiple

Finora abbiamo supposto che $f'(\alpha)$ sia non nulla in α. Se ciò non accade significa che la radice è almeno doppia. Una radice è multipla di molteplicità q per la funzione $f(x)$ se essa può mettersi nella forma $f(x) = (x-\alpha)^q g(x)$, con $g(\alpha) \neq 0$. È facile verificare che in tal caso in α si annullano tutte le derivate di $f(x)$ fino all'ordine $q-1$, mentre $f^{(q)}(x) \neq 0$. Naturalmente vale anche il viceversa, come ci si può convincere usando lo sviluppo in serie di Taylor nell'intorno di α.

Nel caso in cui α abbia molteplicità maggiore di 1, l'ordine del metodo di Newton si abbassa e la convergenza diventa lineare. Si dimostra facilmente infatti che in tal caso si ha:

$$\phi'(\alpha) = 1 - \frac{1}{q} \tag{2.7}$$

Se si conosce a priori la molteplicità q, allora il procedimento iterativo:

$$x_{n+1} = x_n - q\frac{f(x_n)}{f'(x_n)} \tag{2.8}$$

ha ordine di convergenza 2.

Esercizio 2.4.1 Verificare la (2.7).

Esercizio 2.4.2 Dimostrare che il procedimento iterativo (2.8) ha ordine di convergenza 2 nel caso di radici multiple con molteplicità q.

Esercizio 2.4.3 Scrivere la funzione Matlab per il procedimento iterativo (2.8) e calcolare la radice doppia positiva di $x^3 + x^2 - 33x + 63$.

2.5 Criteri di stop

Per decidere quando fermare un procedimento iterativo è necessario ovviamente decidere a priori con quante cifre significative esatte si vuole approssimare la radice. Sappiamo già che quando la radice è diversa da zero, è l'errore relativo che fornisce tale informazione. Ad esempio se la radice fosse $\alpha = 1234567$, l'approssimazine $x = 1230000$ avrebbe tre cifre esatte, ma l'errore assoluto è $|\alpha - x| = 4567$ che è molto grande, mentre l'errore relativo è:

$$\left|\frac{\alpha - x}{\alpha}\right| = 3.69 \cdot 10^{-3}.$$

Se invece la radice fosse $\alpha = 0.1234567 \cdot 10^{-2}$ e l'approssimazione $x = 0.1230000 \cdot 10^{-2}$, l'errore assoluto sarebbe $0.4567 \cdot 10^{-5}$, molto piccolo, mentre l'errore relativo sarebbe lo stesso di prima. Quindi l'indicatore più appropriato in entrambi i casi è l'errore relativo. Poichè la radice α è incognita, non è possibile purtroppo usare questo indicatore. Bisogna scegliere una grandezza calcolabile che si avvicini il più possibile ad esso. Tenendo conto che:

$$x_{n+1} - \alpha = \phi(x_n) - \phi(\alpha) = \phi'(\xi)(x_n - \alpha),$$

con ξ un punto compreso fra x_n e α, si ha:

$$x_{n+1} - x_n = (\phi'(\xi) - 1)(x_n - \alpha),$$

da cui si ha:

$$\left|\frac{\alpha - x_n}{\alpha}\right| = \left|\frac{x_{n+1} - x_n}{\alpha(\phi'(\xi) - 1)}\right| = \tag{2.9}$$

$$\left|\frac{1}{\alpha(\phi'(\xi) - 1)}\right| |x_{n+1} - x_n| \leq \tag{2.10}$$

$$\left|\frac{1}{(\phi'(\xi) - 1)}\right| \left|\frac{x_{n+1} - x_n}{\min(|x_n|, |x_{n+1}|)}\right|. \tag{2.11}$$

Nell'ottenere l'ultima espressione si è tenuto conto che, ovviamente, quando il procedimento converge, i punti x_n sono molto vicini alla radice. Dalla (2.10) si osserva che solo nel caso in cui:

$$\left|\frac{1}{\alpha(\phi'(\xi) - 1)}\right| \tag{2.12}$$

è dell'ordine dell'unità ha senso usare la differenza tra due iterate successive al posto dell'errore relativo. Infatti se si vuole una precisione dell'ordine di $\tau = 0.001$, si potrebbe in tal caso continuare il procedimento fino a quando non sia:

$$|x_{n+1} - x_n| < \tau \tag{2.13}$$

Nel caso in cui il valore che la (2.12) assume nell'intorno della radice sia molto diverso da 1, l'uso della (2.13) potrebbe portare o all'arresto precoce del procedimento, quando cioè l'approssimazione della radice non è ancora buona (caso in cui la (2.12) assume valori molto grandi), oppure all'arresto tardivo del procedimento. Il primo caso si verifica, ad esempio, se $\phi'(x)$ è vicino al valore 1 nell'intorno della radice e quindi interessa solamente procedimenti iterativi con convergenza lineare. Per il metodo di Newton quindi l'uso del criterio di arresto (2.12) è giustificato nel caso in cui la radice non sia troppo grande o troppo piccola (escluso lo zero) in valore assoluto. Un criterio migliore è quello che si ottiene dalla (2.11), cioè:

$$\left| \frac{x_{n+1} - x_n}{\min(|x_n|, |x_{n+1}|)} \right| < \epsilon \tag{2.14}$$

Anche la (2.14) ha il difetto di non rappresentare bene l'errore relativo quando $\phi'(x)$ è vicino ad uno nell'intorno della radice. Per i metodi superlineari invece va molto bene.

Un altro criterio, anch'esso molto usato, è quello di arrestare il procedimento quando:

$$|f(x_n)| < \epsilon. \tag{2.15}$$

Essendo $f(x_n) = f'(\xi)(x_n - \alpha)$, con $\xi \in (\alpha, x_n)$, si ha:

$$\left| \frac{\alpha - x_n}{\alpha} \right| = \left| \frac{f(x_n)}{\alpha f'(\xi)} \right|$$

da cui si deduce che il test (2.15) è valido se $|\alpha f'(\xi)|$ non è troppo grande o troppo piccolo.

2.6 Metodi quasi-newtoniani

Uno dei problemi del metodo di Newton è che richiede il calcolo della derivata prima di f ad ogni iterazione, infatti il calcolo della derivata può essere costoso o complesso, oppure quando si conosce la funzione f solo come risultato di un lungo calcolo numerica, una formula per la derivata può non essere disponibile. Un modo per risolvere questo problema è considerare il seguente procedimento iterativo

$$x_{n+1} = x_n - \frac{f(x_n)}{g_n}$$

con g_n una approssimazione di $f'(x_n)$. Questi metodi vengono chiamati quasi-newtoniani. Un esempio è il metodo delle secanti, in cui la derivata è approssimata dal rapporto incrementale

$$g_n = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

un'altro esempio è dato dal metodo della direzione costante in cui g_n è costante e il procedimento itarativo diventa:

$$x_{n+1} = x_n - \frac{f(x_n)}{g} = \phi(x_n)$$

Se calcoliamo $\phi'(x) = 1 - \dfrac{f'(x_n)}{g}$ vediamo che il procedimento iterativo è convergente se

$$|\phi'(\alpha)| = |1 - \frac{f'(\alpha)}{g}| < 1$$

e il metodo converge linearmente quando $g \neq f'(\alpha)$; se $g = f'(\alpha)$ la convergenza è superlineare.

2.7 Metodo delle secanti

Abbiamo visto che dal metodo di Newton si può immediatamente ricavare un altro metodo. Basta sostituire alla derivata $f'(x_n)$ una sua approssimazione, cioè il rapporto:

$$\frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}.$$

Si ottiene quindi:

$$x_{n+1} = x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n). \tag{2.16}$$

Questo procedimento iterativo non rientra nella classe dei procedimenti iterativi visti finora, cioè del tipo (2.4). In questo caso il nuovo punto dipende da due punti precedenti, cioè il procedimento iterativo è del tipo:

$$x_{n+1} = \phi(x_{n-1}, x_n). \tag{2.17}$$

Un procedimento di questo tipo necessita, per poter partire, di due punti iniziali.

L'interpretazione geometrica del procedimento iterativo (2.16) è semplice. Il nuovo punto è l'intersezione con l'asse x della retta secante passante per i punti $(x_{n-1}, f(x_{n-1}))$ e $(x_n, f(x_n))$ e per questo motivo viene detto metodo delle secanti (Fig. 2.3).

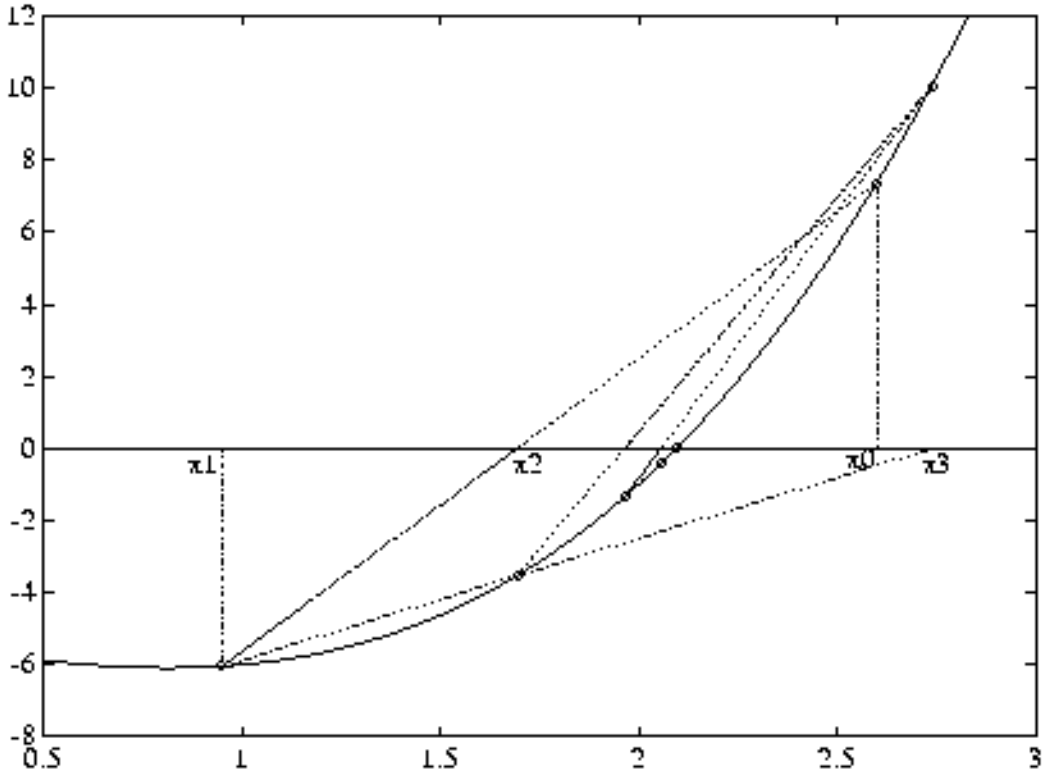

Figura 2.3: Metodo delle secanti relativo all'esempio 2.2.2

Per quel che riguarda la velocità di convergenza, il metodo delle secanti è superlineare nel caso in cui la funzione f abbia derivata seconda continua nell'intorno della radice. Infatti, posto $e_n = x_n - \alpha$, si ha:

$$e_{n+1} = \frac{f(x_n)e_{n-1} - f(x_{n-1})e_n}{f(x_n) - f(x_{n-1})} = \frac{1}{2}\frac{f''(\xi)}{f'(\eta)} e_n e_{n-1},$$

essendo ξ ed η opportuni punti nell'intorno della radice. Se il metodo converge si ha $\lim_{n\to\infty} e_{n-1} = 0$ e quindi:

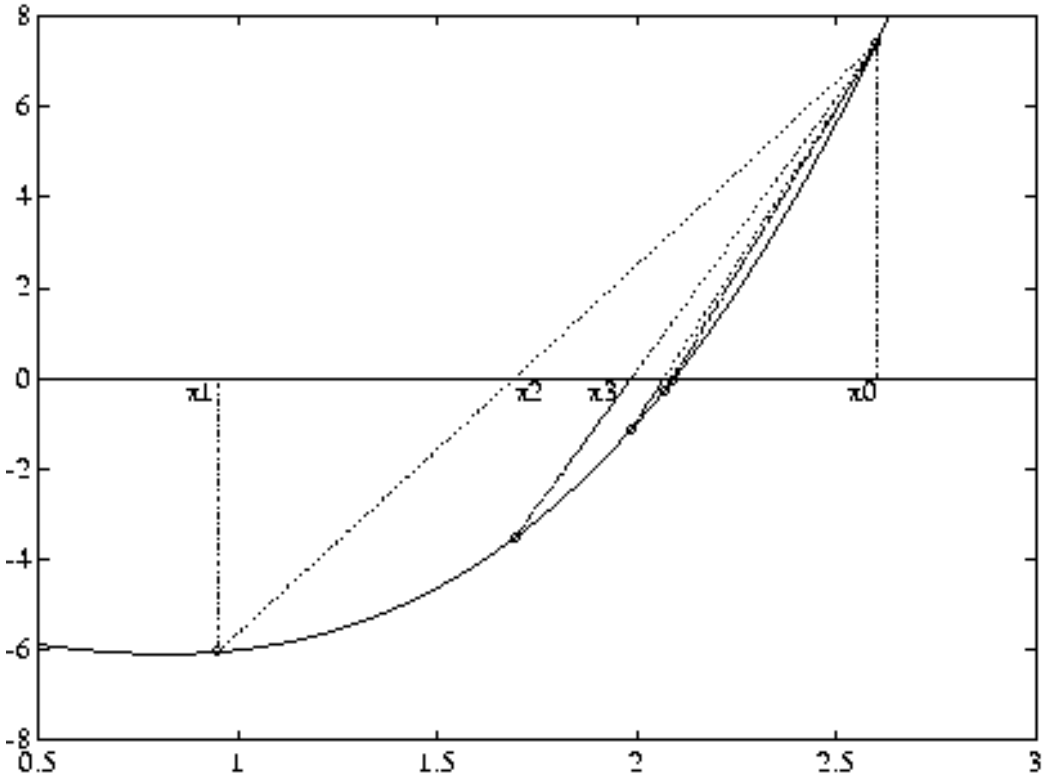

Figura 2.4: Metodo della falsa posizione relativo all'esempio 2.2.2

$$\lim_{n\to\infty} \frac{|e_{n+1}|}{|e_n|} = 0$$

da cui segue la superlinearità. Si può dimostrare che in questo caso l'ordine è $p = 1.618$. È un po' meno veloce del metodo di Newton ma ha il vantaggio che ad ogni passo calcola solo $f(x_n)$.

La stessa idea può essere usata per ottenere un procedimento iterativo ad un punto. Basta tenere fisso uno dei due punti, per esempio x_0. Si ottiene quindi:

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

Questo metodo, detto *regula falsi* o metodo della falsa posizione, è lineare e quindi meno veloce del metodo delle secanti (Fig. 2.4).

2.8 Algoritmi ibridi

Esistono procedimenti iterativi ad uno o più punti con ordine di convergenza più elevato. Quelli che abbiamo riportato sono i più importanti ed i più comunemente usati. D'altra parte non esiste, per una generica funzione $f(x)$, il metodo iterativo ideale che sia veloce con convergenza garantita quando si parte da un punto iniziale generico. Possiamo cercare di sfruttare la convergenza globale del metodo delle bisezioni combinandola con la velocità del metodo delle secanti. Le idee di base sono riunite in un unico algoritmo (noto come algoritmo di Brent) implementato nella funzione Matlab `fzero`.

FZERO

Questa funzione accetta in input due parametri. Il primo è il nome della funzione di cui si vuol calcolare la radice, il secondo è un punto iniziale. Ad esempio se si vuol calcolare la radice di $x^3 - 2x - 5$*, si definisce la funzione, che chiameremo* **cub.m***, mediante il file cub.m:*

```
function y = cub(x)
y = x ^3 - 2*x - 5;
```

e si usa la **fzero***, prendendo come punto iniziale x0 =* -2*,*

```
≫ fzero('cub',x0)
```

La risposta è una approssimazione della radice 2.0946.

Una semplificazione dell'algoritmo di Brent è la seguente. Data una funzione f e un intervallo $[a, b]$ con $f(a)f(b) < 0$, e quindi contenente una radice, iniziamo effettuando una iterazione del metodo delle secanti, usando gli estremi dell'intervallo come punti iniziali. Continuiamo l'iterazione con il metodo delle secanti fino a quando le iterate sono contenute nell'intervallo. Contemporanemente aggiorniamo l'intervallo sfruttando le iterate successivamente calcolate e scegliendo sempre l'intervallo più piccolo in cui la funzione cambia di segno. Se l'iterata non cade nell'intervallo eseguiamo un passo del metodo delle bisezioni, in modo da avvicinarsi alla radice e riprovare con il

metodo delle secanti. La successione x_k viene aggiornata in modo da considerare sempre le approssimazioni più accurate della radice. Un possibile algoritmo è il seguente:
Dato un intervallo iniziale $[a, b] = [a_1, b_1]$, con $f(a_1)f(b_1) < 0$, poni $k = 1$, $x_0 = a_1$, $x_1 = b_1$ esegui:

1. calcola $c = x_k - f(x_k)(x_k - x_{k-1})/(f(x_k) - f(x_{k-1})$
2. Se $c < a_k$ o $c > b_k$ allora esegui il metodo delle bisezioni
   calcola $c = a_k + (b_k - a_k)/2$
   Se $f(a_k)f(c) < 0$, allora $a_{k+1} = a_k, b_{k+1} = c, x_{k+1} = c, x_k = a_k$
   Se $f(c)f(b_k) < 0$ allora $a_{k+1} = c, b_{k+1} = b_k, x_{k+1} = c, x_k = b_k$
3. altrimenti aggiorna l'intervallo usando il passo delle secanti
   Se $f(a_k)f(c) < 0$, allora $a_{k+1} = a_k, b_{k+1} = c, x_{k+1} = c$
   Se $f(c)f(b_k) < 0$ allora $a_{k+1} = c, b_{k+1} = b_k, x_{k+1} = c$
4. $k = k + 1$, vai al passo 1.

Naturalmente questo algoritmo va implementato considerando un opportuno criterio di arresto basato su una stima dell'errore relativo.

Eseguiamo adesso alcune function Matlab per trovare lo zero di una funzione. La function `bzero` implementa il metodo delle bisezioni, la function `nzero` il metodo di Newton, la function `szero` il metodo delle secant e la function `bszero` il metodo ibrido appena descritto. Tutte le function tranne la `nzero` accettano come dati di input: la funzione, l'intervallo $[a, b]$ che contiene lo zero e una variabile strutturata con dei dati opzionali. La function `nzero` ha come dati di input: la funzione, la derivata prima, il punto iniziale della succession x_0 e una variabile strutturata con dei dati opzionali. I dati di output sono: la soluzione x, $f(x)$, un flag che informa sul risultato (se e' maggiore di zero la soluzione è stata trovata correttamente, altrimenti si è verificato un errore), una struttura contenente alcune informazioni di output, come il numero di valutazioni di funzioni, il numero di iterate e l'algoritmo utilizzato.

Usiamo la seguente funzione test:

```
>> f = inline('2*x*exp(-15)-2*exp(-15*x)+1')
f =
     Inline function:
     f(x) = 2*x*exp(-15)-2*exp(-15*x)+1

>> f(0)
ans =
    -1
>> f(1)
ans =
     1
```

Poichè la funzione cambia di segno agli estremi dell'intervallo $[0, 1]$ possiamo usare il metodo delle bisezioni:

```
>> options=struct('TolX',1e-10,'Display','iter','Nmax',100);
>> [x,fx,exitflag,output]=bzero(f,[0 1],options)
Val. funzioni Iterationi            x           f(x)
         3          1             0.5      0.998894
         4          2            0.25      0.952965
         5          3           0.125       0.69329
         6          4          0.0625      0.216789
         7          5         0.03125     -0.251568
         8          6        0.046875    0.00992823
         9          7       0.0390625     -0.113168
        10          8       0.0429688     -0.049817
        11          9       0.0449219    -0.0195068
        12         10       0.0458984   -0.00468151
        13         11       0.0463867    0.00265011
        14         12       0.0461426   -0.00100899
        15         13       0.0462646   0.000822236
        16         14       0.0462036  -9.29574e-05
        17         15       0.0462341   0.000364744
        18         16       0.0462189    0.00013592
        19         17       0.0462112   2.14876e-05
        20         18       0.0462074  -3.57333e-05
        21         19       0.0462093  -7.12241e-06
        22         20       0.0462103   7.18271e-06
        23         21       0.0462098    3.0176e-08
```

```
        24         22       0.0462096  -3.54611e-06
        25         23       0.0462097  -1.75796e-06
        26         24       0.0462098  -8.63894e-07
        27         25       0.0462098  -4.16859e-07
        28         26       0.0462098  -1.93341e-07
        29         27       0.0462098  -8.15827e-08
        30         28       0.0462098  -2.57033e-08
        31         29       0.0462098   2.23635e-09
        32         30       0.0462098  -1.17335e-08
        33         31       0.0462098  -4.74857e-09
        34         32       0.0462098  -1.25611e-09
        35         33       0.0462098   4.90123e-10
        36         34       0.0462098  -3.82992e-10
x =
   4.6210e-02
fx =
  -3.8299e-10
exitflag =
     1
output =
     algorithm: 'bisezioni'
     funcCount: 36
    iterations: 35
>>
```

Il numero di iterate è piuttosto elevato, proviamo quindi ad usare un metodo più veloce come il metodo di Newton o delle secanti:

```
>>df = inline('2*exp(-15)+30*x*exp(-15*x)')
df =
     Inline function:
     df(x) = 2*exp(-15)+30*x*exp(-15*x)
>>
>> [x,fx,exitflag,output]=nzero(f,df,0,options);
Val. funzioni Iterationi            x           f(x)
         3          1     1.63451e+06             2
         5          2    -1.63451e+06          -Inf
         7          3             NaN           NaN
>> [x,fx,exitflag,output]=szero(f,[0 1],options);
```

```
Val. funzioni Iterationi            x           f(x)
         3          1             0.5      0.998894
         4          2         -451.136          -Inf
         5          3              NaN           NaN
```

Entrambi non riescono a trovare la soluzione, il metodo misto riesce a risolvere il problema:

```
>> [x,fx,exitflag,output]=bszero(f,[0 1],options);
Val. funzioni Iterationi            x           f(x)
         3          1             0.5      0.998894  S
         5          2            0.25      0.952965  B
         6          3        0.128011      0.706832  S
         8          4       0.0640053      0.234275  B
         9          5       0.0518566     0.0812135  S
        10          6       0.0454105    -0.0120611  S
        11          7       0.0462441   0.000513718  S
        12          8         0.04621   3.07437e-06  S
        13          9       0.0462098  -7.89951e-10  S
        14         10       0.0462098   1.33227e-15  S
>> [x,fx,exitflag,output]=fzero(f,[0 1],options);

 Func-count      x           f(x)         Procedure
    1              0            -1        initial
    2              1             1        initial
    3            0.5      0.998894        bisection
    4           0.25      0.952965        bisection
    5          0.125       0.69329        bisection
    6         0.0625      0.216789        bisection
    7      0.0411481    -0.0788823        interpolation
    8      0.0468446    0.00947653        interpolation
    9      0.0462336   0.000357385        interpolation
   10      0.0462098  -8.47183e-08        interpolation
   11      0.0462098   1.51421e-11        interpolation
   12      0.0462098  -2.98486e-09        interpolation
Zero found in the interval: [0, 1].
```

Per le funzioni polinomiali esistono metodi ad hoc, oltre naturalmente a quelli già visti che possono essere usati in generale. Molti di questi metodi

hanno ormai solo un interesse storico. Attualmente per i polinomi vi sono metodi molto efficienti che tendono a fornire tutte le radici contemporaneamente e sono basati su concetti di cui non ci siamo ancora occupati come, ad esempio, la soluzione di sistemi non lineari, il calcolo di autovalori di matrici ecc. Citeremo, perché è utile nelle applicazioni, soltanto la funzione Matlab **roots**, senza approfondire, per ora, il metodo su cui è basata.

ROOTS

Questa funzione calcola le radici di un polinomio i cui coefficienti sono gli elementi del vettore C, cioè del polinomio:

$$C_1 x^N + C_2 x^{N-1} + \ldots + C_N x + C_{N+1}$$

Prendendo in considerazione il polinomio dell'esempio precedente, le seguenti istruzioni Matlab permettono di calcolare le radici del polinomio $x^3 - 2x - 5$ *eseguendo la funzione roots:*

```
≫   C = [ 1 0 -2 -5];
≫   roots(C)
ans  =
       2.0946
      −1.0473   1.1359i
      −1.0473 −1.1359i
```

La prima è la radice già trovata precedentemente, le altre due sono una coppia di radici complesse e coniugate.
