The mainspring of numerical analysis
is the entire aquifer of mathematics, pure and applied.
A. Iserles

CAPITOLO 1
Idee introduttive

In questo capitolo, a carattere *introduttivo*, si cerca inizialmente di chiarire, attraverso esempi, il significato dell'espressione *fare dell'analisi numerica*.

Successivamente, si evidenziano, in maniera schematica, i principi che fanno da supporto e da filo conduttore ai vari problemi e metodi dell'analisi numerica. In questo obiettivo si inseriscono pure i brevi richiami di analisi funzionale che, oltre fornire un linguaggio unificatore, è strumento efficace di sviluppo di nuove teorie.

Segue, infine, un tentativo di *classificazione* dei problemi numerici con lo scopo di individuare e caratterizzare i diversi livelli di difficoltà presenti nella loro risoluzione.

1.1 Analisi numerica

Naturalmente, la questione di *che cosa sia l'analisi numerica*[1] potrebbe essere trattata in maniera più adeguata alla *fine* del volume e, *soprattutto*, dopo aver *fatto dell'analisi numerica*.

Può essere, comunque, di qualche utilità, almeno come introduzione, fare alcune considerazioni.

1.1.1 Scopo dell'analisi numerica

Un modo per comprendere il *carattere* di una disciplina è di esaminarne gli obiettivi, che possono essere enunciati in forme diverse, ma equivalenti nella sostanza. Ad esempio, si può dire che obiettivo principale dell'analisi numerica è

trovare gli algoritmi che risolvono un problema matematico
nel minimo tempo
con la massima accuratezza.

In altra forma si ha la seguente caratterizzazione

analisi numerica $\Longrightarrow$ *arte di dare una risposta numerica ad un problema matematico mediante un calcolatore automatico digitale.*

Analizziamo, in particolare, gli *elementi* di quest'ultima definizione.

[1] o *calcolo numerico*; *calcolo* dal latino *calculus*: pietruzza (da *calce*, pietra calcare).

Risposta numerica

Esempio 1.1 Consideriamo il seguente problema: data l'equazione algebrica

$$P(x) = x^2 + 2bx + c = 0$$

con b, c reali e $b^2 > c$, dimostrare che essa ammette *due radici reali*.

Per la sua risoluzione possiamo seguire i seguenti due differenti metodi.

1. *metodo non costruttivo*, per riduzione all'assurdo. Supponiamo che *non* esistano soluzioni reali; allora, essendo $P(x)$ una funzione *continua*, si ha: $P(x) > 0$ o $P(x) < 0 \quad \forall x$; ma

$$P(-b) = b^2 - 2b^2 + c < 0, \quad P(|x|) > 0 \quad \text{per } x \text{ grande.}$$

2. *metodo costruttivo*. Come è ben noto, si ha

$$x^2 + 2bx + c = (x+b)^2 - b^2 + c = 0$$

da cui

$$x = -b \pm \sqrt{b^2 - c}, \quad \text{oppure} \quad x = \frac{c}{-b \mp \sqrt{b^2 - c}}$$

Nel secondo metodo, a differenza del primo, si fornisce anche una procedura per il calcolo effettivo (*numerico*) delle soluzioni, cioè un *algoritmo*. Come abbiamo messo in evidenza, la formula risolutiva può essere scritta in due modi diversi, che risultano *equivalenti* dal punto di vista teorico, ma non dal punto di vista *numerico*[2].

Naturalmente, l'algoritmo precedente non è l'unico disponibile; in realtà esso è specifico del particolare problema considerato. Più in generale, come vedremo nel seguito, per calcolare le radici di un polinomio o di una funzione non lineare esistono numerosi altri metodi di tipo iterativo: ad esempio, il metodo delle tangenti, delle secanti, ecc.

Necessità di una risposta numerica

L'esempio precedente è particolare in quanto la *risposta numerica* può essere data in forma *esplicita*. Questa non è chiaramente la situazione generale. Vediamo, a questo proposito, due esempi. Il primo illustra la impossibilità di avere sempre risposte *analitiche*; il secondo esamina in che senso una soluzione *analitica* può *non essere numerica*.

Esempio 1.2 *Dinamica di una popolazione cellulare.*

Esaminiamo il modello matematico dell'accrescimento di una popolazione, ad esempio di cellule, nell'ipotesi che il tasso di accrescimento sia una costante λ (*modello di Malthus*). Indicando con $y(t)$ una *misura* della quantità di cellule all'istante t si ha il seguente sistema dinamico

$$\begin{cases} y'(t) = \lambda y(t) \\ y(0) = y_0, \qquad y_0 > 0 \end{cases}$$

la cui soluzione *analitica* è data, come è noto, da: $y(t) = y_0 e^{\lambda t}$.

Consideriamo, ora, il problema di *calcolare* λ nota la quantità di popolazione in due istanti differenti, ad esempio in $t = 0$ e $t = \bar{t}$, con $\bar{t} > 0$. Il problema equivale, naturalmente, a risolvere la seguente equazione in λ

$$y_0 e^{\lambda \bar{t}} = y(\bar{t})$$

[2] Ad esempio, in corrispondenza a $b = -90.995$, $c = -1.82$, su un calcolatore in semplice precisione (4 bytes, cfr. il successivo Capitolo 2) si ottiene, con la prima formula $x_1 = 182$, $x_2 = -0.01000214$, mentre con la seconda: $x_1 = 181.9611$, $x_2 = -0.01$. La soluzione *vera* è data da: $x_1 = 182, x_2 = -0.01$.

la cui soluzione può essere espressa nella seguente forma *analitica*

$$\lambda = \frac{1}{\bar{t}} \log\left(\frac{y(\bar{t})}{y_0}\right)$$

Supponendo di poter calcolare la funzione *logaritmo*, la forma precedente è di tipo *esplicito*, in quanto non richiede ulteriori algoritmi.

Supponiamo, ora, che l'accrescimento avvenga in presenza di *emigrazione* o *immigrazione* con *velocità* d. L'equazione della dinamica del modello si trasforma, allora, nella seguente

$$y'(t) = \lambda y(t) + d$$

per la quale è ancora possibile calcolare la soluzione analitica $y(t)$. Tuttavia, il problema del calcolo di λ supposti noti: $d, y(0), \bar{t}, y(\bar{t})$ si trasforma ora nell'*equazione non lineare* seguente

$$e^{-\lambda \bar{t}} y(\bar{t}) = -\frac{d}{\lambda}\left(e^{-\lambda \bar{t}} - 1\right) + y_0$$

e diversamente dal caso precedente non è più possibile una esplicitazione in termini di funzioni *elementari*.

Naturalmente il problema si complica ulteriormente se anziché il modello di Malthus (valido solo in prima approssimazione) si adottano modelli strutturalmente più significativi, che tengano conto ad esempio dei fenomeni di *sovraffollamento*, con conseguenti termini non lineari nell'equazione della dinamica o con possibilità di effetti non istantanei (*equazioni con ritardo*).

Esempio 1.3 *Problema differenziale ai limiti.*

L'equazione di Bessel

$$x^2 \frac{d^2y}{dx^2} + x\frac{dy}{dx} + (x^2 - \nu^2)y = 0 \tag{1.1}$$

trova diverse applicazioni in problemi di matematica applicata, in particolare in problemi ai limiti relativi a cilindri circolari retti, o la propagazione di onde elettromagnetiche in conduttori circolari.

È noto che se ν non è un intero, due soluzioni indipendenti dell'equazione (1.1) sono $J_\nu(x)$ e $J_{-\nu}(x)$ dove

$$J_\nu(x) = \sum_{k=0}^{\infty} \frac{(-1)^k}{k!\,\Gamma(\nu + k + 1)} \left(\frac{x}{2}\right)^{\nu+2k}$$

Si tratta di una soluzione dal punto di vista analitico di tipo *esplicita*, ma certamente non ancora in forma conveniente per il calcolo *numerico*. Per renderla tale sono necessarie, in particolare, delle indicazioni sulla rapidità di *convergenza* della serie e delle *maggiorazioni dell'errore* che si commette approssimando la somma della serie con una particolare somma parziale.

Arte

Per risolvere un determinato problema matematico vi possono essere *diversi* algoritmi. Inoltre lo stesso algoritmo applicato a problemi *diversi* può comportarsi in maniera *diversa*. Nei capitoli successivi si esamineranno numerosi esempi di situazioni di questo tipo. Come introduzione suggeriamo, ad esempio, la lettura di Moler, Van Loan [300].

La scelta di un *particolare* algoritmo per la risoluzione di un *determinato* problema è il risultato, come vedremo, di un'*analisi del problema e di un'analisi comparativa* dei vari algoritmi, sulla base del loro *costo* e della loro *accuratezza*.

Una componente importante per una scelta ottimale è *l'intuizione affinata dall'esperienza*.

In questo senso, *trovare l'algoritmo che risolva nel minimo tempo e con la massima precisione un problema matematico* può essere, a ragione, considerato *un'espressione artistica*.

Uso del calcolatore

La disponibilità di calcolatori *programmabili, potenti e affidabili* è tra i maggiori responsabili dello sviluppo del calcolo numerico; essa è una *componente imprescindibile* nel progetto di un algoritmo.

Legata ad aspetti tecnici in continuo sviluppo (velocità, capacità di memoria, possibilità di uso interattivo, di lavoro in parallelo ...), può *suggerire* nuovi orientamenti, *rendere obsoleti* alcuni metodi, *rivalutarne* altri.

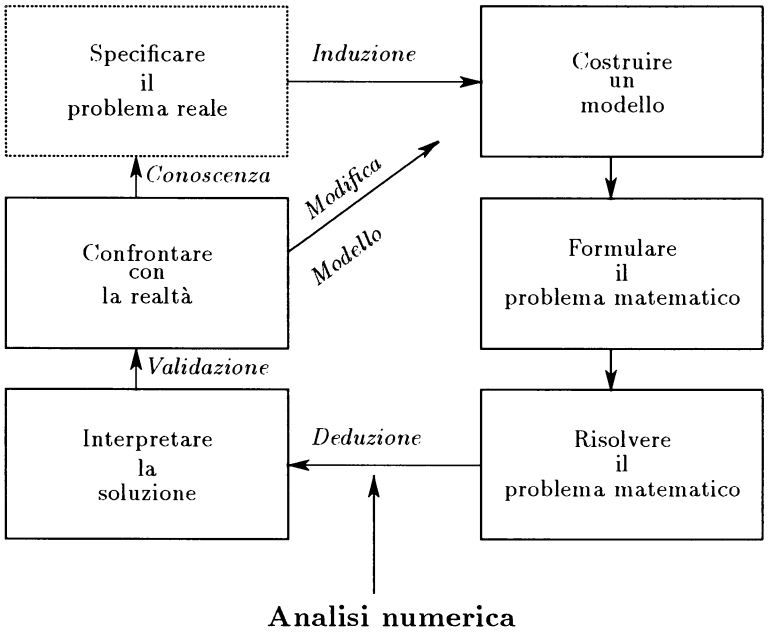

Figura 1.1 Collocazione dell'analisi numerica nel processo di matematizzazione di un problema reale.

1.1.2 Brevi note storiche

La storia dell'analisi numerica è in realtà quella della matematica, in quanto la matematica nasce per dare una *risposta numerica* a problemi *reali*.[3]

[3]Alcuni dati: circa 1650 A.C., in un papiro egiziano (*papiro Rhind* di Ahmes) un metodo per trovare le radici di una 'semplice' equazione (*regula falsi*) (cfr. [76], p. 88); *metodo di esaustione* introdotto da Archimede (287–212 A.C.) per il calcolo de lunghezze, aree e volumi di figure geometriche (cfr. [128], Ch. 2): quando utilizzato per trovare approssimazioni è decisamente nello spirito della moderna integrazione numerica ed è un precursore importante del calcolo di I. Newton e G. Leibniz.

Un impulso importante allo sviluppo dei procedimenti numerici è stata l'introduzione del calcolo da parte di Newton e Leibniz, dal momento che esso ha portato alla costruzione di modelli matematici per lo studio della realtà fisica, in particolare delle scienze fisiche, ma anche dell'ingegneria, della medicina e dell'economia. Tali modelli non possono usualmente essere risolti in maniera esplicita, e quindi sono necessari metodi numerici per ottenere delle soluzioni approssimate.

Newton in particolare ha introdotto per risolvere una varietà di problemi diversi metodi, alcuni dei quali portano attualmente il suo nome. Seguendo Newton, molti dei 'giganti' della matematica del 18-mo e 19-mo secolo hanno dato importanti contributi alla soluzione numerica dei problemi matematici: L. Eulero (1707–1783), J.L. Lagrange (1736–1813) e K.F. Gauss (1777-1855).

Durante i secoli questo aspetto è stato talvolta sovrastato da una concezione più filosofica della matematica, ma ogniqualvolta si è avuto un avvicinarsi della matematica alle *scienze applicate*, si è avuto insieme un risveglio di interesse alla *costruzione* e allo *studio* di algoritmi.

In effetti, il grande interesse attuale all'analisi numerica è dovuto da una parte, come già detto, alla disponibilità di idonei strumenti di calcolo, ma anche all'idea sempre più *convinta*, che la Matematica *è un utile strumento per lo studio del mondo reale*.

In questo senso l'analisi numerica è un aspetto della matematica applicata.

In Figura 1.1 è delineato in sintesi il cammino della ricerca, a partire dai dati *sperimentali*, verso la costruzione di *teorie* e lo sviluppo di *modelli*. L'analisi numerica si presenta in tale contesto come *l'anello decisivo* (il linguaggio) per la possibilità di una *validazione quantitativa* di un modello e quindi di una *verifica* delle teorie.

Per notizie riguardanti la *storia* di alcuni classici metodi numerici si veda, ad esempio, Cassina [74], Goldstine [168], [169], Metropolis et al. [284], Edwards [128], Nash [310], Chabert [76].

In bibliografia sono riportate numerose opere introduttive all'analisi numerica. Per brevità, ci limitiamo qui a segnalarne alcune, divenute ormai *classiche*; in particolare, Dahlquist, Björck [105], Forsythe et al. [144], Hamming [189], Henrici [196], Householder [209], Isaacson, Keller [216], Milne [293], Ostrowski [325], Ralston, Rabinowitz [351], Richtmyer, Morton [358], Stoer, Bulirsch [388], Varga [414], Wilkinson [429], [431], Wilkinson, Reinsch [430], Wendroff [422], Young, Gregory [441].

1.2 Principi di fondo

In maniera schematica si possono distinguere nell'analisi numerica i seguenti due aspetti

- **Metodologia**, che tratta, in particolare, la costruzione di algoritmi specifici, la loro efficienza, l'implementazione per un particolare calcolatore.
- **Analisi**, che studia i principi di fondo, le stime degli errori, la convergenza dei metodi.

Il primo è un aspetto più *pratico*, mentre il secondo è più *teorico*, di base. Nella risoluzione numerica di un problema i due aspetti, in generale ambedue presenti, si integrano a vicenda.

In questo paragrafo daremo una breve panoramica dei *principi di fondo* dell'analisi numerica. Si tratta, in realtà, di una anticipazione, in quanto essi verranno ripresi e approfonditi nel seguito, ma tale anticipazione può essere utile come filo conduttore di una materia che altrimenti potrebbe sembrare frammentaria.

1.2.1 Iterazione

In modo generale, l'idea dell'*iterazione* indica la *ripetizione di un processo "semplice" per migliorare la stima della soluzione di un problema più "complicato"*.

Diamo una illustrazione dell'idea mediante due particolari applicazioni.

Zero di una funzione

Il calcolo dello zero di una funzione $f(x)$, $\mathbb{R} \to \mathbb{R}$, può essere ricondotto, in vari modi, alla ricerca del *punto fisso* di una trasformazione $g(x)$; si ponga, ad esempio: $g(x) = x + kf(x)$, k reale $\neq 0$. Scelte meno *semplici* ma più *utili* di $g(x)$ saranno introdotte nel seguito.

Il problema *si trasforma*, allora, nel calcolo del numero reale α tale che

$$\alpha = g(\alpha) \tag{1.2}$$

Un'idea per il calcolo del punto fisso α consiste nel cercare di *migliorare* una *stima* iniziale x_0, mediante un procedimento *iterativo* del tipo seguente ed illustrato in Figura 1.2

$$x_{\nu+1} = g(x_\nu), \quad \nu = 0, 1, 2, \cdots \tag{1.3}$$

Naturalmente, affinché il procedimento sia *utile* occorre che siano verificate alcune condizioni. In particolare, la successione x_ν deve rimanere nel dominio di definizione della funzione $g(x)$ e *convergere* al punto fisso α. Per essere, poi, *conveniente*, la convergenza non deve essere *troppo lenta*. In Figura 1.2 sono rappresentati i quattro possibili comportamenti. In maniera intuitiva si vede che, se la funzione $g(x)$ è *derivabile*, si ha convergenza quando è verificata la condizione $|g'(x)| < 1$ (più in generale, come vedremo nel seguito, quando la funzione $g(x)$ è una *contrazione*). La *velocità di convergenza* dipende da quanto è *piccola* la funzione $|g'(x)|$ nel punto fisso α (più in generale, la costante di contrazione).

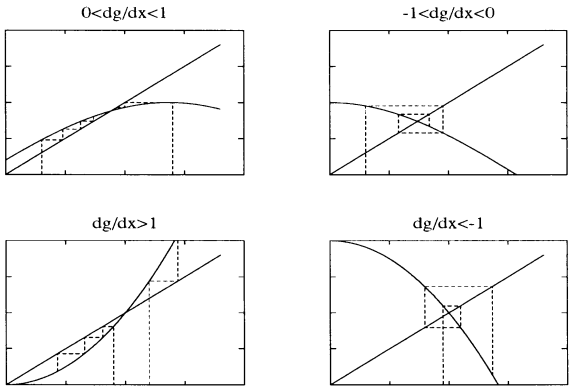

Figura 1.2 Metodo iterativo per il calcolo del punto fisso.

L'*analisi* del metodo, per ora soltanto intuitiva, è servita a mettere in luce alcuni *criteri utili* per la scelta della funzione $g(x)$. Vale la pena sottolineare che, se il problema di partenza è la ricerca di uno zero della funzione $f(x)$, la funzione $g(x)$ deve essere *scelta*. Un altro aspetto di cui tenere conto, in tale scelta, è il *costo* (in termini di *operazioni elementari*) del calcolo di $g(x)$. Il *costo complessivo* risulta, infatti, dal *costo di ogni iterazione per il numero delle iterazioni* e quindi è il risultato sia della velocità di convergenza che del costo per ogni valutazione di $g(x)$.

Osservazione 1.1 *Il metodo (1.3) può essere generalizzato in diverse direzioni. Da un lato si può considerare il punto unito come intersezione di due curve $\psi(x)$, $g(x)$, con $\psi(x)$ non necessariamente data dalla retta $y = x$, e considerare, quindi, il procedimento iterativo*

$$\psi(x_{\nu+1}) = g(x_\nu), \quad \nu = 0, 1, 2, \ldots$$

Naturalmente la funzione ψ dovrà, per quanto riguarda il problema del calcolo degli zeri, essere più "semplice" della funzione $f(x)$.

Un'altra generalizzazione consiste nel tenere memoria *di più termini della successione* $\{x_\nu\}$*, considerando procedimenti a* più passi *definiti nel modo seguente*

$$x_{\nu+1} = G(x_\nu, x_{\nu-1}, x_{\nu-2}, \cdots, x_{\nu-r+1})$$

Per tali metodi si pone il problema di dare r stime iniziali; ma in generale, a parità di velocità di convergenza essi risultano più "economici", in termini di numero di operazioni, dei metodi a un passo. Essi realizzano, infatti, un migliore sfruttamento della memoria.

Vediamo, ora, una classica applicazione del procedimento iterativo (1.3).

Esempio 1.4 *Calcolo della radice quadrata.*

L'equazione $x^2 = c$, con $c > 0$, può essere scritta nella forma $x = g(x)$ ad esempio nei due modi seguenti

$$x = \frac{1}{2}\left(x + \frac{c}{x}\right) \tag{1.4}$$

$$x = \frac{c}{x} \tag{1.5}$$

Nel caso dell'equazione (1.4) si ha: $g'(\alpha) = 0$, mentre per l'equazione (1.5) si ha: $g'(\alpha) = -1$, ove $\alpha = \sqrt{c}$.

Nel primo caso si ha convergenza, mentre nel secondo la successione oscilla alternativamente fra x_0 e c/x_0.

È interessante vedere la rapidità di convergenza nel primo caso. Assumendo ad es. $c = 2$ e $x_0 = 1.5$ mediante il seguente programma in *doppia precisione (8 bytes)*

```
c=2.; xt=sqrt(c);
x=1.5;
while (abs(x-xt)>=1.e-14)
x=(x+c/x)/2;
[x,xt]
end
```

si ottengono i risultati riportati in Tabella 1.1.

ν	x_ν	$\sqrt{2}$
1	1.416666666666667	1.414213562373095
2	1.414215686274510	1.414213562373095
3	1.414213562374690	1.414213562373095
4	1.414213562373095	1.414213562373095

Tabella 1.1 Risultati dell'iterazione di Newton.

Da essa si ricava l'informazione che la convergenza è di tipo *quadratico*, cioè se x_ν ha t cifre esatte, allora $x_{\nu+1}$ ha almeno $2t - 1$ cifre esatte. Si può facilmente verificare che il metodo iterativo basato sulla trasformazione (1.4) non è altro che il metodo delle tangenti, o metodo di Newton (vedi Figura 1.3) applicato all'equazione: $f(x) = x^2 - c = 0$. Come vedremo nel seguito, la convergenza quadratica è una caratteristica del metodo di Newton quando le radici sono *distinte*.

Problema differenziale a valori iniziali

Un problema differenziale a valori iniziali del tipo

$$\begin{cases} y'(x) = f(x, y) \\ y(x_0) = y_0 \end{cases} \tag{1.6}$$

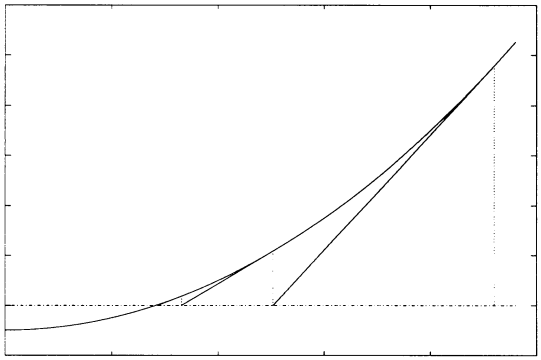

Figura 1.3 Illustrazione del metodo di Newton–Raphson.

può essere trasformato nella seguente *equazione integrale* equivalente

$$y(x) = y_0 + \int_{x_0}^{x} f(t, y(t))dt$$

per la quale è possibile applicare un procedimento *iterativo* del tipo visto in precedenza. In effetti, posto, ad esempio: $y_0(x) = y_0$ si costruisce la successione di funzioni $\{y_\nu(x)\}$ calcolando

$$y_{\nu+1}(x) = y_0 + \int_{x_0}^{x} f(t, y_\nu(t))dt \tag{1.7}$$

Sotto opportune ipotesi di regolarità sulla funzione $f(x, y)$ (lipschitziana in y) , si ha che la successione converge alla soluzione del problema a valori iniziali e, quindi, il procedimento può essere di tipo *costruttivo*. Osserviamo, comunque, che per essere veramente tale è ancora necessario un metodo numerico per il calcolo dei successivi integrali.

1.2.2 Approssimazione locale

L'idea dell'approssimazione locale consiste nel sostituire ad una funzione complicata, cioè non calcolabile direttamente, una funzione più *semplice*. Il significato preciso del termine *semplice* dipende dal contesto, cioè dall'uso che di tale funzione viene fatto. Le procedure più comuni per ottenere questo tipo di approssimazione sono basate su uno sviluppo in serie troncato oppure su una operazione di interpolazione.

Sviluppo in serie

Consideriamo, ad esempio, il problema del calcolo della soluzione del seguente sistema non lineare

$$F(x) = 0 \tag{1.8}$$

ove $F : \mathbb{R}^n \to \mathbb{R}^n$.

A partire da una stima della soluzione, che indichiamo con x_0, cerchiamo un vettore $\delta \in \mathbb{R}^n$ tale che

$$F(x_0 + \delta) = 0 \tag{1.9}$$

A tale scopo, supponendo la funzione $F(x)$ sufficientemente regolare, sviluppiamo la funzione F in serie intorno al punto x_0. Arrestandoci al termine di primo grado, si ottiene il seguente *problema approssimato*

$$F(x_0) + F'(x_0)\bar{\delta} = 0$$

che, essendo di tipo *lineare*, è in generale più facilmente risolubile del problema originario.

Naturalmente, avendo trascurato nello sviluppo in serie i termini di grado superiore al primo, il vettore $\bar{\delta}$ è, in generale, solo una *approssimazione* della soluzione del sistema (1.9). Per migliorare la precisione si può utilizzare la precedente idea dell'iterazione, *sostituendo* a x_0 il valore $x_0 + \bar{\delta}$.

Interpolazione

L'idea dell'interpolazione consiste nel sostituire ad una funzione $f(x)$ definita su un intervallo (a, b) un *polinomio* $P(x)$, che assume gli stessi valori della funzione $f(x)$ in punti prefissati dell'intervallo di definizione, detti *punti di collocazione*.

Un'idea più generale consiste, data una suddivisione dell'intervallo, nel sostituire alla funzione $f(x)$ una funzione polinomiale a tratti, cioè una funzione definita su ogni tratto della suddivisione come un polinomio di un grado prefissato.

Come vedremo più dettagliatamente nel seguito, in particolari situazioni, come ad esempio in problemi relativi allo smoothing di dati sperimentali o all'approssimazione di soluzioni di equazioni differenziali, l'interpolazione mediante polinomi a tratti presenta dei *vantaggi* di migliore adattabilità rispetto alla interpolazione con lo stesso polinomio su tutto l'intervallo.

La procedura di interpolazione è di base per costruire numerosi algoritmi. Vediamo alcuni esempi.

Esempio 1.5 *Approssimazione di un integrale.* Per approssimare l'integrale:

$$I = \int_a^b f(x)dx$$

di una funzione $f(x)$, di cui non è nota una primitiva, si può *sostituire* alla $f(x)$ una funzione, *più semplice* rispetto all'operazione di integrazione. Utilizzando l'idea dell'interpolazione, si sostituisce alla funzione $f(x)$ il polinomio interpolatore relativo a un insieme di punti (*i nodi*) dell'intervallo (a, b) oppure un polinomio a tratti relativo alla suddivisione introdotta dai nodi.

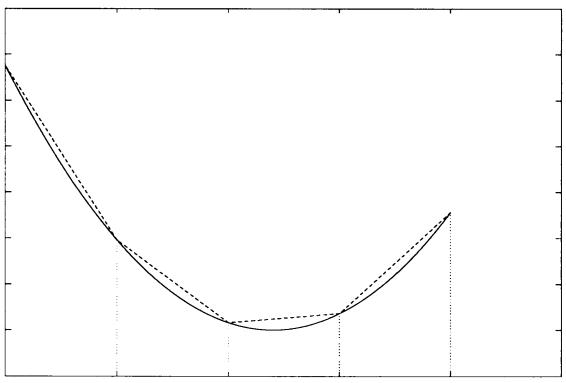

Figura 1.4 Formula dei trapezi.

Introducendo, ad esempio, una suddivisione dell'intervallo (a, b) in parti uguali, con passo $h = (b - a)/n$, n, intero ≥ 1, e utilizzando i polinomi a tratti di primo grado, si ottiene come approssimazione la nota *formula dei trapezi*, illustrata in Figura 1.4 e definita nel modo seguente

$$I \simeq T(h) := \frac{1}{2}h\sum_{i=0}^{n-1}(f_i + f_{i+1}) \tag{1.10}$$

Supponendo la funzione $f(x)$ sufficientemente regolare, ad esempio $f \in C^2$, vedremo successivamente che

$$T(h) - I = O(h^2) \tag{1.11}$$

Cioè, l'*errore* è un infinitesimo del *secondo ordine* per $h \to 0$.

Per avere *maggiore accuratezza* con meno lavoro di quello richiesto dalla formula dei trapezi, si possono seguire le seguenti due idee

1. *approssimare localmente* la funzione $f(x)$ mediante polinomi di grado più elevato o scegliere opportunamente i nodi.
2. calcolare la formula dei trapezi per differenti valori di h e allora *estrapolare* per $h = 0$.

Esaminiamo, in particolare, la seconda idea.

Supponiamo di aver calcolato la formula (1.10) per $2h$ e h. Allora, ricordando la (1.11), abbiamo i seguenti risultati

$$T(2h) - I \simeq k(2h)^2$$
$$T(h) - I \simeq k(h)^2$$

ove k è indipendente da h. Considerando le equazioni precedenti come un sistema nelle due *incognite* I, k, si ha in particolare

$$I \simeq T(h) + \frac{1}{3}(T(h) - T(2h))$$

In questo modo, trattando il simbolo di $\simeq$ come se fosse un simbolo di uguaglianza, si è ottenuto per l'integrale un valore "più accurato". Effettivamente, come vedremo successivamente, ciò è vero per h sufficientemente piccolo, cioè *asintoticamente*.

Il procedimento ora seguito, di cui lasciamo come esercizio l'interpretazione grafica, è noto come *metodo di estrapolazione*.

Il suo *interesse numerico* sta nel fatto che per ottenere il nuovo valore *non si è più calcolata la formula* (1.10), che, contenendo il calcolo della funzione $f(x)$, rappresenta, in generale, la parte più costosa del procedimento.

Naturalmente l'idea non si limita a questo particolare caso, ma trova applicazione, come vedremo, in numerose altre situazioni.

Esempio 1.6 *Problemi differenziali ai limiti.* Consideriamo il seguente problema ai limiti

$$-y''(x) + \alpha(x)y = f(x), \quad \alpha(x) \geq 0 \quad x \in (a, b) \tag{1.12}$$
$$y(a) = y(b) = 0 \tag{1.13}$$

Vedremo successivamente che una formulazione del problema "equivalente" alla precedente è la seguente

$$\min_z \left[\frac{1}{2} \left(\int_a^b z'^2 \, dx + \int_a^b \alpha(x) z^2 \, dx \right) - \int_a^b z f(x) \, dx \right] \tag{1.14}$$

ove il *minimo* è preso su tutte le funzioni $z(x)$ che sono di quadrato sommabile su (a, b) insieme alle derivata prima e nulle in a e b.

Un'idea per approssimare la soluzione di (1.12), (1.13) consiste nell'introdurre una *reticolazione* dell'intervallo (a, b) e nell'approssimare *localmente* la derivata seconda mediante dei *rapporti incrementali*.

Per il problema nella forma (1.14), si può, invece, approssimare lo spazio di funzioni in cui si cerca il minimo mediante uno spazio di funzioni a *dimensione finita* (ad esempio lo spazio delle funzioni polinomiali a tratti).

Naturalmente, si pone anche qui il problema dell'*accuratezza* della soluzione approssimata, che nel caso della formulazione in termini di derivate, è legata alla precisione con cui si approssima la derivata (*errore di discretizzazione locale*) e nel caso della formulazione (1.14) alla *distanza* tra lo spazio delle funzioni continue e quello delle funzioni discrete (*errore di interpolazione*).

1.3 Brevi richiami di analisi funzionale

L'*analisi funzionale* fornisce insieme uno strumento efficiente per lo sviluppo di nuove teorie e la possibilità di estrarre le idee essenziali con un linguaggio preciso ed elegante.

Per questo motivo, in questo paragrafo verrà fatta una rapida *rassegna* dei principali concetti, rinviando per una più adeguato approfondimento ad esempio a Brezis [60], Milne [294], Yosida [439], Zeidler [442].

1.3.1 Spazi lineari

Definizione 1.1 *Sia* $X = \{x, y, z, \ldots\}$ *un insieme e* $K = \{\alpha, \beta, \gamma, \ldots\}$ *un campo di scalari. Sia definita un'operazione di* somma *tra due qualunque elementi di* X *ed un'operazione di* moltiplicazione scalare *tra ciascun elemento di* K *e ciascun elemento di* X *in modo tale che*

- **(i)** $x \in X$, $y \in X \Rightarrow x + y \in X$
- **(ii)** $x \in X, \alpha \in K \Rightarrow \alpha x \in X$
- **(iii)** $x + y = y + x$
- **(iv)** $(x + y) + z = x + (y + z)$
- **(v)** *Esiste un elemento* $0 \in X$ *tale che* $x + 0 = x$ *per ogni* $x \in X$
- **(vi)** *Per* $x \in X$ *esiste un unico elemento, chiamato l'opposto di* x *e denotato con* $-x$*, tale che* $x + (-x) = 0$
- **(vii)** $\alpha(\beta x) = (\alpha\beta)x$
- **(viii)** $\alpha(x + y) = \alpha x + \alpha y$
- **(ix)** $(\alpha + \beta)x = \alpha x + \beta x$
- **(x)** $1x = x$

Allora X *è detto* spazio lineare *sul campo* K*.*

Nel caso in cui K è il campo dei numeri reali X è detto uno *spazio lineare reale*. Nel seguito, salvo avviso contrario, considereremo, in particolare, spazi lineari reali.

Esempio 1.7 Vi sono numerosi esempi di spazi lineari di interesse nella matematica applicata e nell'analisi numerica

(a) Lo spazio $\mathbb{R}^n$ dei vettori ad n componenti, con l'usuale definizione di addizione e moltiplicazione per uno scalare.

(b) Lo spazio delle funzioni m-volte continuamente differenziabili su un intervallo della retta reale $[a, b]$, denotato con $C^{(m)}([a, b])$. L'addizione e la moltiplicazione per uno scalare è intesa nel senso usuale, cioè per *punti*.

(c) Lo spazio $L^p(\Omega)$ delle funzioni a potenza p-ma sommabile su un insieme misurabile $\Omega \subset \mathbb{R}^n$, $n \geq 1$.

(d) Lo spazio l_p delle successioni di numeri reali $\{x_i\}$ tali che:

$$\sum_{i=1}^{\infty} |x_i|^p < \infty$$

(e) Lo spazio dei polinomi $\mathbb{P}_n$ di grado $\leq n$.

Definizione 1.2 *Siano $x_1, x_2, \ldots x_n$ elementi fissati di un spazio lineare X sul campo K. La somma*

$$\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n$$

con $\alpha_i \in K$ è chiamata una combinazione lineare *degli elementi x_i. Gli elementi x_i sono detti* linearmente indipendenti *se $\forall \alpha_i \in K$ si ha la seguente implicazione*

$$\alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n = 0 \Rightarrow \alpha_1 = \alpha_2 = \cdots = \alpha_n = 0$$

altrimenti si dicono linearmente dipendenti.

Definizione 1.3 *Se esistono n elementi $x_1, x_2, \ldots, x_n$ di uno spazio lineare X linearmente indipendenti, e ogni insieme di $n+1$ elementi di X è linearmente dipendente, allora n è la* dimensione *di X, denotata con* $\dim(X)$. *Se per ogni $n > 0$ esistono n elementi linearmente indipendenti in X, allora X è detto di dimensione* infinita.

Definizione 1.4 *Un insieme di n elementi $x_1, x_2, \ldots, x_n$ linearmente indipendenti di uno spazio X è chiamata una* base *per X se ogni $x \in X$ può essere espresso come una combinazione lineare degli elementi x_i, $i = 1, 2, \ldots, n$.*

Esempio 1.8 $\{1, x, x^2, \ldots, x^n\}$ è una base per lo spazio $\mathbb{P}_n$ dei polinomi di grado $\leq n$.

Esempio 1.9 Lo spazio delle funzioni continue $C^0([a, b])$ ha dimensione infinita.

Esempio 1.10 $\mathbb{R}^n$ ha come base l'insieme di vettori $\{e_1, e_2, \ldots, e_n\}$, ove:

$$e_i = \begin{pmatrix} 0, & 0, & \cdots, & 0, & 1, & 0, & \cdots, & 0 \\ 1, & 2, & \cdots, & i-1, & i, & i+1, & \cdots, & n \end{pmatrix}$$

Si dimostra facilmente il seguente risultato:

Teorema 1.1 *Uno spazio lineare X è di dimensione finita n se e solo se ha una base di n elementi.*

Inoltre, ogni insieme di elementi $x_1, x_2, \ldots, x_n$ di X genera uno spazio di dimensione finita, chiamato lo spazio generato da tali elementi e indicato con $\text{span}(x_1, x_2, \ldots, x_n)$.

1.3.2 Norme

Analizzando i metodi di approssimazione è necessario spesso *confrontare* o *misurare* la differenza fra le varie risposte. Si tratta, pertanto, di introdurre una distanza tra due punti di uno spazio lineare.

Definizione 1.5 *Uno spazio lineare X è detto* normato *se esiste una applicazione: $X \to \mathbb{R}_+$, chiamata* norma *e indicata usualmente con $\|x\|$, tale che*

- **(i)** $\|x\| \geq 0$, *per ogni* $x \in X$.
- **(ii)** $\|x\| = 0$ *se e solo se* $x = 0$.
- **(iii)** $\|\alpha x\| = |\alpha| \, \|x\|$, *per ogni* $x \in X$, $\alpha \in K$.
- **(iv)** *Per ogni* $x, y \in X$, *vale la* disuguaglianza triangolare

$$\|x + y\| \leq \|x\| + \|y\|$$

Il numero $\|x - y\|$ definisce una *distanza* tra i punti x e y.

Osservazione 1.2 *Una* seminorma *è un'applicazione: $X \to \mathbb{R}_+$, indicata talvolta con $|x|$, che soddisfa le condizioni $(i), (iii), (iv)$. Si dice che $|x|$ è non* triviale *se e solo se $|x| > 0$ per almeno un elemento $x \in X$.*

Osservazione 1.3 *Nel linguaggio della matematica astratta l'introduzione di una norma comporta l'introduzione nello spazio lineare di una* topologia *che permette di* generalizzare *i concetti geometrici di* intorno, convergenza*, ecc. Per definire la distanza tra due punti si potrebbe utilizzare la nozione più generale di* metrica *e considerare gli* spazi metrici*, anziché gli spazi normati. Tuttavia, la maggior parte dei problemi in analisi numerica può essere discussa adeguatamente nel contesto degli spazi lineari normati.*

Successivamente si esamineranno in particolare le norme di vettore e di matrice.

1.3.3 Spazi a dimensione infinita

Estendiamo ora l'idea di *base* ad uno spazio a *dimensione infinita*.

Definizione 1.6 *Un insieme di elementi $x_1, x_2, \ldots$ di uno spazio normato X è detto* chiuso *(o completo) in X se per ogni $x \in X$ e $\epsilon > 0$ esiste un n e un insieme di scalari $\alpha_1, \alpha_2, \ldots, \alpha_n$ tali che*

$$\|x - \sum_{i=1}^{n} \alpha_i x_i\| \leq \epsilon$$

Se l'insieme $\{x_i\}$ è chiuso in X e linearmente indipendente (cioè, tutti i sottoinsiemi finiti sono linearmente indipendenti), allora si dice che $\{x_i\}$ è una *base* per X.

Uno spazio è detto *separabile* se possiede una base *numerabile* (o finita). Sono gli spazi più interessanti dal punto di vista numerico e nel seguito considereremo, in particolare, spazi che verificano tale proprietà.

Spazi di Banach e di Hilbert

Definizione 1.7 *Una successione* $\{x_n\}$ *in uno spazio lineare normato è detta* successione di Cauchy *se*

$$\lim_{n\to\infty} \lim_{p\to\infty} \|x_{n+p} - x_n\| = 0 \tag{1.15}$$

Definizione 1.8 *Uno spazio lineare normato* X *è detto* completo *se ogni successione di Cauchy in* X *converge a un elemento in* X*. Uno spazio normato completo è detto spazio di* Banach*.*

♦ **Esercizio 1.1** *Dimostrare che ogni spazio lineare normato a dimensione finita è completo.*

♦ **Esercizio 1.2** *Dimostrare che lo spazio* $C([a,b])$ *non è completo rispetto alla norma*

$$\|x\| = \left(\int_a^b x^2(t)\, dt \right)^{1/2}$$

Fornire un controesempio.

Definizione 1.9 *Sia* X *uno spazio lineare reale. Su di esso è definito un* prodotto scalare *se ad ogni* $x, y \in X$ *è associato un numero reale, denotato usualmente con* (x, y)*, tale che*

1. $(x, x) \geq 0$ *e se* $(x, x) = 0 \Rightarrow x = 0$.
2. $(x, y) = (y, x)$.
3. $(\alpha x, y) = \alpha(x, y)$.
4. $(x + y, z) = (x, z) + (y, z)$.

Se si pone

$$\|x\| = (x, x)^{1/2}$$

si ottiene uno spazio normato. Se X *risulta completo allora viene detto spazio di* Hilbert*.*

Ricordiamo, lasciando la dimostrazione come esercizio, la seguente utile disuguaglianza.

Proposizione 1.1 (Cauchy-Schwarz) *Dato uno spazio lineare* X *dotato di un prodotto scalare* $(\cdot, \cdot)$*, si ha per ogni* $x, y \in X$*:*

$$|(x, y)| \leq \sqrt{(x, x)(y, y)} = \|x\|\, \|y\|$$

L'angolo θ tra due elementi x, y è definito da

$$\theta = \arccos \frac{(x, y)}{\|x\|\, \|y\|}$$

Due elementi x, y sono *ortogonali* se $\theta = \pi/2$, cioè se $(x, y) = 0$. Un elemento x è detto ortogonale a un sottospazio Φ di X se

$$(x, \phi) = 0 \quad \forall \phi \in \Phi.$$

L'elemento $z = (x, y)x/(x, x)$ è la proiezione ortogonale di y su x; il vettore $y - z$ è ortogonale a x.

Si lascia come esercizio la dimostrazione della seguente proprietà.

Proposizione 1.2 *Per ogni x, y in uno spazio dotato di prodotto scalare si ha la seguente uguaglianza, detta legge del parallelogramma*

$$\|x+y\|^2 + \|x-y\|^2 = 2\|x\|^2 + 2\|y\|^2 \tag{1.16}$$

Se

$$\|x+y\| = \|x\| + \|y\|$$

allora x e y sono linearmente dipendenti e se inoltre $\|x\| = \|y\|$, allora $x = y$.

1.3.4 Trasformazioni e operatori

Sia S_1 un sottoinsieme di uno spazio lineare X e S_2 un sottoinsieme di uno spazio lineare Y. Una *trasformazione* da S_1 in S_2 è definita assegnando una "regola" che associa ad ogni elemento di S_1 un unico elemento di S_2. Chiamiamo *operatore* la rappresentazione di una trasformazione. L'equazione

$$Tx = y$$

indica che l'operatore T trasforma l'elemento $x \in S_1$ in un elemento $y \in S_2$. L'insieme di tutti gli elementi per i quali T è definito è il *dominio* di T, denotato con $\mathrm{D}(T)$. L'insieme degli elementi $y = Tx$ per $x \in \mathrm{D}(T)$ è l'*immagine* o *range* di T, denotato con $\mathcal{R}(T)$.

La trasformazione è *suriettiva* su S_2 se $\mathcal{R}(T) = S_2$; *iniettiva* se $Tx_1 = Tx_2 \Rightarrow x_1 = x_2$.

Per una trasformazione iniettiva è definita una trasformazione da $\mathcal{R}(T)$ in $\mathrm{D}(T)$ tale che $\forall y \in \mathcal{R}(T)$ esiste un unico x con $Tx = y$; l'operatore che rappresenta tale trasformazione è chiamato l'*inverso* di T e denotato con T^{-1}.

Le operazioni di *addizione* di operatori e di *moltiplicazione per uno scalare* si definiscono in modo ovvio; se T_1 e T_2 hanno un dominio comune X e α è uno scalare allora per ogni $x \in X$ si ha

$$\begin{aligned} &(T_1 + T_2)x = T_1x + T_2x, \qquad \forall x \in X \\ &(\alpha T)x = \alpha(Tx) \end{aligned}$$

Se T_1 e T_2 sono due operatori tali che $\mathcal{R}(T_2)$ è contenuto in $\mathrm{D}(T_1)$, allora possiamo definire il prodotto T_1T_2 ponendo

$$(T_1T_2)x = T_1(T_2x)$$

Naturalmente, non vale in generale la proprietà *commutativa*, cioè, in generale, $T_1T_2 \neq T_2T_1$.

Se un operatore T è invertibile, allora

$$T^{-1}T = I, \qquad TT^{-1} = I, \qquad (I = \text{ identità})$$

Per gli operatori che non hanno inversa si può introdurre un un concetto meno restrittivo.

Definizione 1.10 *Se esiste un operatore T_L^{-1} tale che*

$$T_L^{-1}T = I$$

allora esso è chiamato inverso a sinistra *di T. Analogamente T_R^{-1} è detto* inverso a destra *se:*

$$TT_R^{-1} = I$$

Si lascia come esercizio la dimostrazione delle seguenti proprietà.

1. Se esiste sia l'inverso a sinistra che l'inverso a destra, allora essi coincidono con l'inverso.
2. Se esiste T_R^{-1}, allora T è suriettiva su Y e l'equazione $Tx = y$ ha almeno una soluzione $x = T_R^{-1}y$.
3. Se esiste T_L^{-1}, allora T è iniettiva. Se l'equazione $Tx = y$ ha una soluzione, allora tale soluzione è unica, data da $x = T_L^{-1}y$.

♦ **Esercizio 1.3** *Mostrare che l'operatore $d/dx : C^1([a,b]) \to C([a,b])$ è suriettivo ma non iniettivo. Quali sono gli operatori inversi a destra di tale operatore?*

Definizione 1.11 *Un operatore L è detto* lineare *se il suo dominio è uno spazio lineare e se*

$$L(\alpha x + \beta y) = \alpha Lx + \beta Ly$$

per tutti gli $x, y \in \mathrm{D}(L)$ e tutti gli scalari α, β.

Definizione 1.12 *Un operatore $T : X \to Y$, con X, Y spazi lineari normati, è detto* limitato *se e solo se esiste una costante $c < \infty$ tale che*

$$\|Tx_1 - Tx_2\| \le c\|x_1 - x_2\|, \qquad \forall x_1, x_2 \in X. \tag{1.17}$$

Il valore $\mu(T) = \inf c$ è la limitazione *dell'operatore T su X.*

Osservazione 1.4 *Per semplicità di notazioni qui e nel seguito, quando non si presentano situazioni di equivoco, si utilizza la medesima notazione per indicare sia la norma in X che in Y, anche se le due possono, in generale, essere diverse.*

Per gli operatori lineari si può dimostrare il seguente risultato.

Proposizione 1.3 *Sia L un operatore lineare e limitato da X in Y. Allora*

$$\mu(L) = \sup_{x \in X, x \neq 0} \frac{\|Lx\|}{\|x\|} = \sup_{\|x\|=1} \frac{\|Lx\|}{\|x\|} \tag{1.18}$$

L'insieme di tutti gli operatori lineari e limitati da uno spazio lineare X in uno spazio lineare Y è esso stesso uno spazio lineare, con le definizioni di operazione di addizione e moltiplicazione per uno scalare introdotte in precedenza. Esso viene indicato usualmente $\mathcal{L}(X, Y)$.

Per un operatore lineare, anziché di limitazione dell'operatore si parla usualmente di *norma*, definita quindi da

$$\|L\| = \sup_{\|x\|=1} \|Lx\|$$

♦ **Esercizio 1.4** *Si consideri l'operatore integrale $T : C([0,1]) \to C([0,1])$ definito da*

$$(Tx)(t) = \int_0^1 K(t,s)x(s)\,ds$$

ove $K(t,s)$ è una funzione continua di t e s. Dimostrare che la norma di operatore indotta dalla norma del massimo è la seguente

$$\|T\|_\infty = \max_{0 \le t \le 1} \int_0^1 |K(t,s)|\,ds.$$

e che la norma indotta dalla norma 1, cioè $\|x\|_1 = \int_0^1 |x(t)|\,dt$ è la seguente

$$\|T\|_1 = \max_{0 \le s \le 1} \int_0^1 |K(t,s)|\,dt$$

Definizione 1.13 *Un operatore T è* continuo *in x se per ogni successione $\{x_n\}$ convergente a x si ha*

$$\lim_{n\to\infty} \|Tx_n - Tx\| = 0$$

Lasciamo come esercizio il seguente risultato.

Proposizione 1.4 *Un operatore lineare L è continuo nel dominio di definizione se e solo se è continuo in 0 e se e solo se esso è limitato.*

Per operatori non lineari, invece, si ha che la limitatezza implica la continuità ma non il viceversa (dare controesempi). Per una funzione di variabile reale la condizione (1.17) significa la continuità di Lipschitz, che è una condizione più forte della continuità ordinaria.

♦ **Esercizio 1.5** *Se L è un operatore lineare, mostrare che $\mathcal{R}(A)$ è uno spazio lineare. Mostrare con esempi che questo non è necessariamente vero per un operatore nonlineare.*

Ricordiamo infine il seguente importante risultato.

Teorema 1.2 (Banach–Steinhaus) *Siano X e Y due spazi di Banach e $\{T_i\}$, $i \in I$, una famiglia (non necessariamente numerabile) di operatori lineari e continui da X in Y. Se*

$$\sup_{i\in I} \|T_i x\| < \infty \qquad \forall x \in X$$

allora

$$\sup_{i\in I} \|T_i\|_{\mathcal{L}(X,Y)} < \infty$$

In altre parole esiste una costante c tale che

$$\|T_i x\| \le c\,\|x\| \qquad \forall x \in X,\ \ \forall i \in I$$

Il risultato precedente è anche noto come *principio della limitatezza uniforme* (principle of uniform boundedness): da una *stima puntuale* si ottiene una *stima uniforme*.

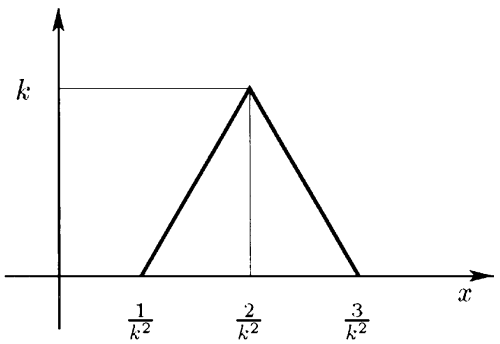

Figura 1.5 Successione $f_k(x)$.

1.3.5 Importanza della scelta della norma

Terminiamo questa panoramica di nozioni di analisi funzionale richiamando l'attenzione sull'importanza di un'opportuna scelta della *norma* in uno spazio lineare. A tale scopo, consideriamo il seguente esempio.

Sia $k \geq 1$ un numero assegnato, destinato a tendere a $+\infty$. Definiamo per ogni k la seguente funzione sull'intervallo $[0, 3]$

$$f_k(x) = \begin{cases} k(k^2x - 1), & \text{per } \dfrac{1}{k^2} \leq x \leq \dfrac{2}{k^2} \\ -k(k^2x - 3), & \text{per } \dfrac{2}{k^2} \leq x \leq \dfrac{3}{k^2} \\ 0, & \text{altrove.} \end{cases}$$

Indicando con $\|\cdot\|_p$ la norma nello spazio $L^p(a, b)$, si possono dimostrare facilmente i seguenti risultati, che mostrano il diverso comportamento della successione, per $k \to \infty$, a secondo della norma utilizzata

$$\|f_k(x)\|_1 := \int_0^3 |f_k(x)|\, dx = \frac{1}{k} \to 0$$

$$\|f_k(x)\|_2 := \left(\int_0^3 |f_k(x)|^2\, dx \right)^{1/2} = \frac{\sqrt{2}}{\sqrt{3}}$$

$$\|f_k(x)\|_\infty := \max_{x \in [0,3]} |f_k(x)| = k \to \infty$$

1.4 Classificazione dei problemi computazionali

Come ogni classificazione, anche quella che ora presenteremo, non ha lo scopo, del resto impossibile, di catalogare in forma precisa i vari tipi di problemi numerici, quanto piuttosto di fornire un quadro macroscopico di riferimento per quanto riguarda le differenti difficoltà e quindi le differenti tecniche.

In forma schematica e facendo riferimento alle notazioni del paragrafo precedente, possiamo rappresentare un *problema numerico* nella forma seguente

$$Tx = y \tag{1.19}$$

ove $x \in X$, $y \in Y$; X, Y sono spazi lineari e T è un operatore : $X \to Y$. Si possono distinguere i seguenti tre tipi di problemi, in ordine crescente di difficoltà.

1. *Il problema diretto.* Dato x e T, determinare y. Problemi di questo tipo sono, ad esempio: il calcolo del valore di una funzione assegnata, per un valore fissato della variabile indipendente; il calcolo di un integrale definito. Nel secondo esempio l'*input* è la funzione integranda e l'insieme di integrazione mentre T è l'operatore di integrazione.
2. *Il problema inverso.* Dato T e y, determinare x. Esempi sono: la risoluzione di un sistema di equazioni; la risoluzione di un problema differenziale a valori iniziali o ai limiti. Nel caso, ad esempio, della risoluzione di un sistema lineare $Ax = b$, si conosce l'*output* b, l'operatore A e si vuole conoscere l'*input* x.

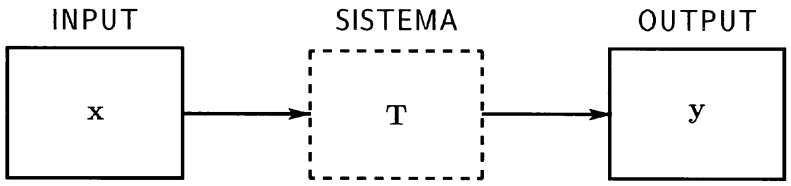

Figura 1.6 Rappresentazione schematica di un problema numerico.

3. *Problema di identificazione.* Dato una famiglia di x e y, trovare T. È, in sostanza, il problema della *costruzione* di un *modello matematico*.

È opportuno sottolineare che *problema diretto* non significa necessariamente assenza di *difficoltà numeriche*. Nel seguito studieremo ad esempio l'*approssimazione* di una funzione mediante funzioni *più semplici* (ad esempio, *polinomi*), cioè calcolabili con un numero finito di operazioni. Tale approssimazione è la *base* per la costruzione di *algoritmi* per il calcolo di funzioni *non elementari* e per affrontare il problema dell'integrazione e della derivazione. Le questioni *numeriche* che si pongono per un problema diretto riguardano lo studio degli *errori di troncamento* e della *convergenza*, nonché della *stabilità* degli algoritmi proposti.

Il *problema inverso* rappresenta, anche per le sue importanti applicazioni, il *problema centrale* nell'analisi numerica. In alcuni casi particolari esso può essere ricondotto a un problema diretto. Si pensi ad esempio alla formula risolutiva di un sistema lineare mediante il calcolo dell'inversa, oppure alle formule per l'integrazione di particolari equazioni differenziali ordinarie (ad esempio lineari), alle funzioni di Green per certe equazioni differenziali ellittiche. La conoscenza dell'*operatore inverso* può essere interessante dal punto di vista teorico e, talvolta, anche dal punto di vista pratico, per evidenziare il *comportamento qualitativo* della soluzione. Tuttavia, dal punto di vista numerico, cioè per una valutazione *quantitativa*, essa non è necessariamente di aiuto. Nel caso, ad esempio, di un sistema lineare il calcolo dell'inversa della matrice non è sempre lo strumento numerico più adatto per il calcolo della soluzione.

Il *problema di identificazione*, il più ambizioso dei tre tipi di problemi, in generale presenta le più grosse difficoltà. Analizziamo come esempio una situazione particolare che si inquadra nella teoria dell'*approssimazione*. Supponiamo che una funzione sia data *solo* per punti, cioè in forma *tabellare* e che si voglia *conoscere* i valori della funzione in punti diversi da quelli dati. Il problema è, quindi, quello di *approssimare* una funzione sulla base della conoscenza di alcuni suoi valori, che possono essere, ad esempio, *dati sperimentali*. Chiaramente, il problema può non avere soluzione unica, a meno che non si restringa opportunamente la classe delle *funzioni ammissibili* e si precisi il senso di *vicino*, mediante la scelta di una particolare norma. Quando gli spazi X e Y sono a dimensione infinita, si può assumere come *approssimazione* di T *una combinazione finita* di elementi di una base, ad esempio una combinazione di polinomi. Il problema è in questo modo ricondotto alla ricerca di un numero finito di parametri (*gradi di libertà*), che usualmente vengono calcolati *minimizzando* una opportuna misura dello scarto tra i dati del problema e quelli forniti dalla approssimazione di T. Si ha, quindi, in definitiva un *problema di programmazione matematica*, che sarà trattato nel seguito.

Un approccio, concettualmente simile, ma con evidenti difficoltà aggiuntive, può essere adottato nella situazione più generale quando, anziché una funzione, l'operatore incognito T è un

problema differenziale, integrale, alle differenze, ecc. In questo caso l'operatore T, viene *stimato* con operatori "semplici", ad esempio *lineari*. Si costruisce cioè un *modello matematico*. Solitamente nel modello sono presenti dei parametri, in generale con un significato *fisico, chimico*, ecc. La loro funzione è quella di *adattare* il modello approssimato ai dati sperimentali. Il loro calcolo comporta, quindi, come abbiamo visto in precedenza, la risoluzione di problemi di ottimizzazione. Una volta calcolati i parametri si tratta di vedere se il modello corrispondente è adeguato a descrivere il fenomeno corrispondente ai dati sperimentali. È questo un punto estremamente *delicato* del processo di modellizzazione, che comporta, in particolare, una scelta *significativa* dei dati sperimentali.

In definitiva, per quanto riguarda l'aspetto numerico, i problemi di identificazione *richiedono* la risoluzione (eventualmente iterata) di problemi di tipo diretto o inverso e, in particolare, di problemi di ottimizzazione.

1.5 Esempi introduttivi di Matlab

Condizione *indispensabile* per imparare ad usare Matlab (**link** 1) è poter disporre su calcolatore di una sua versione (meglio se versione 6, o versioni successive).

Mediante le istruzioni

```
>> help
>> help help
>> demo
```

si possono apprendere rapidamente le *nozioni* e le *istruzioni di base*, che saranno quindi supposte note per il seguito. Come un qualunque altro linguaggio, vale sempre il principio che il modo migliore per apprenderlo è usarlo!

In questo libro l'ambiente Matlab è utilizzato principalmente come strumento 'utile' per *implementare e analizzare i metodi e gli algoritmi numerici*. Allo stesso scopo possono, naturalmente, essere utilizzati altri ben noti linguaggi di programmazione (Fortran, C,...).

Per comodità, in questo capitolo a carattere introduttivo vengono evidenziate, mediante alcune semplici applicazioni, differenti caratteristiche e possibilità di Matlab (per una introduzione più organica si veda ad esempio [309]).

Ad esempio, il seguente segmento di programma

```
     % simbolo utilizzato per commenti
u=rand(1,2000000); v=rand(2000000,1);% generazione di due vettori random
t=cputime;  %inizializza il tempo di cpu
tic         % altro modo di valutare il tempo di esecuzione
p=u*v;      % prodotto scalare
toc         % chiude tic e mostra il tempo impiegato in secondi
cputime-t   % tempo di esecuzione in secondi
-----------
t=cputime; tic
p=0;
for i=1:2000000 %
p=p+u(i)*v(i);  % prodotto scalare eseguito componente per componente
end             %
toc; cputime-t
```

evidenzia il fatto che Matlab è un linguaggio *vettoriale*: può operare direttamente su matrici e in maniera decisamente più conveniente che componente per componente. Nel seguito vedremo altri esempi illustrativi di tale stile di programmazione.

1.5.1 Grafica mediante Matlab

Un altro aspetto interessante di Matlab è la possibilità di ottenere facilmente dei grafici.

Si consideri ad esempio il seguente `script`, ossia una successione di istruzioni in Matlab (scritte mediante, ad esempio, MATLAB Editor/Debugger, l'editore implementato nel sistema) e salvato come un file matlab, ossia `nome.m`.

```
% Script File: SinePlot
% grafico della funzione  sin(2*pi*x) con accuratezza crescente
close all % chiude tutte le figure esistenti
for n = [4 8 12 16 20 50 100 200 400]
   x = linspace(0,1,n); % costruisce una suddivisione in n parti uguali di [0,1];
                        % x vettore riga
   y = sin(2*pi*x);     % y vettore dei valori della funzione in x
   plot(x,y)            % grafico
   title(sprintf('Plot di sin(2*pi*x) con n = %3.0f punti ',n))
   pause     % arresto momentaneo dell'esecuzione; premere un tasto per ripartire
end
```

Il seguente script per la stessa funzione produce una tabella di valori.

```
% Script File: SineTable
clc  % Clear command window: annulla i precedenti comandi
     % sulla finestra dei comandi di Matlab
n = 21;
x = linspace(0,1,n);
y = sin(2*pi*x);
disp(' ')
disp('  k     x(k)    sin(x(k))')
disp('-------------------------')
for k=1:21
   deg = (k-1)*360/(n-1);
   disp(sprintf(' %2.0f     %3.0f     %6.3f    ',k,deg,y(k)));
end
disp( ' ');
disp('x(k) è dato in gradi.')
disp(sprintf('Un grado = %5.3e radianti',pi/180))
```

Ulteriori esemplificazioni sono fornite dai seguenti script.

```
% Script File: Poligoni
% grafici di alcuni poligoni regolari
close all
theta = linspace(0,2*pi,361);
c = cos(theta); s = sin(theta);
k=0;
for lati = [3 4 5 6 8 10 12 18 24]
   passo = 360/lati;
   k=k+1;
```

```
   subplot(3,3,k)
   plot(c(1:passo:361),s(1:passo:361))
   axis([-1.2 1.2 -1.2 1.2])
   axis equal
end
```

La figura (cfr. Figura 1.7) ottenuta può essere salvata in un file di diverso formato (.bmp, .jpg, .ps, .eps,...) e utilizzata per la stampa.

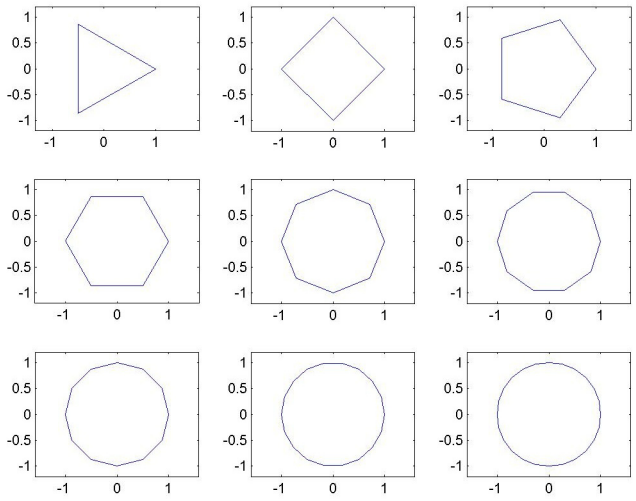

Figura 1.7 Alcuni poligoni regolari.

```
% grafico(efficiente)di un cerchio
close all
x = linspace(0,1,200);
y = sqrt(1 - x.^2);
plot(x,y)
axis square equal
hold on
plot(x,-y); plot(-x,y); plot(-x,-y)
hold off
```

♦ Esercizio 1.6 *Se*

```
x=linspace(0,2*pi,100),
y=sin(x); z=exp(-x);
```

scrivere un segmento di programma in matlab per tracciare il grafico della funzione $e^{-x}\sin(x)$ *sull'intervallo* $[0, 4\pi]$ *senza ulteriori calcoli delle due funzioni.*

Colore parametrico

```
co=['y' 'g' 'm' 'r'];
x=0:0.1:1;
A=[];
figure
 for m=1:3
     y=sin(2*m*pi*x);
     A=[A;y];
     col=co(m);
     h=plot(x,y,col);
     hold on
 end
figure
k=plot(x,A)
legend(k,'1','2','3',-1)
axis square

% spirale di Archimede
a=1;
theta=0:pi/60:2*pi; rho=a*theta;
polar(theta,rho);
grid
title('spirale di Archimede, \rho= a \theta')


function[xt,yt]=rot2d(t,x,y)
% rotazione (con centro l'origine) di un oggetto bidimensionale rappresentato
% da due vettori riga  x e y. L'angolo di rotazione t è dato in gradi. I vettori
% trasformati  sono dati in xt e yt.
t1=t*pi/180;
r=[cos(t1) -sin(t1); sin(t1) cos(t1)];
x=[x x(1)]; y=[y y(1)];
hold on; grid on; axis equal
fill(x,y,'b')
z=r*[x;y];
xt=z(1,:); yt=z(2,:);
fill(xt,yt,'r')
title(sprintf('rotazione piana di %3.2f gradi',t))
hold off
----
% ad esempio:
x=[1 2 3 2]; y=[3 1 2 4];
[xt,yt]=rot2d(75,x,y)


%rappresentazione grafica di una funzione
f='(2*x-sqrt(2))*sin(2*x)';
fplot(f,[-1,2])
xlabel('x');ylabel('f(x)')
title('Rappresentazione grafica di una funzione')
hold on
g='exp(x)*cos(x)';
fplot(g,[-1 2], 'r')
legend('f(x)=(2x-sqrt(2)) sin(2x)', 'g(x)=e^x cos(x)')
```

Il seguente programma visualizza in maniera geometrica il metodo di Newton per l'approssimazione di $\sqrt{2}$.

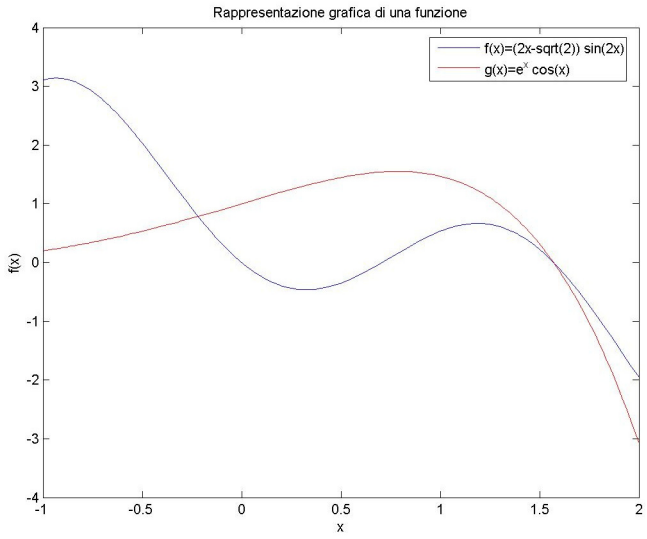

Figura 1.8 Esempio di rappresentazione grafica.

```
% radice2.m
% Visualizza 4 iterazioni della costruzione  geometrica della radice di due
A=2;
x(1)=1;
for k=1:4
   x(k+1)=0.5*(x(k)+A/x(k)); %metodo di Newton
   subplot(2,2,k);           % successivi sottografici
   xp=[0 x(k) x(k) 0 0];
   yp=[0 0 A/x(k) A/x(k) 0];
   fill(xp,yp,'y');      % riempimento del rettangolo con colore y(ellow)
   xlabel(sprintf('x(%d)=%4.3f',k,x(k)));
   axis([0 A 0 A]);  % definizione degli assi
   axis('square'); % help axis
end
```

I risultati ottenuti sono rappresentati in Figura 1.9.
Il seguente script esamina graficamente la bontà dell'approssimazione di una funzione mediante una funzione razionale fratta.

```
-----------
% Script File: ExpPlot
% Esamina la funzione
%      f(x) = ((1 + x/24)/(1 - x/12 + x^2/384))^8
% come una approssimazione di exp(x) sull'intervallo [0,1].
close all
x  = linspace(0,1,200);
num   = 1 + x/24; denom = 1 - x/12 + (x/384).*x; quot = num./denom;
y = quot.^8;
plot(x,y,x,exp(x))
err=norm(y-exp(x),inf) %distanza nella norma del massimo
```

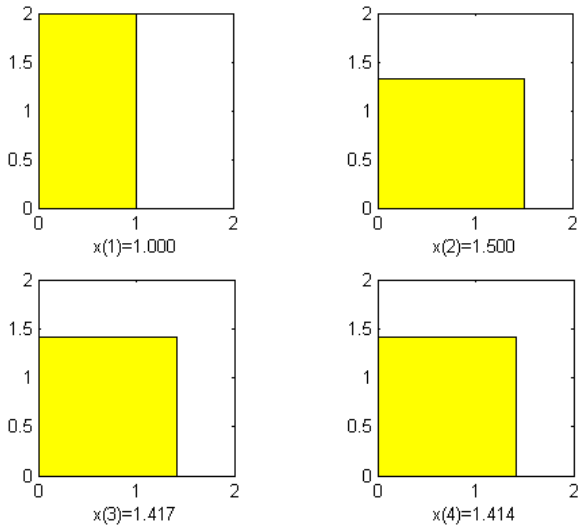

Figura 1.9 Costruzione geometrica della radice di 2.

Grafica in 3D

```
% illustra le varie possibilità di rappresentazione grafica in 3D
[x,y]=meshgrid(-2:0.1:2, -2:0.1:2); % x y matrici
clf
f='x.*exp(-x.^2-y.^2)'; % definisce la funzione z=f(x,y)
z=eval(f); % calcola la funzione in x, y)
surf(x,y,z)
pause
colorbar
pause
mesh(x,y,z)
pause
meshc(x,y,z)
pause
surfc(x,y,z)
pause
pcolor(x,y,z)
pause
surf(x,y,z,gradient(z))
pause
contour(x,y,z)
pause
plot3(x,y,z)


% superficie parametrica
[r,theta]=meshgrid(0:0.1:2,0:0.1:6*pi)
x=r.*cos(theta);
y=r.*sin(theta);
z=theta;
surf(x,y,z)
```

```
function cones(alpha)
% 3-D di un cono con semiangolo alpha
% provare alpha=0.5236
[X,Y]=meshgrid(-10:0.5:10, -10:0.5:10);
Z=-sqrt(X.^2+Y.^2)/tan(alpha);
i=find(Z<Z(1,21));
for n=i'
    Z(n)=Z(1,21);
end
subplot(1,2,1)
mesh(X,Y,Z), axis('equal')
t=['\alpha= ' num2str(alpha) ' radians'];
title(t);
subplot(1,2,2)
contour3(Z),axis('equal')
```

Il risultato del successivo script è rappresentato in Figura 1.10.

```
N = 10; % numero di incrementi. Provare ad aumentare.
z  = linspace(-5,5,N)';
radius = sqrt(1+z.^2);    % provare con altre funzioni
theta = 2*pi*linspace(0,1,N);
X = radius*cos(theta);
Y = radius*sin(theta);
Z = z(:,ones(1,N));
surf(X,Y,Z)
axis equal
```

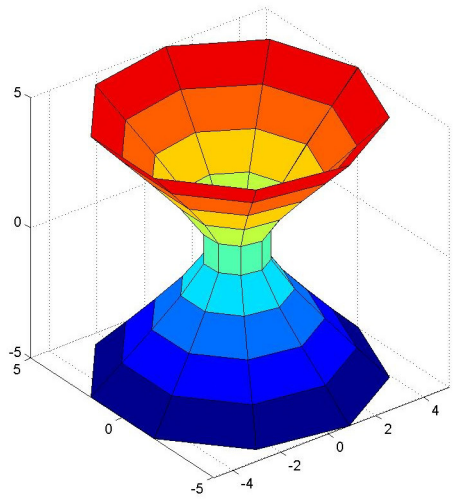

Figura 1.10 Esempio di rappresentazione grafica.

mesh non strutturate (non equidistanti)

```
x=[-2 -1 0 1 2 -2 -1 0 1 2 -2 -1 0 1 2]';
y=[ 0  0 0 0 0  1  1 1 1 1  2  2 2 2 2]';
tri=delaunay(x,y); % tri è una matrice 3 per M (ordine di x e y)
                   % che contiene le informazioni per generare i triangoli
                   % ogni riga contiene gli indici dei tre vertici
 z=x+y;
trimesh(tri,x,y,z)
```

Dimostrazione della possibilità di eseguire animazioni

Si vuole visualizzare le oscillazioni di una molla di forma elicoidale. Le equazioni parametriche della curva sono

$$x = \frac{d}{2}\cos p; \qquad y = \frac{d}{2}\sin p; \qquad z = p$$

dove d è il diametro dell'elica. Per simulare le oscillazioni di ampiezza A e frequenza angolare ω si modifica l'equazione di z nel modo seguente

$$z = (1 + A\cos(\omega(t-1)))p$$

ove t è la variabile temporale.

```
p=0:pi/60:8*pi;
d=2;A=0.2;T=5;
omega=2*pi/T;
t=0;
%plot del frame di riferimento
x=d*cos(p)/2; y=d*sin(p)/2;
z=(1+A*cos(omega*(t-1)))*p;
plot3(x,y,z)
pause
M=moviein(6);
for t=1:6  %memorizza i movie
    x=d*cos(p)/2; y=d*sin(p)/2;
    z=(1+A*cos(omega*(t-1)))*p;
    plot3(x,y,z)
    axis([-1 1 -1 1 0 10.0*pi]);
    M(:,t)=getframe;
end
pause
movie(M,10) %esegue l'animazione per dieci volte

%superficie
[x,y]=meshgrid(-pi:0.1:pi);
f='sin(x).*y/5*cos(t)';
nframes=20;
tt=linspace(0,2*pi,nframes);
figure(1);clf
Mv=moviein(nframes);
for n=1:nframes
    t=tt(n);z=eval(f);surf(x,y,z);
    axis([-pi pi -pi pi -1 1]);
    Mv(:,n)=getframe;
end
movie(Mv,4);
```

1.5.2 Il problema $3x+1$

Dato un intero positivo x_1 si definisce una successione $\{x_k, k = 1, 2, \ldots\}$ nel seguente modo

$$x_{k+1} = \begin{cases} x_k/2 & \text{se } x_k \text{ è pari} \\ 3x_k + 1 & \text{se } x_k \text{ è dispari} \end{cases}$$

Ad esempio, se $x_1 = 7$, si ha la successione

$$7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1, 4, 2, 1, \ldots$$

La successione diventa ciclica quando raggiunge il valore 1. Si hanno diverse questioni interessanti

- la successione raggiunge sempre la situazione ciclica?
- se n è il più piccolo indice tale che $x_n = 1$, come si comporta n come funzione del valore iniziale x_1?
- vi sono dei 'pattern' interessanti nella successione?

Ne seguito si costruisce uno script che può aiutare nell'indagine precedente[4].

```
% Script File: UpDown
% genera un vettore colonna  x(1:n) di interi positivi
% dove x(1) è richiesto come input e
%          x(k+1) = x(k)/2    if x(k) is even.
%          x(k+1) = 3x(k)+1   if x(k) is odd.
% Il valore di n è o 500 (soglia fissata ad arbitrio)
% oppure il primo indice per il quale si ha x(n) = 1
x = zeros(500,1);
x(1) = input('valore iniziale (intero positivo): ');
k = 1;
while ((x(k) ~= 1) & (k < 500))
   if rem(x(k),2) == 0
      x(k+1) = x(k)/2;
   else
      x(k+1) = 3*x(k)+1;
   end
   k = k+1;
end
n = k;
x = x(1:n);
clc
disp(sprintf('x(1:%1.0f) = \n',n))
disp(sprintf('%-8.0f',x))
[xmax,imax] = max(x);
disp(sprintf('\n x(%1.0f) = %1.0f è il massimo.',imax,xmax))
density = sum(x<=x(1))/x(1); %sum(x<=x(1)) è il numero delle componenti
                             %in x che sono minori o uguali a x(1)
disp(sprintf(' La densità è %5.3f.',density))
```

[4] Il problema posto da L. Collatz nel 1937, in connessione allo studio di grafi orientati, è anche noto come *algoritmo di Hasse*, *problema di Kakutani*, *congettura di Thwaites*, *problema di Syracuse*, *problema di Ulam*. Si hanno 'serie' indicazioni che non esista nessun altro ciclo oltre quello semplice 4-2-1. Il problema della convergenza è comunque tuttora aperto. cfr. **link** 10, **link** 11.

```
close all
figure
plot(x)
title(sprintf('x(1) = %1.0f, n = %1.0f',x(1),n));
figure
plot(-sort(-x))
title('successione in ordine crescente.')
I = find(rem(x(1:n-1),2)==1);
if length(I)>1
   figure
   plot((1:n),zeros(1,n),I+1,x(I+1),I+1,x(I+1),'*')
   title('massimi locali')
end
```

```
% Script File: RunUpDown
% memorizza i risultati nel file UpDownOutput.
while(input('altro esempio? (1=si, 0=no)'))
   diary UpDownOutput
   UpDown
   diary off
   disp(' ')
   if (input('memorizzare? (1=si, 0=no)')~=1)
      delete UpDownOutput
   end
end
```

♦ **Esercizio 1.7** *Sia $\{x_i\}$ la successione up/down con $x_1 = m$ e $g(m)$ l'indice del primo x_i che è uguale a uno. Tracciare il grafico di g per $m = 1 : 200$.*

1.5.3 Numeri primi

```
function p=primi(N)
% Ricerca dei primi <= N
% p=primi(N)
% con N>=11
% (Semplice implementazione crivello di Eratostene)
r=13;
p=[2 3 5 7 11];
i=5;  %i conta l'attuale numero di primi
while r<=N
        s=2;
        while p(s)*p(s)<=r & rem(r,p(s))>0
                s=s+1;
        end
        if  rem(r,p(s))>0
                i=i+1;
                p(i)=r;
        end
        r=r+2;
end
```

1.5.4 Metodo Monte Carlo

```
----------------
% Script File: Dice
% simula 1000 lanci di una coppia di dadi.
close all
First  = 1 + floor(6*rand(1000,1));
Second = 1 + floor(6*rand(1000,1));
Throws = First + Second;
hist(Throws, linspace(2,12,11));
title('Outcome of 1000 Dice Rolls.')
```

Calcolo approssimato di π

Si considera un cerchio di centro l'origine e di raggio 1. Il quadrato è il bersaglio di successivi colpi con uguale probabilità. Il rapporto tra l'area del quadrato circoscritto (= 4) e l'area del cerchi (= π) è allora approssimato dalla frazione di colpi sul cerchi e il totale dei colpi

$$\frac{\pi}{4} \approx \frac{\text{numero di colpi entro il cerchio}}{\text{numero totale dei colpi}}$$

```
% Script File: Darts
% stima di pi usando bersagli random
close all
rand('seed',.123456)
NIn = 0;
PiEstimate = zeros(500,1);
for k=1:500
   x = -1+2*rand(100,1);
   y = -1+2*rand(100,1);
   NIn = NIn + sum(x.^2 + y.^2 <= 1);
   PiEs(k) = (NIn/(k*100))*4;
end
plot(PiEs)
title(sprintf('stima mediante Monte Carlo di Pi = %5.3f',PiEs(500)));
xlabel('centinaia di tentativi')
```

Metodo di Archimede Il metodo di Archimede per il calcolo del numero π è basato sul calcolo del perimetro di una successione di poligoni inscritti in una circonferenza di diametro d assegnato. Dato un poligono di n lati inscritto in una circonferenza di diametro d, si ha

$$\theta = \frac{2\pi}{n}, \quad l = d \sin\left(\frac{\theta}{2}\right)$$

Il perimetro del poligono è dato da $p_n = n\,l$ e quindi

$$p_n = n\,d \sin\left(\frac{\pi}{n}\right)$$

Tenendo conto che $\lim_{x\to 0} \sin x/x = 0$, si ha

$$\lim_{n\to \text{inf}} p_n = d\,\pi$$

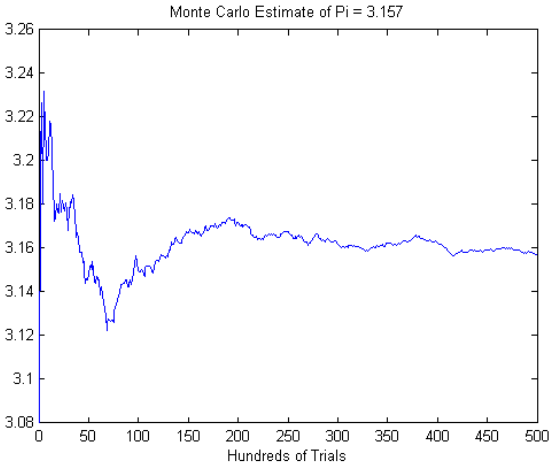

Figura 1.11 bersagli.

```
for k=2:6
n=2^k;
errore1=abs(n*sin(pi/n)-pi);
fprintf('%2d \t %5.4e\n',n,errore1);
end
 4   3.1317e-001
 8   8.0125e-002
16   2.0148e-002
32   5.0442e-003
64   1.2615e-003
```

Dai risultati si intuisce una convergenza *quadratica*, ossia al raddoppio di n corrisponde una riduzione di 4 dell'errore ($E_n = \mathrm{O}(1/n^2)$).

♦ **Esercizio 1.8** *Implementare una simulazione per stimare il volume di* $\{(x_1, x_2, x_3, x_4) : x_1^2 + x_2^2 + x_3^2 + x_4^2 \leq 1\}$*, la sfera unitaria nello spazio a 4 dimensioni.*

1.5.5 Ricorsione

Illustriamo con alcuni esempi la possibilità di utilizzare in matlab la procedura di ricorsione.

Permutazioni

```
function y=rndprm1(x)
% permutazione random utilizzando un  for loop
% x=[x1 x2 ...] o x='c1c2...', xi numeri, ci caratteri
[m,n]=size(x);
if m> 1
    error('rndprm accetta solo vettori riga')
end
y=[];
l=n;
for i=1:n
```

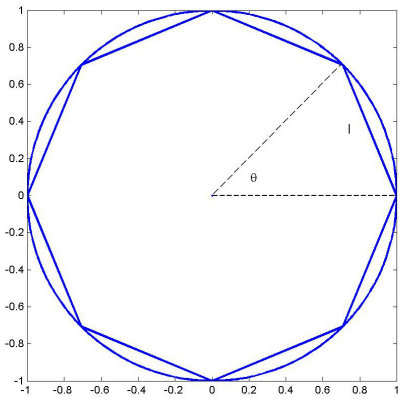

Figura 1.12 Approssimazione di π tramite un poligono inscritto nella circonferenza (metodo di Archimede).

```
     i=1+fix(l*rand);
     xs=x(i);
     y=[y,xs]; %aggiunge xs al vettore y
     x(i)=[]; %toglie x(i) dal vettore
     l=l-1;
 end
---------
function y=rndprm1(x)
% permutazione random utilizzando un while loop
[m,n]=size(x);
if m> 1
    error('rndprm accetta solo vettori riga')
end
y=[];
l=n;
while l>0
     i=1+fix(l*rand);
     xs=x(i);
     y=[y,xs];
     x(i)=[];
     l=l-1;
 end
---------
function y=rndprm(x)
% permutazione random utilizzando una ricorsione
%x=[x1 x2 ...] o x='c1c2...', xi numeri, ci caratteri
[m,n]=size(x);
if m> 1
    error('rndprm accetta solo vettori riga')
end
if n<=1
    y=x;
else i=1+fix(n*rand);
    xs=x(i);
    x(i)=[];
```

```
    z=rndprm(x);
    y=[z,xs];
end
```

Numeri di Fibonacci

$$f_1 = 1, \quad f_2 = 2 \qquad f_n = f_{n-1} + f_{n-2}$$

```
function f=fibonacci(n)
% genera i primi n elementi della sequenza di Fibonacci
f=zeros(n,1);
f(1)=1; f(2)=2;
for k=3:n
   f(k)=f(k-1)+f(k-2);
end
---------
% provare
% n=40; f=fibonacci(n);
% f(2:n)./f(1:n-1)
---------
function f=fibnum(n)
% calcola l'elemento n-mo della sequenza di Fibonacci
if n<=1
   n=1;
else
   f=fibnum(n-1)+fibnum(n-2);
end
----------
% provare tic, fibnum(24), toc

----------
% formula esplicita
phi=(1+sqrt(5))/2;
format long e
n=(1:20)';
f=(phi.^(n+1)-(1-phi).^(n+1))/(2*phi-1)
```

Frattali

```
%Iterated Function System
z=0+0*i;
s=(1+i)/2;
for j=1:30000
   p=floor(2*rand)+1;
   if p==1
      z=s*z+1;
   else
      z=s*z-1;
   end
   x(j)=real(z);
   y(j)=imag(z);
end
```

```
plot(x,y,'.')
axis equal
axis off
```

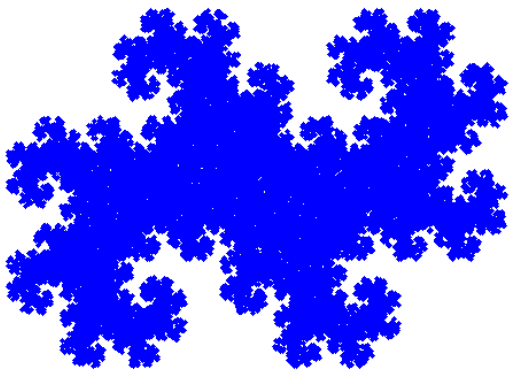

Figura 1.13 Iterated Function System.

```
function gasket(Pa,Pb,Pc,level)
%GASKET  Sierpinski gasket mediante ricorsione.
%        PA, PB and PC vettori a due componenenti
%        che definiscono i vertici del tiangolo
%        LEVEL livello di ricorsione
% Ad esempio PA=[0 0]; PB=[1 0]; PC=[0.5 1];
if level == 0
  % riempi il triangolo con vertici Pa, Pb, Pc.
  fill([Pa(1),Pb(1),Pc(1)],[Pa(2),Pb(2),Pc(2)],[0.5 0.5 0.5]);
  hold on
else
  % Ricorsione
  gasket(Pa,(Pa+Pb)/2,(Pa+Pc)/2,level-1)
  gasket(Pb,(Pb+Pa)/2,(Pb+Pc)/2,level-1)
  gasket(Pc,(Pc+Pa)/2,(Pc+Pb)/2,level-1)
end
axis('equal','off')
```

Si toglie successivamente l'interno del triangolo con vertici corrispondenti ai punti di mezzo. Provare con `level=0` e `level=1`, ecc. (porre successivamente `level off`).

Fiocco di neve di Kock

```
% Koch Curve, fiocco di neve
function cfkoch(n)
s=get(0,'ScreenSize');
set(gcf,'Position',[0 0 s(3) s(4)-70],'Color',[0 0 0])
if (n==0)
   x=[0;1];
```

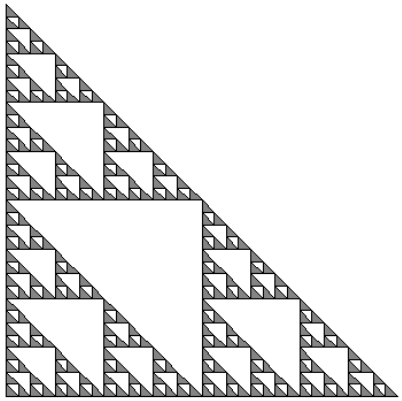

Figura 1.14 Sierpinski gasket.

```
   y=[0;0];
   line(x,y,'Color','w');
   good_axis
   set(gcf,'Nome','curva di Koch: Initiator (n=0)')
else
   levelcontrol=10^n;
   L=levelcontrol/(3^n);
   l=ceil(L);
   kline(0,0,levelcontrol,0,l);
   good_axis
   if(n==1)
      set(gcf,'Nome','curva di Koch: Generator (n=1)')
   else
      N=num2str(n);
      NN=strcat('curva di Koch ',' n=',N);
      set(gcf,'Nome',NN)
   end
end
------
function good_axis
axis equal
set(gca,'Visible','off')
-----
function plotline(a1,b1,a2,b2)
x=[a1;a2];
y=[b1;b2];
line(x,y,'Color','w');
-------
function kline(x1,y1,x5,y5,limit)
length=sqrt((x5-x1)^2+(y5-y1)^2);
if(length>limit)
   x2=(2*x1+x5)/3;
   y2=(2*y1+y5)/3;
   x3=(x1+x5)/2-(y5-y1)/(2.0*sqrt(3.0));
   y3=(y1+y5)/2+(x5-x1)/(2.0*sqrt(3.0));
```

```
   x4=(2*x5+x1)/3;
   y4=(2*y5+y1)/3;
   % ricorsione
   kline(x1,y1,x2,y2,limit);
   kline(x2,y2,x3,y3,limit);
   kline(x3,y3,x4,y4,limit);
   kline(x4,y4,x5,y5,limit);
else
   plotline(x1,y1,x5,y5);
end
```

Felce, fern

La felce è generata da trasformazioni ripetute di un punto nel piano. Se $\mathbf{x}$ è un vettore di componenti x_1, x_2 vi sono quattro differenti trasformazioni della forma

$$\mathbf{x} \rightarrow \mathbf{A}\mathbf{x} + \mathbf{b}$$

con differenti matrici $\mathbf{A}$ e vettori $\mathbf{b}$ (*trasformazioni affini*).

La trasformazione più frequentemente utilizzata è la seguente

$$\mathbf{A} = \begin{bmatrix} 0.85 & 0.04 \\ -0.04 & 0.85 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 0.0 \\ 1.6 \end{bmatrix}$$

Tale trasformazione accorcia e ruota di 'un bit' il vettore e aggiunge 1.6 alla seconda componente. Ripetute applicazioni della trasformazione muove il punto in su e a destra. 'Ogni tanto' si sceglie in maniea random una delle altre tre trasformazioni che hanno l'effetto di muovere il punto nella sottofelce inferiore sulla destra, e rispettivamente sulla sinistra, oppure sul gambo.

```
function fern
%FERN  implementazione MATLAB del Fractal Fern
%   Michael Barnsley, Fractals Everywhere, Academic Press, 1993.
%   il programma è eseguito fino allo stop
shg
clf reset
set(gcf,'color','white','menubar','none', ...
   'numbertitle','off','name','Fractal Fern')
x = [.5; .5];
h = plot(x(1),x(2),'.');
darkgreen = [0 2/3 0];
set(h,'markersize',1,'color',darkgreen,'erasemode','none');
axis([-3 3 0 10])
axis off
stop = uicontrol('style','toggle','string','stop', ...
   'background','white');
drawnow
p  = [ .85   .92   .99   1.00];
A1 = [ .85   .04; -.04   .85];  b1 = [0; 1.6];
A2 = [ .20 -.26;   .23   .22];  b2 = [0; 1.6];
A3 = [-.15   .28;   .26   .24];  b3 = [0; .44];
A4 = [  0     0 ;    0    .16];
cnt = 1;
```

```
tic
while ~get(stop,'value')
   r = rand;
   if r < p(1)
      x = A1*x + b1;
   elseif r < p(2)
      x = A2*x + b2;
   elseif r < p(3)
      x = A3*x + b3;
   else
      x = A4*x;
   end
   set(h,'xdata',x(1),'ydata',x(2));
   drawnow
   cnt = cnt + 1;
end
t = toc;
s = sprintf('%8.0f points in %6.3f seconds',cnt,t);
text(-1.5,-0.5,s,'fontweight','bold');
set(stop,'style','pushbutton','string','close','callback','close(gcf)')
```

1.5.6 Ulteriori esempi illustrativi

```
----------------
% uso della feval
fun=['cos'; 'sin'; 'log'];
k=input('numero funzione: ');
x=input('valore di ingresso: ');
feval(fun(k,:),x)


function y=haa(x,j,k)
% wavelet haar
n=length(x);
t=(2^j)*x-k;
jm=2^(j/2);
y=zeros(size(x));
I=find(t>0 & t<0.5);
y(I)=jm;
II=find(t>=0.5 & t<1);
y(II)=-jm;
```

Esempio di utilizzo del `switch statement`

```
x=ceil(10*rand);
switch x
      case {1,2}
         disp('probabilità = 20%')
      case {3,4,5}
         disp('probabilità = 30%')
      otherwise
         disp('probabilità = 50%')
end
```

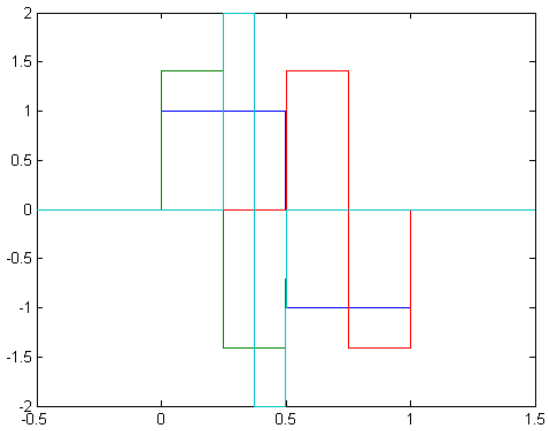

Figura 1.15 Wavelet di Haar.

```
%menu2.m
%help menu
scelta=menu('scegli una figura piana', 'triangolo','quadrato',...
    'pentagono','cerchio','stop');

t=linspace(pi/2,5*pi/2,361);
switch scelta
    case 1
        lati=3;
    case 2
        lati=4;
    case 3
        lati=5;
    case 4
        lati=360;
    otherwise
        close all
        return
end
angolo=360/lati;
x=cos(t(1:angolo:361));
y=sin(t(1:angolo:361));
fill(x,y,'r');
axis([-1 1 -1 1]);
axis('square');
clear scelta;
menu2


%try... catch
try
    z
catch
    disp('z non e'' definita')
end
```

```
z=0;
try
    z
catch
    disp('z non e'' definita')
end
```

Istruzione `persistent`

```
function a= perman
persistent b
if b>=1;
     b=1+b;
 else
     b=[b 1];
 end
     a=b;
```

Il parametro di uscita `a` conta il numero di accessi alla funzione `perman`. L'istruzione `persistent` permette di ritenere in memoria la variabile `b`. Differisce dall'istruzione `global` per il fatto che le variabili persistent sono note solo alla funzione nella quale sono dichiarate.

Media Aritmetica-Geometrica

La media aritmetica-geometrica $M(a,b)$ (AGM(a,b), agm(a,b)) di due numeri a e b è definita nel modo seguente

$$a_0 = a, \quad b_0 = b$$
$$a_{n+1} = \frac{a_n + b_n}{2}, \quad b_{n+1} = \sqrt{a_n\, b_n}$$

```
function M=arigeo(a,b)
% a,b vettori dello stesso ordine
while any(abs(a-b) > 1.e-10)
    an=(a+b)/2;
    bn=sqrt(a.*b);
    a=an;b=bn;
end
M=b;

%%%% variante ottimizzata
function M=arigeo(a,b)
I=find(abs(a-b)>1.e-12);
while  ~isempty(I)
    an(I)=(a(I)+b(I))/2;
    bn(I)=sqrt(a(I).*b(I));
    a(I)=an(I);b(I)=bn(I);
    I=find(abs(a-b)>1.e-12)
end
M=b;
```

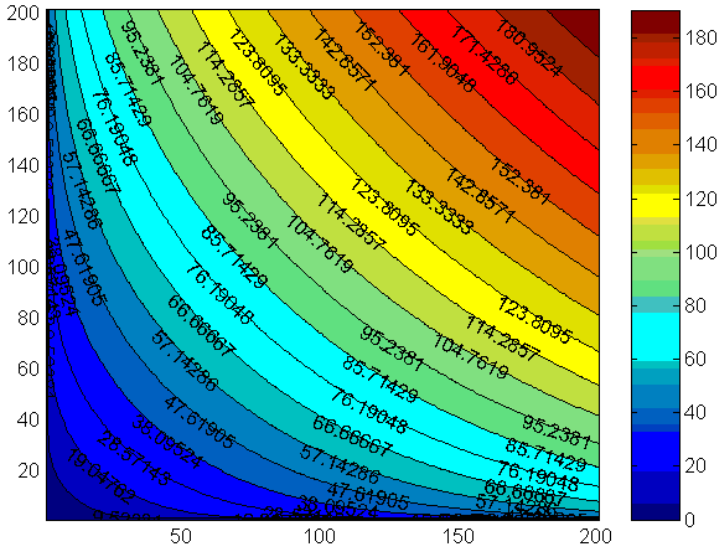

Figura 1.16 Media aritmetica-geometrica $M(a,b)$ per $a \le a, b \le 200$.

Le successioni a_n, b_n sono monotone e limitate, inoltre si ha

$$a_{n+1} - b_{n+1} = \frac{1}{2}(a_n + b_n) - \sqrt{a_n b_n} = \frac{a_n - 2\sqrt{a_n b_n} + b_n}{2}$$

Tenendo conto che $\sqrt{b_n} < \sqrt{a_n}$ si ha

$$a_{n+1} - b_{n+1} < \frac{1}{2}(a_n - b_n)$$

La Figura 1.16 mostra $M(a,b)$ per $a \le a, b \le 200$.

```
[x,y]=meshgrid(0:01:200,0:01:200);
z=arigeo(x,y);
[c,h]=contourf(z,20);clabel(c,h);colorbar
```

Tra i vari risultati interessanti relativi all'algoritmo della media aritmetica-geometrica ricordiamo la possibilità di calcolare gli *integrali ellittici*.

1.5.7 Strutture dati: array di celle e array di strutture

Array di celle

Una array di celle (`cell array`) è una array nella quale gli elementi sono celle (*cells*), ossia contenitori di matrici, testi, vettori di numeri complessi, ecc.

```
A(1,1)={rand(2)}; A(1,2)={'text'};
A(2,1)={rand(5)}; A(2,2)={2+3i};
-------
```

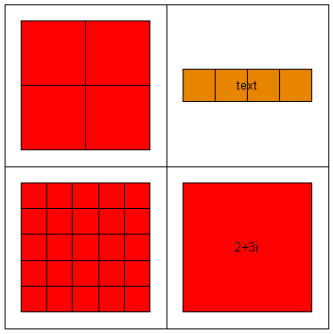

Figura 1.17 cellplot.

```
A = [2x2 double]    'text'
    [5x5 double]    [2.0000+ 3.0000i]
A{1,1}
ans =0.9501    0.6068
     0.2311    0.4860
-------
celldisp(A)
A{1,1} = 0.9501    0.6068
         0.2311    0.4860
A{2,1} = 0.8913    0.4447    0.1763    0.8936    0.1389
         0.7621    0.6154    0.4057    0.0579    0.2028
         0.4565    0.7919    0.9355    0.3529    0.1987
         0.0185    0.9218    0.9169    0.8132    0.6038
         0.8214    0.7382    0.4103    0.0099    0.2722
A{1,2} = text
A{2,2} = 2.0000 + 3.0000i
-------
cellplot(A)  %cfr. figura
```

Array di strutture

Il significato di una array di strutture (`structure array`) è illustrato dai seguenti esempi

```
 A=struct('d',16,'m',23,'s',47)
 A = d: 16
     m: 23
     s: 47
 A.d
 ans = 16
 A.m
 ans = 23
---------
 s = struct('strings',{{'hello','yes'}},'lengths',[5 3])
 s = strings: {'hello'  'yes'}
    lengths: [5 3]
 s.lengths
 ans = 5     3
```

```
 s.lengths(2)
ans =  3
```

Mentre per le array di celle è possibile accedere ai dati contenuti in base alla posizione che occupano, con le array di strutture è possibile ai dati anche attraverso i loro nomi. Questa caratteristica consente di creare e utilizzare i database che contengono tipi di dati differenti (per esempio, un elenco di clienti con i loro indirizzi e numeri di telefono).

Esempio 1.11 Il seguente m-file `fstru` calcola il valore, il gradiente e la matrice hessiana della funzione

$$f(x) = (x_1 - 1)^2 + x_1 x_2$$

e li memorizza in una struttura

```
function fx=fstru(x)
fx.Value=(x(1)-1).^2+x(1).*x(2);
fx.Gradient=[2*(x(1)-1)+x(2); x(1)];
fx.Hessian=[2 1; 1 0];
```

Ad esempio

```
>> x=[2;1];
>> fx=fstru(x)
fx =
       Value: 3
    Gradient: [2x1 double]
     Hessian: [2x2 double]
>> whos
  Name      Size                    Bytes  Class
  fx        1x1                       428  struct array
  x         2x1                        16  double array
Grand total is 12 elements using 444 bytes
>> fx.Gradient
ans =3
     2
>> fx.Hessian
ans =2     1
     1     0
```

♦ **Esercizio 1.9** *Creare una structure array che contiene i fattori di conversione delle unità di massa, forza e lunghezza fra sistema metrico SI e quello anglosassone.*

Utilizzare l'array per calcolare:

- *il numero di metri in 24 piedi (feet)*
- *il numero di piedi in 65 metri*
- *il numero di libbre (pound) equivalente a 18 newton*
- *il numero di newton equivalente a 5 libbre*
- *il numero di chilogrammi in 6 slug (slug è l'unità di massa che subisce un'accelerazione di 1 piede/sec^2 quando è sottoposta alla forza di una libbra)*
- *il numero di slug in 15 chilogrammi*