Capitolo 2

Metodi numerici per zeri di funzioni

2.1 Introduzione

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. Riscriviamo l'equazione nella forma:

$$x = ax + b$$

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

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

Questo procedimento ammette la soluzione costante:

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

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:

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

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 2.1.1 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 $\alpha = 1.666...$

2.2 Iterazione Funzionale

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 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).

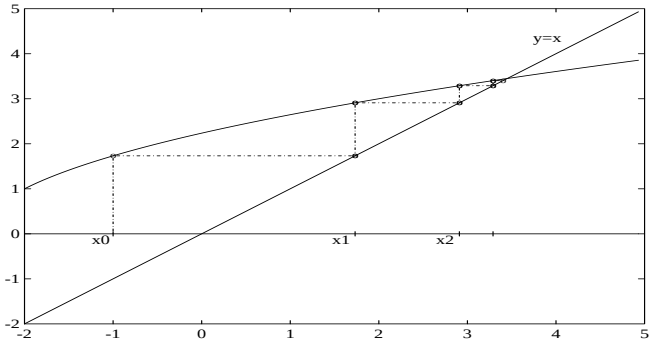

Figura 2.1: Procedimento iterativo $x_{n+1} = \phi(x_n) \equiv (2x_n + 5)^{1/3}$

Se l'equazione da risolvere è non lineare, allora i procedimenti iterativi possono diventare indispensabili, sia perché i metodi diretti, quando esistono, diventano costosi, sia perché, 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 $|a| < 1$. 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 $|\phi'(pha)| < \lambda < 1$, 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 $|\phi'(x)| < \lambda < 1$. Supponendo che $x_0 \in I$, si ha

$$|x_1 - \alpha| = |\phi(x_0) - \phi(\alpha)| < |\phi'(\xi)||x_0 - \alpha|$$

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

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

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

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 2.2.1 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 $|a|$. Infatti, considerato che l'errore, definito da $e_n = x_n - \alpha$, soddisfa l'equazione:

$$e_{n+1} = ae_n$$

si ha $e_n = a^{n-1}e_0$ 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 $|\phi'(\alpha)|$.

È possibile definire un opportuno parametro che effettua questa distinzione. Questo parametro è l'*ordine di convergenza*.

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 definitivamente

$$|e_{n+1}| \leq c|e_n|^p.$$

Nel caso $p = 1$ la convergenza si dice lineare; in tal caso c deve essere strettamente minore di 1. Se $p = 2$ la convergenza si dice quadratica.

Se la funzione ϕ è sufficientemente regolare in un intorno del punto fisso α, l'ordine di convergenza p è legato al valore delle derivate successive della ϕ, come il seguente teorema mostra.

Teorema 2.2.2 *Sia $\phi(x)$ derivabile p volte in un intorno di α con derivata p-esima continua. Allora la successione generata dalla (2.4) ha ordine di convergenza p se e solo se risulta*

$$\phi'(\alpha) = \phi''(\alpha) = \ldots\ldots = \phi^{(p-1)}(\alpha) = 0, \quad e \quad \phi^{(p)}(\alpha) \neq 0.$$

Dimostrazione. È sufficiente considerare lo sviluppo della serie di Taylor di ϕ in un intorno del punto fisso α, arrestato all'ordine p:

$$\phi(x) = \phi(\alpha) + \phi'(\alpha)(x-\alpha) + \ldots \frac{\phi^{(p-1)}(\alpha)}{(p-1)!}(x-\alpha)^{p-1} + \frac{\phi^{(p)}(\xi)}{p!}(x-\alpha)^p$$

dove ξ è un opportuno punto tra x ed α. Si ha che

$$e_{n+1} = x_{n+1} - \alpha = \phi(x_n) - \phi(\alpha) = \frac{\phi^{(p)}(\xi)}{p!}(x_n - \alpha)^p$$

da cui la tesi.□

Come conseguenza osserviamo che se $\phi'(\alpha) \neq 0$, la convergenza del metodo è lineare, mentre se $\phi'(\alpha) = 0$, la convergenza è almeno quadratica o superlineare. Un esempio classico di un metodo di ordine $p = 2$ è il metodo di Newton (o Newton-Rhapson) descritto di seguito.

2.3 Il Metodo di Newton

Supponiamo che 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 negli zeri 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 opportuno intorno dello zero la convergenza del metodo è superlineare.

Mostriamo che il metodo di Newton (2.6), quando la funzione $f(x)$ ha derivata prima non nulla e derivata seconda continua in un intorno dello zero, ha ordine 2. Infatti si ha, essendo ξ_n un opportuno punto tra lo zero 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 $p = 2$ (avremmo potuto anche verificare che $\phi''(\alpha) \neq 0$).

Esempio 2.3.1 Consideriamo la funzione dell'esempio (2.2.1) . Nella tabella seguente sono riportati i risultati ottenuti con il procedimento iterativo (**??**) e (2.6).

n	x_n	x_n
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 uno zero. Ovviamente gli zeri 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:

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

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.

2.4 Criteri di stop

Per decidere quando fermare un procedimento iterativo è necessario ovviamente decidere a priori con quante cifre significative esatte si vuole approssimare lo zero. Sappiamo già che quando lo zero è non nullo, è l'errore relativo che fornisce tale informazione. Ad esempio se lo zero 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:

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

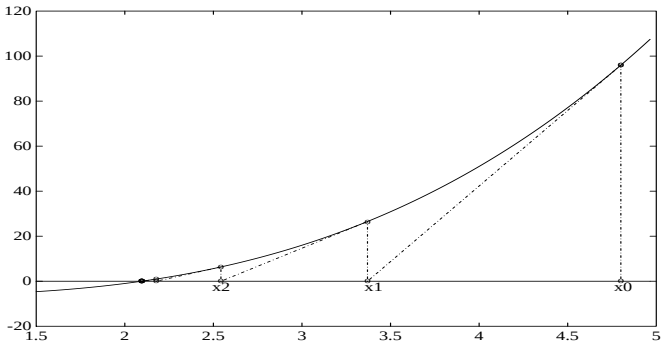

Figura 2.2: Metodo di Newton relativo all'esempio 2.2.1

Se invece lo zero 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 lo zero α è incognito, 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:

$$\left|\frac{\alpha - x_n}{\alpha}\right| = \left|\frac{x_{n+1} - x_n}{\alpha(\phi'(\xi) - 1)}\right| = \tag{2.7}$$

$$\left|\frac{1}{\alpha(\phi'(\xi) - 1)}\right| |x_{n+1} - x_n| \simeq \tag{2.8}$$

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

Nell'ottenere l'ultima espressione si è tenuto conto che, ovviamente, quando il procedimento converge, i punti x_n sono molto vicini allo zero.

Se si conosce una maggiorazione $K < 1$ del modulo di $\phi'(x)$ in un intorno di α, cioè se esiste $r > 0$ tale che

$$\forall x \in]\alpha - r, \alpha + r[: \qquad |\phi'(x)| \leq K < 1$$

allora è possibile arrestare la procedura quando

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

Uno studio a priori di $\phi'(x)$ in un intorno di α non è sempre possibile, o comunque può risultare complesso. Si può utilizzare in tali casi la condizione più semplice ma meno precisa:

$$\left| \frac{x_{n+1} - x_n}{\min(|x_n|, |x_{n+1}|)} \right| < \epsilon. \tag{2.10}$$

La (2.10) ha il difetto di non rappresentare bene l'errore relativo quando $\phi'(x)$ è vicino ad uno nell'intorno dello zero. Per i metodi superlineari (come il metodo di Newton) va invece molto bene.

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

$$|f(x_n)| < \epsilon. \tag{2.11}$$

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 (2.11) è valido se $|\alpha f'(\xi)|$ non è troppo grande o troppo piccolo.

2.5 Metodi quasi Newtoniani

Uno dei problemi del metodo di Newton è che richiede il calcolo della derivata prima di f a ogni iterazione. Questo può essere un problema quando

- La derivata può essere costosa.
- La funzione f può essere data con una formula complessa, e quindi è difficile da calcolare.

- La funzione f si conosce solo come risulatao di un lungo calcolo numerica e una formula per la derivata non è 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 é 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.

2.6 Il metodo delle secanti

Il metedo delle secanti, definito dalla iterazione

$$\begin{aligned} x_{n+1} &= x_n - \frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})} f(x_n) \\ &\equiv \frac{f(x_n)x_{n-1} - f(x_{n-1})x_n}{f(x_n) - f(x_{n-1})}. \end{aligned}$$

non rientra nella classe dei procedimenti $x_{n+1} = \phi(x_n)$, ma il nuovo punto dipende da due punti precedenti, cioè:

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

e necessita, per poter partire, di due punti iniziali.

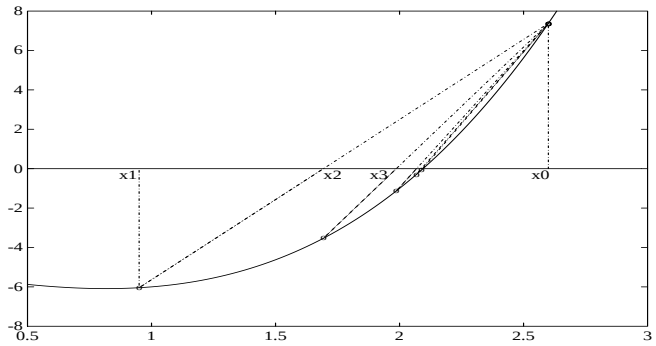

Figura 2.3: Metodo della falsa posizione relativo all'esempio 2.2.1

L'interpretazione geometrica del metodo delle secanti è semplice. Il nuovo punto è l'ascissa del punto di intersezione con l'asse x della retta secante passante per i punti
$(x_{n-1}, f(x_{n-1}))$ e $(x_n, f(x_n))$.

Il metodo delle secanti è superlineare nel caso in cui la funzione f abbia derivata seconda continua nell'intorno dello zero.
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* è lineare e quindi meno veloce del metodo delle secanti. (Fig. 2.3).

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.

2.7 Metodo delle successive bisezioni

Tutti i metodi finora visti sono, in generale, solo localmente convergenti. Esistono solo alcune funzione f, che verificano particolari ipotesi, per cui è possibile dimostrare una convergenza globale, cioè per qualsiasi punto iniziale. Il metodo che introduciamo adesso, non è basato su una iterazione funzionale ma è molto utile, sia per la sua semplicità che per la proprietà di globale convergenza. Di solito viene combinato con uno dei metodi descritti precedentemente, permettendo la costruzione di metodi ibridi particolarmente potenti.

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

Uno dei più noti per trovare lo zero di una funzione che si basa solo sul questa proprietà è sicuramente il metodo delle bisezioni, detto anche suddivisione binaria, o metodo di Bolzano.

Supponiamo che α sia l'unico zero contenuto nell'intervallo $[a,b]$. Tale ipotesi non è restrittiva ai fini della convergenza del metodo delle bisezioni e non fa perdere di generalità alla seguente trattazione ma semplifica lo studio del problema.

Supponiamo che α sia l'unico zero in $[a,b]$. Posto $a_0 = a$ e $b_0 = b$, risulta $f(a_0)f(b_0) < 0$; si considera come prima stima di α 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: s

- se $f(c_0) = 0$ abbiamo trovato la soluzione;
- se $f(a_0)\,f(c_0) < 0$ allora lo zero cade in $[a_0, c_0]$ e si definisce un nuovo intervallo $[a_1, b_1] = [a_0, c_0]$;
- se $f(c_0)f(b_0) < 0$ allora lo zero cade in $[c_0, b_0]$ e si definisce un nuovo intervallo $[a_1, b_1] = [c_0, b_0]$;
- 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 del punto di transizione e quindi dello zero, ed il processo si ripete.

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

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

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

$$[a_n, b_n] = \begin{cases} [c_n, b_{n-1}] & \text{se } f(c_n)\, f(b_{n-1}) < 0 \\ \\ [a_{n-1}, c_n] & \text{se } f(a_{n-1})\, f(c_n) < 0. \end{cases} \tag{2.12}$$

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

Definiamo

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

allora

$$|e_0| \leq \frac{b_0 - a_0}{2},$$

$$|e_1| \leq \frac{b_0 - a_0}{2^2}$$

e, per ricorsività,

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

quindi

$$\lim_{n \to \infty} |e_n| = 0$$

e l'errore viene dimezzato ad ogni iterata.

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 stima buona dello zero. Ad esempio, è possibile arrestare l'implementazione quando l' errore relativo scende al di sotto di una certa tolleranza ε, cioè quando

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

In tal caso si ha la certezza che il valore dello zero sia ottenuto esattamente con la precisione richiesta. In realtà questa strategia non è fondata correttamente in quanto non tiene conto che la stima di un simile errore si basa sulla conoscenza del valore esatto dello zero cercato; una tale situazione non è verosimile in un caso reale, dato che non ci sarebbe motivo di utilizzare un metodo numerico se si conoscesse già il risultato esatto. Ciò che è invece necessaria è una stima che non richieda la conoscenza del valore dello zero.

A tale scopo è possibile calcolare un errore relativo approssimato, con la formula

$$E_a = \frac{|b_n - a_n|}{\min(|a_n, |b_n|)}. \tag{2.13}$$

Quando E_a diventa minore di un valore di riferimento per la terminazione, ε, il calcolo viene interrotto. Se si conosce lo zero α e si valuta l'errore relativo "vero" E_v, riportando i valori che E_v ed E_a assumono al variare delle iterate, si osserva una certa discontinuità dell'errore vero rispetto a quello approssimato. Questo è dovuto al fatto che nel metodo di bisezione lo zero dell'equazione può trovarsi in qualsiasi punto dell'intervallo di ricerca.

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

$$\frac{|b_n - a_n|}{2}$$

Una alternativa al criterio di arresto (2.13) la si ottiene considerando non più la successione dei punti medi ma quella degli estremi dell'intervallo definiti ad ogni passo ed imponendo che l'ampiezza di quest'ultimo sia sufficientemente piccola, cioè che risulti

$$b_n - a_n < \varepsilon \tag{2.14}$$

dove ε è una costante prefissata. Poichè

$$b_n - a_n = \frac{b - a}{2^n}$$

tale condizione è equivalente alla

$$\frac{b - a}{2^n} < \varepsilon. \tag{2.15}$$

ed è sicuramente verificata se

$$n > \log_2 \left(\frac{b-a}{\varepsilon} \right).$$

Essa garantisce che α sia approssimata da c_n con un errore assoluto minore in modulo di ε.

Conviene comuneque usare sempre un errore relativo, ma poichè 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:

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

Un altro possibile criterio di terminazione è

$$|f(c_n)| < \varepsilon. \tag{2.16}$$

Si osserva che la costante ε non deve essere scelta troppo piccola perchè a causa degli errori di arrotondamento le condizioni di arresto potrebbero non essere mai soddisfatte.

Il metodo delle bisezioni converge linearmente ad α ed ha costante asintotica pari ad $\frac{1}{2}$, risultando con ciò molto lento.

In particolare applicando il metodo al calcolo degli zeri di una data funzione, 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

$$3.3 \simeq \log_2 10.$$

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 globale convergenza 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 "*localmente convergenti*" che per poter essere implementati hanno però bisogno di una stima iniziale prossima allo zero. Il metodo delle bisezioni è inoltre molto lento; per questo se è necessario avere una certa velocità di convergenza, si preferiscono i metodi localmente convergenti, che però non sempre convergono. Tuttavia l'affidabilità dell'analisi dell'errore che esso consente lo rende talvolta migliore rispetto ad altri in certe applicazioni.

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:

```
{\tt
function y = cub(x)
y = x^3 - 2*x - 5;}

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

\begin{verbatim}
{\tt fzero('cub',-2)}
```

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 $k = 1$, $x0 = a_1$, $x_1 = b_1$ 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 è 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
```

verifichiamo se nell'intervallo $[0, 1]$ la funzione cambia di segno

```
>> 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. Inizializziamo i dati di input per la function bzero che implementa il metodo delle bisezioni

```
options = struct('TolX',1e-10,'Display','iter','Nmax',100)

options =

       TolX: 1.0000e-10
    Display: 'iter'
       Nmax: 100
```

e richiamiamo la function bzero

```
>> [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: 34
```

Il numero di iterate è piuttosto elevato, proviamo quindi ad usare un metodo più veloce come il metodo delle secanti o il metodo di Newton. La function szero che implementa il metodo delle secanti richiede gli stessi dati di input di bzero, i risultati sono i seguenti:

```
>> [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

x =

   NaN


fx =
```

```
   NaN


exitflag =

    -1


output =

     algorithm: 'secanti'
     funcCount: 5
    iterations: 3
```

Proviamo ora ad eseguire il metodo di Newton usando $x_0 = 0$ come punto iniziale e definendo la funzione che contiene la derivata di f.

```
>> 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 =

   NaN


fx =
```

```
   NaN


exitflag =

    -1


output =

     algorithm: 'newton'
     funcCount: 7
    iterations: 3
```

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 =

   4.6210e-02


fx =
```

```
   1.3323e-15


exitflag =

     1


output =

     algorithm: 'bisezioni, secanti'
     funcCount: 14
    iterations: 11
```

Un risultato analogo si ottiene utilizzando la function predefinita $fzero$

```
  [x,fx,exitflag,output]=fzero(f,[0 1],options)

 Func-count    x          f(x)             Procedure
    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   1.51421e-11        interpolation

Zero found in the interval [0, 1]

x =

   4.6210e-02
```

```
fx =

   1.5142e-11


exitflag =

     1


output =

    intervaliterations: 0
            iterations: 10
             funcCount: 12
             algorithm: 'bisection, interpolation'
               message: 'Zero found in the interval [0, 1]'
```

2.9 Errore, accuratezza e numero di condizione

Quando cerchiamo di valutare la funzione f nel punto x il valore non sarà esatto, ma otteniamo un valore perturbato

$$\tilde{f}(x) = f(x) + e(x)$$

L'errore $e(x)$ può essere generato da diverse fonti. Può derivare dagli errori di arrotondamento, o dall'approssimazione fatta per valutare la funzione, per esempio se la funzione è definita da un integrale che viene approssimato numericamente.

Dato α uno zero di f e supponiamo di conoscere un limite superiore sull'errore $|e(x)| < \epsilon$, se x_1 è un punto con $f(x_1) > \epsilon$ allora

$$\tilde{f}(x_1) = f(x_1) + e(x_1) \geq f(x_1) - \epsilon \geq 0$$

e quindi $\tilde{f}(x_1)$ ha lo stesso segno di $f(x_1)$. Lo stesso accade se $f(x_1) < -\epsilon$. Quindi se $|f(x)| > \epsilon$ il segno di $f(x)$ ci da informazioni sulla localizzazione dello zero.

Sia $[a, b]$ il più grande intervallo contenente α per cui

$$x \in [a, b] \text{ implica } |f(x)| \leq \epsilon$$

Se siamo fuori da questo intervallo il valore di $\tilde{f}(x)$ da informazioni sullo zero di f. Nell'intervallo il valore di $\tilde{f}(x)$ non ci da informazioni, neanche sul segno della funzione.

L'intervallo $[a, b]$ si chiama intervallo di incertezza dello zero. Gli algoritmi che calcolano una approssimazione dello zero all'interno di questo intervallo di incertezza si dicono stabili.

La grandezza dell'intervallo di incertezza varia da problema a problema. Se l'intervallo è piccolo il problema si dice ben condizionato, quindi un algoritmo stabile risolverà un problema ben condizionato accuratamente. Se l'intervallo è grande il problema si dice mal condizionato.

Per quantificare il grado di malcondizionamento calcoliamo il numero di condizione nel caso in cui $f'(\alpha) \neq 0$.

$$f(x) \approx f(\alpha) + f'(\alpha)(x - \alpha) = f'(\alpha)(x - \alpha)$$

segue che $|f(x)| \lesssim \epsilon$ quando $|f'(\alpha)(x - \alpha)| \lesssim \epsilon$, quindi

$$|x - \alpha| \lesssim \frac{\epsilon}{|f'(\alpha)|}$$

cioè

$$[a, b] \approx [\alpha - \frac{\epsilon}{|f'(\alpha)|}, \alpha + \frac{\epsilon}{|f'(\alpha)|}].$$

quindi

$$\frac{1}{|f'(\alpha)|}$$

è il numero di condizione del problema.