Agostino De Marco
Domenico P. Coiro

Elementi

di

Dinamica e simulazione di volo

Quaderno 5

Richiami di calcolo numerico

Marzo 2017
ver. 2017.a

Dichiarazione di Copyright

Quaderno 5

Richiami di calcolo numerico

Non esiste vento favorevole per il marinaio che non sa dove andare.
– Seneca

Indice

In questa appendice si faranno dei richiami ad alcuni argomenti di calcolo numerico e di metodi matematici per l'ingegneria. Lo scopo è quello di introdurre il lettore ad un uso ragionato di alcuni strumenti di calcolo molto comuni oltre che ad alcuni programmi sviluppati e messi a disposizione dagli autori.

5.1 Interpolazione

Nelle scienze e nell'ingegneria si dispone spesso di una sequenza di valori di una data grandezza fisica y in corrispondenza di altrettanti valori di un'altra grandezza fisica x che gioca il ruolo di variabile indipendente. Si prenda in esame la situazione rappresentata nella figura 5.1.

Spesso questo tipo di corrispondenza è chiamato 'funzione nota per punti' o 'funzione tabellare' (o anche 'tabulare'). Una funzione può essere nota solo per punti perché deriva da misurazioni sperimentali oppure perché è soluzione numerica di un problema matematico. In altre parole non si ha un'espressione analitica della funzione e si deve risolvere un problema di rappresentazione di dati.

Approssimare una funzione tabellare, cioè a dire una corrispondenza che può essere comunque complicata, vuol dire sostituire ad essa una funzione semplice, analitica, facilmente calcolabile. La *funzione approssimante* dovrà essere esprimibile come combinazione di un numero finito di funzioni che siano facili da calcolare, derivare ed integrare,

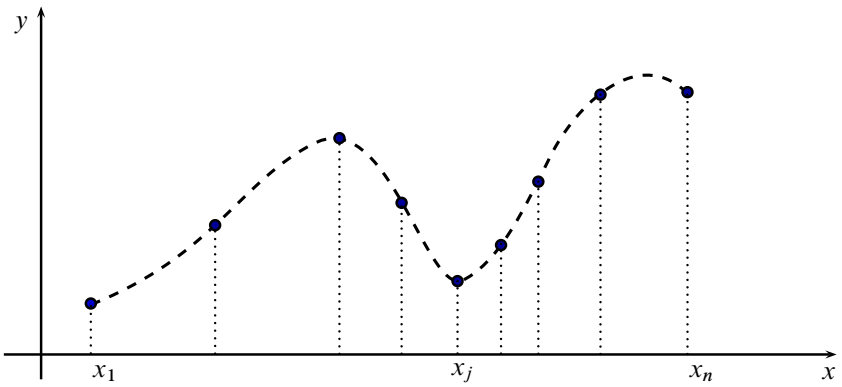

Figura 5.1 Con interpolazione esatta si intende il processo che arriva ad individuare una funzione, spesso polinomiale a tratti, che passi per un insieme dato di punti: (x_j, y_j), $j = 1, \ldots, n$.

mediante algoritmi robusti ed efficienti. Una volta individuata la funzione analitica passante per l'insieme di punti, sarà possibile calcolare il valore della funzione stessa in un nuovo punto interno all'intervallo. Tale azione è detta *interpolazione* e la funzione approssimante potrà essere detta anche *funzione interpolante*.

Detto n il numero di campioni a disposizione, se si vuole rappresentare la funzione nota per punti con un grafico, non si può fare altro che disegnare un diagramma come quello riportato nella figura 5.2. Tale grafico è ottibile in Matlab con la semplice sequenza di comandi riportati nel listato 5.1. Nell'esempio è stato semplicemente costruito un vettore x di $n = 12$ valori equispaziati, a partire da 1 fino a 12, e si è definito un secondo vettore y di altrettanti numeri. Questi ultimi potrebbero essere interpretati come i valori di temperatura media in una stanza nell'arco di 12 ore, misurati ad intervalli di un'ora.

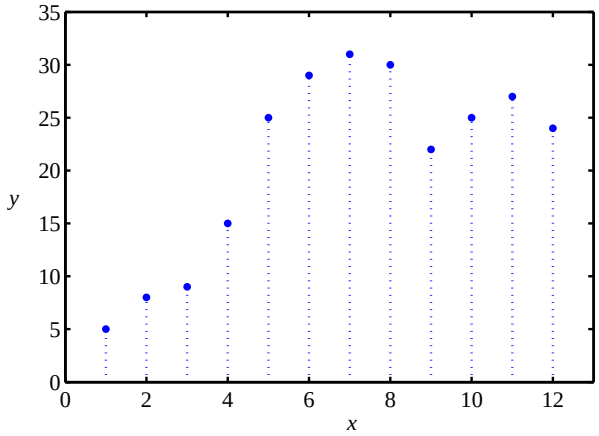

Figura 5.2 Esempio di coppie di dati (x_j, y_j), per $j = 1 \ldots n$, dalle quali si vuole costruire una corrispondenza $y = f(x)$.

Listato 5.1 grafico a barre di una semplice sequenza di coppie (x, y).

```
% n coppie (x,y)
x = 1:12; % ← definisce le x_j, con n = 12
y=[5 8 9 15 25 29 31 30 22 25 27 24]; % ← le y_j
plot(x,y,'o')
xlabel('x'); ylabel('y');
```

Se si assume che esista una legge quantitativa che lega l'insieme dei numeri x_j a quello dei numeri y_j, si può descrivere tale relazione quantitativa come: $y_j = f_{\text{tab}}(x_j)$, con $j = 1, \ldots, n$. In pratica la funzione tabellare f_{tab} non esprime altro che l'insieme discreto di coppie di punti $\{(x_j, y_j)\}_{j=1,\ldots,n}$. Parlare di interpolazione significa voler costruire un modello matematico, cioè una funzione approssimante $g(x)$, che descriva sufficientemente bene il fenomeno o il contesto che determina le coppie (x_j, y_j). Un tale modello deve consentire di fare delle previsioni sul valore della variabile dipendente y per

valori di x diversi dai campioni x_j della variabile indipendente. Nel contesto della teoria dell'interpolazione le coppie (x_j, y_j) prendono il nome di *punti di supporto* e le ascisse di supporto x_j vengono chiamate anche *nodi*.

Si osservi che quando si vuole che la g assuma esattamente valori y_j in corrispondenza delle ascisse di supporto, cioè $y_j = g(x_j)$, si parla di *interpolazione esatta*. Non sempre l'interpolazione esatta rappresenta un approccio corretto. Ad esempio se le grandezze x ed y rappresentano dei valori misurati e sono soggette ad errore sperimentale, una funzione interpolante esatta seguirebbe l'andamento casuale dell'errore. In simili situazioni si sceglie un approccio basato sulla cosiddetta *interpolazione approssimata* (*data fitting*).

Si osservi inoltre che se si vuole determinare la $g(x)$ per valori esterni all'intervallo delle ascisse di supporto, cioè per $x < x_1$ oppure per $x > x_n$, si parla allora, in particolare, di *estrapolazione*.

La scelta del modello di funzione approssimante è condizionata da considerazioni legate al particolare problema da risolvere, ma anche da considerazioni numeriche dovute al fatto che la funzione approssimante deve essere facilmente calcolabile. Nei paragrafi che seguono, con l'aiuto di alcuni esempi in linguaggio Matlab, si faranno dei richiami sull'interpolazione esatta mediante i tipi più comuni di funzioni approssimanti. Successivamente si daranno dei cenni alle diverse tecniche di interpolazione approssimata.

5.1.1 Interpolazione lineare

Spesso si ritiene accettabile una funzione approssimante che sia lineare a tratti. Si avrà in tal caso che la funzione approssimante, detta g_{lin}, avrà una forma del tipo

$$g_{\text{lin}}(x) = \begin{cases} a'x + b' & \text{se } x < x_1 \\ a_1 x + b_1 & \text{se } x_1 \le x < x_2 \\ a_2 x + b_2 & \text{se } x_2 \le x < x_3 \\ \dots & \dots \\ a_{n-1} x + b_{n-1} & \text{se } x_{n-1} \le x \le x_n \\ a''x + b'' & \text{se } x_n < x \end{cases} \tag{5.1}$$

Come è facile far vedere in base a considerazioni geometriche semplici, se si assume che il tratto di curva che unisce due punti consecutivi è una retta, nel generico intervallo $[x_j, x_{j+1}]$ si avrà

$$g_{\text{lin}}(x) = y_j \frac{x - x_{j+1}}{x_j - x_{j+1}} + y_{j+1} \frac{x - x_j}{x_{j+1} - x_j} \tag{5.2}$$

per $j = 1, \dots, n-1$. Dalle (5.2) si possono ricavare facilmente le espressioni dei coefficienti a_j, b_j, a', b', a'' e b'' che compaiono nelle (5.1).

L'interpolazione di una funzione tabellare di una variabile è facilmente ottenibile in Matlab con la funzione `interp1`. Con riferimento all'esempio precedente, figura 5.2, il comando:

```
>> interp1(x,y,4.5)
ans =
    20
```

valuta di *default* la $g_{\text{lin}}(x)$ in corrispondenza del valore scalare passato alla funzione come terzo argomento, cioè restituisce $g_{\text{lin}}(4.5)$. La chiamata dell'esempio precedente

può essere effettuata equivalentemente con il comando `interp1(x,y,4.5,'linear')`, in cui si passa alla funzione `interp1` una stringa come quarto argomento per richiedere esplicitamente di selezionare una funzione approssimante lineare a tratti. È possibile passare anche un quinto argomento nel caso in cui si richieda di estrapolare la funzione tabellare assegnata. A titolo di esempio si esamini il risultato del comando seguente:

```
>> interp1(x,y,0.0,'linear','extrap')% ← valuta g_lin(x) per x < x_1
ans =
     2
```

Generalizzando e considerando che tipicamente le funzioni native di Matlab possono ricevere argomenti di tipo vettoriale, si può pensare di effettuare una chiamata ad `interp1` con passaggio, come terzo argomento, al posto di uno scalare, di un intero vettore `xi` di valori da interpolare. Il risultato sarà un vettore riga che conterrà i valori che la funzione approssimante selezionata assume in ciascun elemento di `xi`. Facendo ancora riferimento all'esempio riportato nella figura 5.2, implementato nel listato 5.1, sarà possibile scrivere una sequenza di comandi come quelli del listato 5.2 ed ottenere il diagramma della figura 5.3. Nell'esempio il vettore `xi` conterrà un numero maggiore di valori rispetto al numero di campioni contenuti nel vettore `x`.

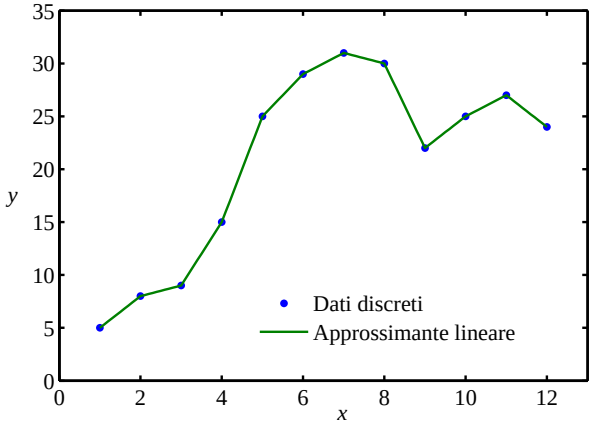

Figura 5.3 Esempio di interpolazione lineare.

Listato 5.2 Esempio d'uso della funzione `interp1`.

```
% n coppie (x, y)
x = 1:12; % ← definisce le x_j, con n = 12
y=[5 8 9 15 25 29 31 30 22 25 27 24]; % ← le y_j

% intervallo delle x infittito
xi=1:0.1:12;        % 111 elementi
y1=interp1(x,y,xi); % ← valuta g_lin

plot(x,y,'o',xi,y1,'-') % diagramma
```

5.1.2 Interpolazione polinomiale

Interpolazione polinomiale classica

Anziché concepire una funzione analitica come g_{lin}, definita diversamente per diversi tratti dell'intervallo delle ascisse di supporto, può risultare naturale pensare alla definizione di funzione polinomiale

$$P_n(x) = a_0 + a_1\,x + a_2\,x^2 + \cdots + a_n\,x^n \tag{5.3}$$

Per definire univocamente il polinomio (5.3) di grado n è necessario determinarne gli $n+1$ coefficienti. Dunque se si vuole trovare una funzione approssimante $g_{\text{pol}}(x)$ definita globalmente come polinomio, trattandosi di interpolazione esatta, si hanno a disposizione n condizioni: $y_j = g_{\text{pol}}(x_j)\ (j = 1, \ldots, n)$. Esse possono essere utilizzate per determinare gli n coefficienti di un polinomio di grado $n-1$. Interpolazione polinomiale esatta di una funzione tabellare definita in n punti significa quindi determinare una $g_{\text{pol}} = P_{n-1}(x)$.

Le condizioni suddette, sempre che si abbiano punti di supporto con ascisse tutte distinte, determineranno univocamente gli n coefficienti $(a_0, a_1, \ldots, a_{n-1})$ del polinomio. Come è noto dall'Analisi matematica il vettore colonna $\boldsymbol{c}$ che ha per elementi i coefficienti del polinomio g_{pol} sarà dato dalla

$$\boldsymbol{c} = \boldsymbol{M}^{-1}\boldsymbol{y}\,, \qquad \text{con } \boldsymbol{M} = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix} \quad \text{ed } \boldsymbol{y} = \begin{Bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{Bmatrix} \tag{5.4}$$

Il vettore $\boldsymbol{y}$ nella (5.4) contiene i valori y_j della funzione tabellare mentre la matrice $\boldsymbol{M}$ è la *matrice di Vandermorde*, i cui elementi sono potenze delle ascisse di supporto x_j.

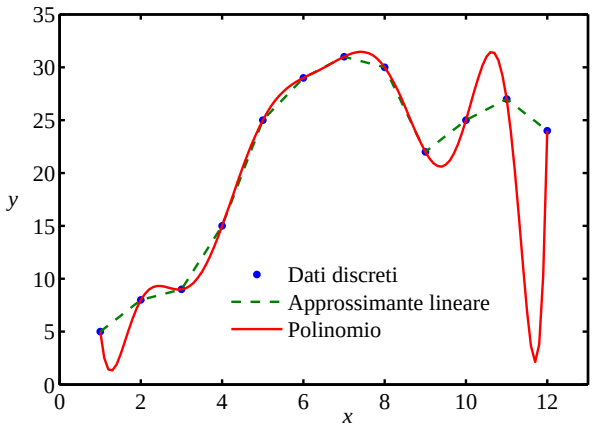

Figura 5.4 Esempio di interpolazione polinomiale.

Listato 5.3 Interpolante polinomiale determinata con i comandi `vander` e `polyval`.

```
% n coppie (x,y)
x = 1:12; % ← definisce le x_j, con n = 12
y=[5 8 9 15 25 29 31 30 22 25 27 24]; % ← le y_j

% matrice di Vandermorde A(i,j) = x(i)^(n-j),
A = vander(x);
% coefficienti del polinomio interpolante
c = inv(A)*y';
% intervallo delle x infittito
xi=1:0.1:12;
% valuta il polinomio di grado n-1
y0=polyval(c,xi);

% interpolante lineare a tratti
y1=interp1(x,y,xi);
```

Interpolazione polinomiale di Lagrange

Per completezza, e probabilmente perché qualche lettore può aver incontrato un simile tipo di ricostruzione di funzioni in altri corsi di base, è il caso di richiamare qui anche la tecnica di interpolazione polinomiale di Lagrange.

Un'alternativa al polinomio tradizionale del paragrafo precedente è costituita da una funzione interpolante che sia esprimibile come somma di polinomi. In tal senso si può immaginare una funzione come un elemento di uno spazio di funzioni, la cui base è un insieme di funzioni notevoli. Tra le possibili scelte della base di funzioni vi è quella dei polinomi di Lagrange. Un generico, i-mo, polinomio di Lagrange $L_{n,i}(x)$ si costruisce a partire da n valori reali ed è definito in modo tale da assumere valore 1 per $x = x_i$ e valori nulli nelle rimanenti ascisse di supporto, x_j per $j \neq i$. In altri termini

$$L_{n,i}(x) = \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)} = \frac{(x - x_1)(x - x_2)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_n)}{(x_i - x_1)(x_i - x_2)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)} \tag{5.5}$$

L'espressione fratta data dalla (5.5), per ascisse di supporto distinte, non dà mai luogo a singolarità di alcun tipo poiché nella produttoria viene saltato il termine $i = j$. Ciascuna funzione $L_{n,i}$ è dunque un polinomio di grado $n - 1$.

In definitiva, una funzione interpolante g_{L} potrà essere espressa come

$$g_{\mathrm{L}}(x) = \sum_{i=1}^{n} y_i L_{n,i}(x) \tag{5.6}$$

cioè come somma di n polinomi di Lagrange, ciascuno pesato con la rispettiva ordinata y_i. Ciò assicura che la g_{L} sia una funzione interpolante esatta.

Considerazioni sull'uso di polinomi interpolanti

Come si vede dall'esame della figura 5.4, ottenuta in Matlab con i comandi del listato 5.3, l'interpolazione polinomiale può essere spesso insoddisfacente. La funzione interpolante polinomiale presenterà delle oscillazioni indesiderate, tanto più pronunciate quanto più è alto il grado del polinomio. Inoltre, l'operazione di estrapolazione con polinomi interpolanti è molto pericolosa in quanto può portare alla stima di valori completamente scorretti. Si pensi infatti che per valori sufficientemente grandi della variabile indipendente x la funzione $g_{\mathrm{pol}}(x)$ tende comunque all'infinito.

Ovviamente non è detto che se i punti di supporto sono n il polinomio interpolante debba essere di grado $n-1$. Si pensi ad esempio a n punti di supporto posti sopra una parabola. Il polinomio interpolante è unico e coincide con la parabola stessa. Il polinomio passante per n punti di supporto distinti è *al più* di grado $n-1$.

Qualora i punti di supporto provengano da dati affetti da errore sperimentale, secondo quanto già sottolineato, non è opportuno applicare la procedura di interpolazione esatta. In tali casi il problema da risolvere è completamente diverso e cade nella sfera delle *regressioni* (lineari o non lineari). Nella pratica, dato un modello come ad esempio quello polinomiale, l'obiettivo della procedura di regressione è quello di minimizzare la distanza esistente tra i dati sperimentali ed il modello proposto. In generale, una volta determinati i parametri del modello proposto, questi *non* passa per i dati sperimentali.

Riguardo alla tecnica di interpolazione polinomiale esatta, che sia essa basata sui polinomi classici o sui polinomi di Lagrange, è bene sottolineare qui alcuni punti. In primo luogo, alcune classi di funzioni *non* vengono interpolate adeguatamente utilizzando i polinomi. Ciò lo si intuisce ad esempio dalla figura 5.4. Le classi di funzioni meno adatte all'interpolazione polinomiale sono quelle dotate di asintoti (orizzontali, verticali, obliqui). In questi casi si preferisce optare per l'interpolazione tramite *spline* di cubiche. In secondo luogo, la qualità dell'interpolazione non migliora in genere aumentando il grado del polinomio interpolante. Anzi si incorre nel rischio opposto.

In sintesi, l'esperienza ha confermato che:

- non conviene usare polinomi interpolanti di grado superiore a $3 \div 5$,
- l'interpolazione può diventare meno accurata verso i bordi dell'intervallo, rispetto alla zona centrale,
- quando è possibile vanno scelte in modo opportuno le ascisse di supporto,
- l'errore può essere molto elevato al di fuori dell'intervallo di interpolazione; per questo motivo occorre evitare ogni forma di estrapolazione.

Si osservi come spesso nei testi scientifici specialistici vengano riportate tabelle di proprietà chimico-fisiche delle sostanze (densità, viscosità, calore specifico, conducibilità termica, eccetera). Si pensi ad esempio ad una descrizione realistica delle proprietà dell'aria al variare della quota. Tali tabelle di valori provengono da banche dati sot-

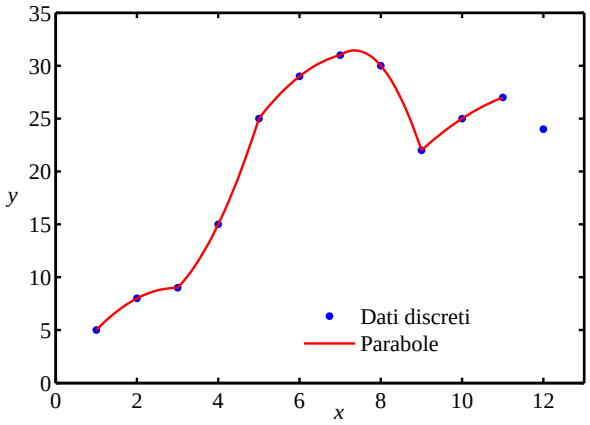

Figura 5.5 Interpolazione a tratti con degli archi di parabola. Il numero di punti di supporto non permette di coprire l'intero intervallo con sole parabole.

to forma polinomiale. Contestualmente, tali banche dati, riportano per ogni proprietà l'intervallo della variabile indipendente all'interno del quale è consentito utilizzare una specifica formula di interpolazione polinomiale. Occorre evitare quindi qualsiasi forma di estrapolazione, cioè di allargamento del *range* di validità specificato.

5.1.3 Interpolazione polinomiale a tratti: *spline* di cubiche

Come detto in precedenza non è opportuno utilizzare polinomi interpolanti di ordine maggiore a $3 \div 5$. Se però i punti di supporto sono numerosi e si utilizza un solo polinomio in tutto l'intervallo il suo grado cresce inevitabilmente. Una valida alternativa è quella di utilizzare dei sotto-intervalli ed al loro interno interpolare con polinomi di grado inferiore. Si ottiene in questo caso una *interpolazione a tratti* (*piecewise polynomial interpolation*).

Nell'esempio della figura 5.5 è riportata una interpolazione a tratti con parabole. Ogni parabola sottende tre punti di supporto aventi ciascuno ascisse distinte. L'esempio mostra che questo tipo di interpolazione non è adatto a quei casi in cui l'intervallo di supporto non è suddivisibile in sotto-intervalli in cui ogni intervallo è caratterizzato da tre punti. In ogni caso la sola condizione di passaggio per i punti di supporto non assicura la "morbidezza" (*smoothness*) della curva totale risultante. È cioè assicurata soltanto la continuità della funzione approssimante ma non quella delle derivate.

Di fatto esiste la possibilità di disporre di una interpolazione a tratti, con polinomi di grado non elevato, che sia moderatamente oscillante e che produca alla vista un senso di morbidezza. I polinomi relativi ai diversi tratti della funzione approssimante saranno definiti nei rispettivi sotto-intervalli delimitati da due ascisse di supporto consecutive. Inoltre, per le coppie di polinomi consecutivi varranno delle condizioni di continuità per le derivate successive.

Dal punto di vista matematico, si può porre il seguente problema: dati n punti di supporto (x_j, y_j) aventi ascisse in ordine crescente, si desidera identificare una funzione polinomiale a tratti $g(x)$, continua in $[x_1, x_n]$ con le sue derivate prima e seconda $g'(x)$ e $g''(x)$, e tale che $g(x_j) = y_j$ con $j = 1, \ldots, n$. Dal momento che esiste più di una possibile funzione che soddisfa le proprietà precedenti, si desidera identificare tra queste quelle che minimizzano l'integrale

$$I\left(g''\right) = \int_{x_1}^{x_n} |g''(x)|^2 \mathrm{d}x$$

della derivata seconda della g. Si osservi che un piccolo valore di $I(g'')$ significa che la derivata prima g' non cambia velocemente e cioè che la g non presenta eccessive oscillazioni.

È possibile dimostrare che esiste una sola soluzione al problema posto. Tale soluzione è detta *spline naturale di cubiche* e verrà detta g_{spline}.

La *spline* di cubiche non è altro che un insieme di polinomi di terzo grado, ciascuno definito nei diversi sotto-intervalli dell'intervallo di supporto e ben raccordati in corrispondenza dei punti di adiacenza. La generica funzione cubica della *spline* sarà detta

$$s_j(x) = a_j(x-x_j)^3 + b_j(x-x_j)^2 + c_j(x-x_j) + d_j\,, \qquad \text{con } j = 1,\ldots,n-1 \tag{5.7}$$

e sarà

$$g_{\text{spline}}(x) = \begin{cases} s_1(x) & \text{se } x_1 \le x \le x_2 \\ s_2(x) & \text{se } x_2 \le x \le x_3 \\ \ldots & \ldots \\ s_{n-1}(x) & \text{se } x_{n-1} \le x \le x_n \end{cases} \tag{5.8}$$

Dati n punti di supporto, per determinare le singole $s_j(x)$, vengono imposte le condizioni:

$$s_j(x_j) = y_j \qquad j = 1,\ldots,n-1 \tag{5.9}$$
$$s_j(x_{j+1}) = y_{j+1} \qquad j = 1,\ldots,n-1 \tag{5.10}$$
$$s_j'(x_{j+1}) = s_{j+1}'(x_{j+1}) \qquad j = 1,\ldots,n-2 \tag{5.11}$$
$$s_j''(x_{j+1}) = s_{j+1}''(x_{j+1}) \qquad j = 1,\ldots,n-2 \tag{5.12}$$

Si osservi che, viste le (5.7)-(5.8), il numero di parametri incogniti è $4(n-1)$ mentre le condizioni (5.9)-(5.12) finora imposte sono: $n-1+n-1+n-2+n-2 = 4n-6$. Quindi per determinare tutti i parametri della *spline* di cubiche occorre aggiungere altre due condizioni. È possibile soddisfare le ultime due condizioni in modo differente:

- imponendo derivate seconde nulle negli estremi

  $$s_1''(x_1) = s_{n-1}''(x_n) = 0$$

  l'interpolante prende allora il nome di *spline naturale*;
- assegnando valori particolari, C_1 e C_2, alle derivate seconde negli estremi,

  $$s_1''(x_1) = C_1 \quad \text{ed} \quad s_{n-1}''(x_n) = C_2$$

  l'interpolante prende il nome di *spline vincolata*;
- oppure ricostruendo il valore delle derivate seconde agli estremi in modo che ciascuno di essi sia proporzionale alla derivata seconda nel punto adiacente,

  $$s_1''(x_1) = \alpha s_1''(x_2)\,, \qquad s_{n-1}''(x_n) = \beta s_{n-1}''(x_{n-1})$$

  Spesso si adottano valori di α e β pari, entrambi, ad 1 o $\frac{1}{2}$.

Utilizzando le $4n-6$ condizioni precedentemente indicate più le due condizioni aggiuntive agli estremi, si ottiene un problema chiuso di $4(n-1)$ condizioni in altrettante

incognite. In pratica gli n valori delle derivate seconde s_j'' saranno dati dalla soluzione di un sistema algebrico lineare la cui matrice avrà la proprietà di essere una matrice sparsa, in particolare una matrice *tridiagonale*. Tale tipo di matrici viene invertita in maniera molto efficiente ed è questo uno dei motivi della popolarità delle interpolanti *spline* cubiche. Note le curvature nei nodi si ottengono agevolmente i parametri a_j, b_j, c_j e d_j nelle (5.7). Per approfondimenti e per maggiori particolari sulla tecnica di determinazione di tali coefficienti si rimanda ai riferimenti bibliografici di questo capitolo, ad esempio il testo di Comincioli [70].

Anche se in questi richiami non si scende nei dettagli matematici di questa tecnica di interpolazione, è bene che il lettore prenda coscienza dell'argomento poiché l'interpolazione polinomiale a tratti, in particolare quella con *spline* di cubiche, è quella che si adatta meglio alla maggioranza dei problemi che potrà incontrare nella pratica ingegneristica.

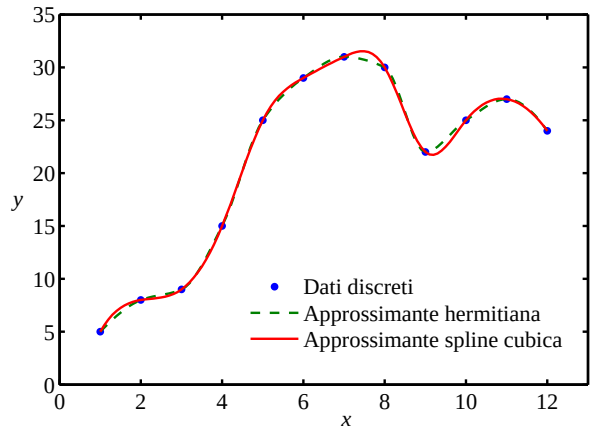

Figura 5.6 Esempi di interpolazione hermitiana e *spline* cubica.

Listato 5.4 Esempio d'uso della funzione `interp1` con opzioni `'pchip'` e `'spline'`.

```
% n coppie (x,y)
x = 1:12; % ← definisce le x_j, con n = 12
y=[5 8 9 15 25 29 31 30 22 25 27 24]; % ← le y_j

% intervallo delle x infittito
xi=1:0.1:12;          % 111 elementi
y2=interp1(x,y,xi,'pchip'); % ← valuta g_herm
y3=interp1(x,y,xi,'spline'); % ← valuta g_spline

plot(x,y,'o',xi,y2,'--',xi,y3,'-') % diagramma
```

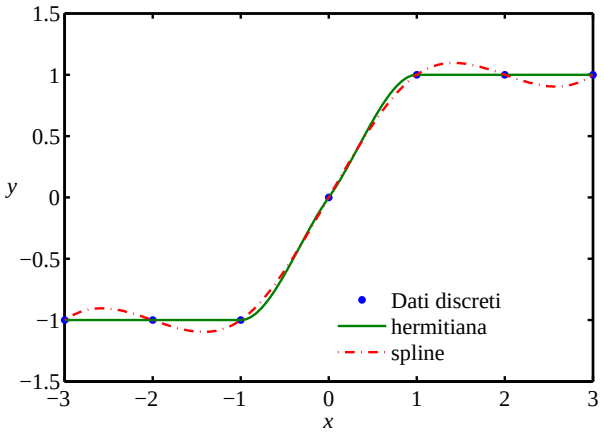

Figura 5.7 Caso in cui l'interpolazione hermitiana si presta meglio della *spline* cubica.

Listato 5.5 Esempio d'uso delle funzioni `pchip` e `spline`.

```
t = -3:3;
V = [-1 -1 -1 0 1 1 1];
ti = -3:.01:3;
p = pchip(t,V,ti);  % ritorna la struttura p
s = spline(t,V,ti); % ritorna la struttura s

plot(t,V,'o',ti,p,'-',ti,s,'-.')
```

La figura 5.6 mostra un grafico ottenuto con la sequenza di comandi implementata nel listato 5.4. Questo esempio mostra due ulteriori funzionalità della funzione `interp1` di Matlab. Passando alla funzione la stringa `'spline'` come quarto argomento si richiede esplicitamente di selezionare una funzione approssimante *spline* cubica. Con questa opzione l'utente di `interp1` può anche cercare di estrapolare dei valori passando come

terzo argomento un vettore di ascisse `xi` che cadono anche al di fuori dell'intervallo di supporto, senza esplicitamente passare `'extrap'` come quinto argomento.

La figura 5.6 mostra anche la possibilità di ottenere con `interp1` una funzione approssimante polinomiale a tratti di Hermite con l'opzione `'pchip'` (*piecewise cubic Hermite interpolation*). Tale tipo di interpolazione garantisce una funzione approssimante g_{herm} continua assieme alla sua derivata prima in tutti i punti dell'intervallo di supporto. La continuità della derivata seconda g''_{herm} non è però assicurata al contrario della *spline* di cubiche. Per approfindimenti si rimanda al testo di de Boor [72]. Nella pratica le interpolanti polinomiali a tratti di Hermite vengono usate quando si ha la necessità di preservare una certa forma della curva (*shape preserving interpolation*). Un esempio di insieme di dati in cui probabilmente l'opzione `'pchip'` fornisce risultati migliori è riportato nella figura 5.7, ottenuta con i comandi del listato 5.5

Qualora il lettore voglia accrescere la sua conoscenza dell'uso delle funzioni di interpolazione in Matlab ed in generale sulle tecniche di interpolazione, si suggerisce di esplorare le varie voci di *help*. Per quanto riguarda l'interpolazione con *spline* è utile esplorare le possibilità di lavoro offerte dallo *Spline toolbox*, con la sua interfaccia grafica richiamabile con il comando `splinetool`.

5.1.4 Interpolazione in più dimensioni

Quando si ha una funzione tabellare in più dimensioni il Calcolo numerico mette a disposizione diverse tecniche per la determinazione di valori interpolati in corrispondenza di valori non nodali delle variabili indipendenti. In più dimensioni le ascisse di supporto di ciascuna variabile indipendente si combineranno in una *griglia di supporto*.

Si consideri, per esempio, una funzione di due variabili nota per punti. Un generico nodo della griglia di supporto sarà un punto $P_{i,j} \equiv (x_i, y_j)$ del piano xy, per $i = 1, \ldots, n_x$ e $j = 1, \ldots, n_y$. Si assuma per semplicità che i nodi siano distribuiti regolarmente lungo ciascuna direzione della griglia. Un'estensione alle funzioni di due variabili della tecnica di interpolazione lineare a tratti è nota in Analisi numerica come *interpolazione bilineare*. L'idea alla base di questa tecnica è semplice: dato il punto $P \equiv (x, y)$ non appartenente alla griglia di supporto, un valore interpolato $z = g(x, y)$ sarà determinato effettuando prima un'interpolazione lineare unidimensionale lungo la direzione x, successivamente un'interpolazione lineare unidimensionale lungo la direzione y. La figura 5.8 aiuta a comprendere questo procedimento. Detti $z_{i,j}$, $z_{i+1,j}$, $z_{i,j+1}$, $z_{i+1,j+1}$ i valori noti della funzione tabelare in corrispondenza dei punti $P_{i,j}$, $P_{i+1,j}$, $P_{i,j+1}$, $P_{i+1,j+1}$, rispettivamente. La prima interpolazione porterà a conoscere i due valori

$$\begin{aligned} Z_A &= \frac{x_{i+1} - x}{x_{i+1} - x_i} z_{i,j} + \frac{x - x_i}{x_{i+1} - x_i} z_{i+1,j} \\ Z_B &= \frac{x_{i+1} - x}{x_{i+1} - x_i} z_{i,j+1} + \frac{x - x_i}{x_{i+1} - x_i} z_{i+1,j+1} \end{aligned} \tag{5.13}$$

La successiva interpolazione lungo la direzione y

$$g(x, y) = \frac{y_{j+1} - y}{y_{j+1} - y_j} Z_A + \frac{y - y_j}{y_{j+1} - y_j} Z_B \tag{5.14}$$

fornirà il valore desiderato della funzione interpolante. Esso può essere espresso per

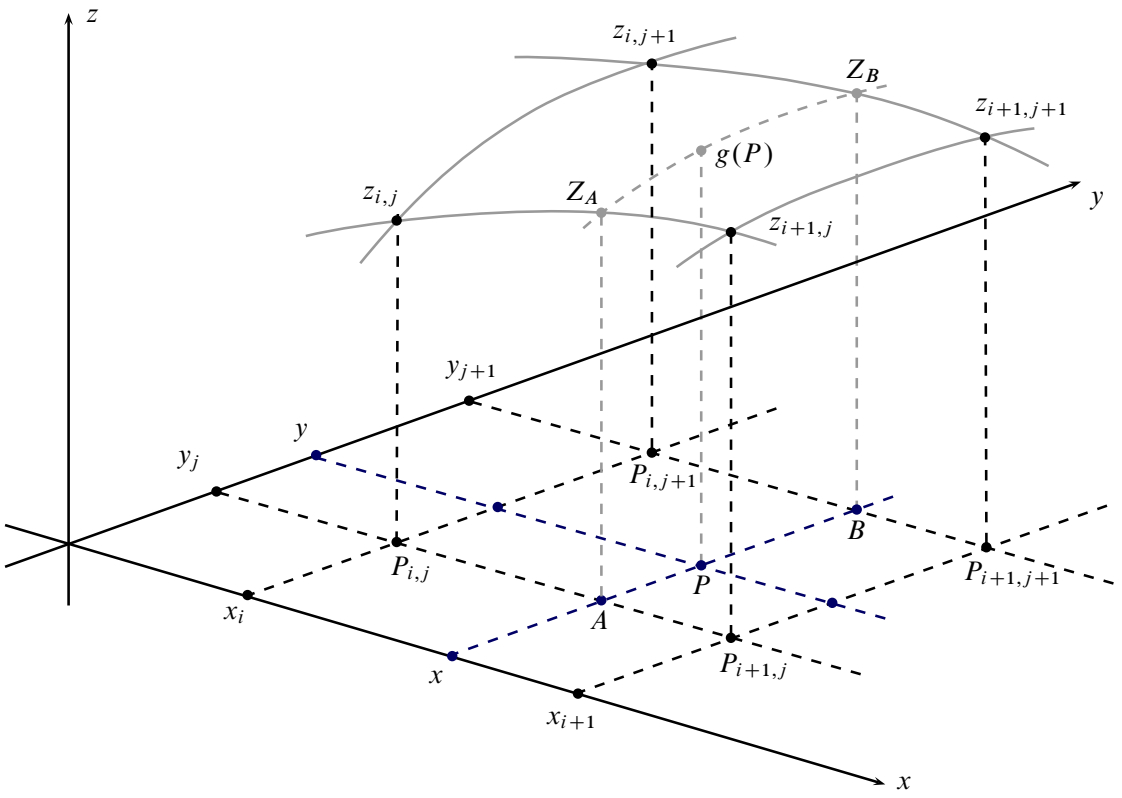

Figura 5.8 Schema di interpolazione in due dimensioni.

mezzo di una sola formula come segue:

$$\begin{aligned} g(x,y) = {} & \frac{z_{i,j}}{(x_{i+1}-x_i)(y_{j+1}-x_j)}(x_{i+1}-x)(y_{j+1}-y) \\ & + \frac{z_{i+1,j}}{(x_{i+1}-x_i)(y_{j+1}-x_j)}(x-x_i)(y_{j+1}-y) \\ & + \frac{z_{i,j+1}}{(x_{i+1}-x_i)(y_{j+1}-x_j)}(x_{i+1}-x)(y-y_j) \\ & + \frac{z_{i+1,j+1}}{(x_{i+1}-x_i)(y_{j+1}-x_j)}(x-x_i)(y-y_j) \end{aligned} \tag{5.15}$$

Si osservi che questo risultato è indipendente dall'ordine in cui vengono effettuate le interpolazioni: se si esegue prima un'interpolazione lungo la direzione y, poi una lungo la direzione x si ottiene ugualmente la formula (5.15).

Contrariamente a quanto lascia intendere il nome, la formula di interpolazione bilineare *non* è lineare. Come si vede esaminando la (5.15), essa è del tipo

$$g(x,y) = c_1 + c_2\,x + c_3\,y + c_4\,xy \tag{5.16}$$

dove $c_1 = z_{i,j}, c_2 = z_{i+1,j} - z_{i,j}, c_3 = z_{i,j+1} - z_{i,j}, c_4 = z_{i,j} - z_{i+1,j} - z_{i,j+1} + z_{i+1,j+1}$. Le costanti da determinare nella formula (5.16) sono quattro proprio quanti sono i punti $P_{i,j}, \ldots, P_{i+1,j+1}$ in cui è valutata la funzione tabellare per ottenere $g(P)$. La funzione interpolante $g(x,y)$ così definita risulta lineare lungo linee parallele agli assi x ed y. Lungo qualsiasi altra linea del piano xy essa è invece una funzione interpolante quadratica.

L'ovvia estensione di questa tecnica di interpolazione a funzioni di tre e, in generale,

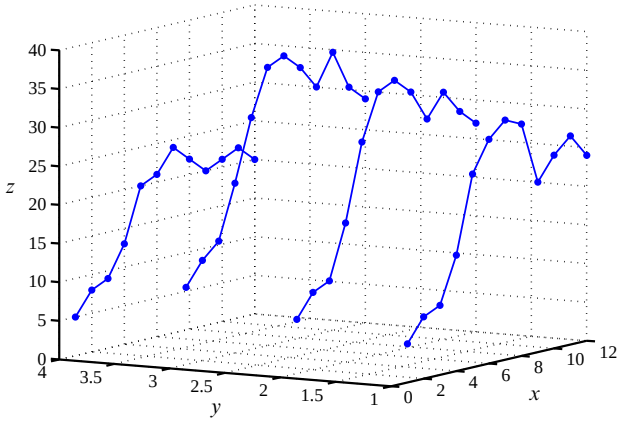

Figura 5.9 Esempio di funzione tabellare di due variabili.

Listato 5.6 Funzione di due variabili nota per punti. Esempio d'uso della funzione `plot3`.

```
% funzione (x,y) --> z
x=1:12;
y=1:4;
z(1,:)=[ 5  8  9 15 25 29 31 30 22 25 27 24];
z(2,:)=[ 7 10 11 18 28 34 35 33 29 32 29 27];
z(3,:)=[10 13 15 22 30 36 37 35 32 36 31 29];
z(4,:)=[ 5  8  9 13 20 21 24 22 20 21 22 20];

plot3(x,y(1)*ones(1,length(x)),z(1,:),'-b',...
      x,y(2)*ones(1,length(x)),z(2,:),'-b',...
      x,y(3)*ones(1,length(x)),z(3,:),'-b',...
      x,y(4)*ones(1,length(x)),z(4,:),'-b')
```

di n variabili viene detta interpolazione *trilineare* ed *n-lineare*.

Per l'interpolazione di funzioni tabellari di due o più variabili Matlab mette a disposizione le funzioni `interp2`, `interp3` ed `interpn`. Esse implementano degli algoritmi cosiddetti di *table lookup* e possono mostrarsi utili nella risoluzione di diversi problemi di dimensionamento preliminare di un velivolo. La figura 5.9 mostra un esempio di funzione di due variabili nota per punti. Essa è stata definita nel listato 5.6 attraverso i vettori `x`, `y` ed attraverso la matrice `z`, di dimensioni 4×12.

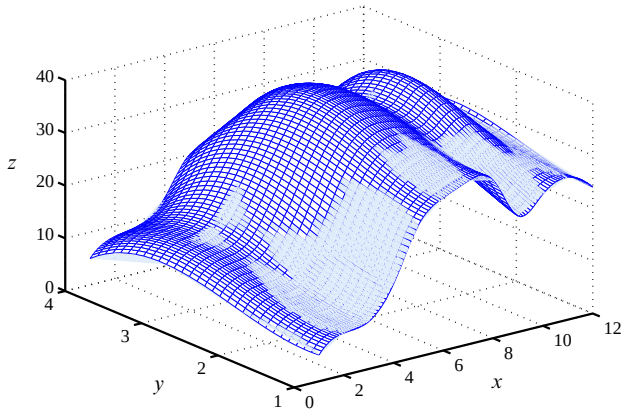

Figura 5.10 Interpolazione di una funzione tabellare di due variabili con *spline* cubiche lungo ciascuna direzione (reticolato). La superficie con campitura uniforme rappresenta l'interpolante bilineare.

Listato 5.7 Interpolazione di una funzione tabellare di due variabili. Esempio d'uso della funzione `interp2`.

```
% funzione (x,y) --> z
x=1:12;
y=1:4;
z(1,:)=[ 5  8  9 15 25 29 31 30 22 25 27 24];
z(2,:)=[ 7 10 11 18 28 34 35 33 29 32 29 27];
z(3,:)=[10 13 15 22 30 36 37 35 32 36 31 29];
z(4,:)=[ 5  8  9 13 20 21 24 22 20 21 22 20];

% intervalli infittiti
xi=1:0.1:12;
yi=1:0.1:4;
[XX,YY]=meshgrid(x,y);
[XI,YI]=meshgrid(xi,yi);

% interpolante
ZI1=interp2(XX,YY,z,XI,YI,'bilinear');
ZI2=interp2(XX,YY,z,XI,YI,'spline');

mesh(XI,YI,ZI1,'EdgeColor','blue'), hold on
mesh(XI,YI,ZI2)
```

L'uso della funzione `interp2` è analogo a quello di `interp1`. Una chiamata del tipo: `interp2(x,y,z,xi,yi)`, fornisce come risultato un valore interpolato in corrispondenza dei valori `xi` ed `yi`. Questi ultimi possono essere anche delle matrici (di uguali dimensioni) ed in tal caso il risultato è una matrice. È possibile specificare il metodo con cui la funzione `interp2` esegue l'interpolazione attraverso una stringa passata come sesto argomento. Tra le possibili scelte vi sono `'bilinear'` e `'spline'`. Per determinare i valori interpolati in una griglia più fitta di quella in cui sono noti i valori tabellari ci si può servire delle

funzioni `griddata` e `meshgrid`. Quest'ultima, la cui utilità è appunto quella di fornire supporto all'interpolazione in più dimensioni e al disegno tridimensionale, è stata usata nel listato 5.7 per produrre il grafico della figura 5.10. Dati `x`, `y` e `z` e definiti i vettori `xi` ed `yi` di valori in cui interpolare, con `meshgrid` sono state generate le matrici `XX`, `YY`, `XI`, `YI`. Le matrici `ZI1` ed `ZI2` di valori interpolati, costruite con due chiamare a `interp2`, sono state passate alla funzione di disegno `mesh`.

Non è raro incontrare la necessità di usare tecniche di interpolazione in più dimensioni. Un esempio molto comune è dato dall'analisi ingegneristica di configurazioni aerodinamiche su cui sono state fatte sperimentazioni in galleria del vento. Spesso, infatti, le caratteristiche aerodinamiche di oggetti di diverse forme ed ingombri sono disponibili in forma di funzioni tabellari di più parametri e, quando utilizzate in calcoli di Dinamica del volo, esse vanno valutate mediante l'uso di algoritmi di interpolazione adeguati. L'esperienza ha mostrato che nella stragrande maggioranza dei casi l'interpolazione bilineare, o n-lineare, è la tecnica da preferirsi. In alcuni casi può essere adatta un'interpolazione con *spline* di cubiche lungo la direzione per la quale si ha il maggior numero di nodi ed una semplice interpolazione lineare lungo le altre direzioni.

5.1.5 Alcune tecniche di *curve fitting*

Come osservato in precedenza una regressione è una procedura che, assegnata una data forma matematica della funzione approssimante a meno di un certo numero di parametri, ha per scopo quello di determinare una particolare $g(x)$ minimizzando la distanza esistente tra i punti della curva approssimante stessa ed i punti campione. L'espressione della $g(x)$ può essere ad esempio una funzione lineare, un polinomio, un polinomio a tratti, una combinazione di funzioni trascendenti, eccetera. Tipicamente, una volta determinata la curva approssimante, essa *non* passa per i punti di supporto. Si vedano ad esempio le figure 5.11 e 5.12.

Più in generale si intende per *curve fitting* il problema di trovare una curva che approssima una serie di dati e risponde possibilmente ad un certo numero di altri vincoli. Spesso non interessa principalmente l'estrazione o l'interpretazione dei parametri che definiscono la curva approssimante. Si desidera invece disegnare semplicemente una curva più regolare possibile che passi accettabilmente vicino ai punti di supporto. In questi casi si parla di *regressione non parametrica* (*nonparametric fitting*). In Matlab esiste un *toolbox* dedicato alla interpolazione approssimata che supporta in particolare il *fitting* non parametrico. Di particolare interesse è il concetto di *smoothing spline* che non è altro che una *spline* di cubiche che approssima i dati in maniera più o meno esatta a seconda delle caratteristiche richieste dall'utente.

Una *smoothing spline* $g_{ss}(x)$ è costruita in modo tale da minimizzare la quantità:

$$J = p \sum_{j=1}^{n} w_j \left| y_j - g_{ss}(x_j) \right|^2 + (1-p) \int_{x_1}^{x_n} \lambda(t)\, g''_{ss}(t)\, \mathrm{d}t \tag{5.17}$$

L'utente potrà assegnare i coefficienti di peso w_j in maniera opportuna se desidera una curva che passi più o meno vicino a determinati punti altrimenti questi sono posti per *default* tutti pari ad 1. D'altra parte, è possibile impostare l'importanza relativa del secondo addendo nella formula (5.17), detto *misura di rugosità* della curva (*roughness measure*),

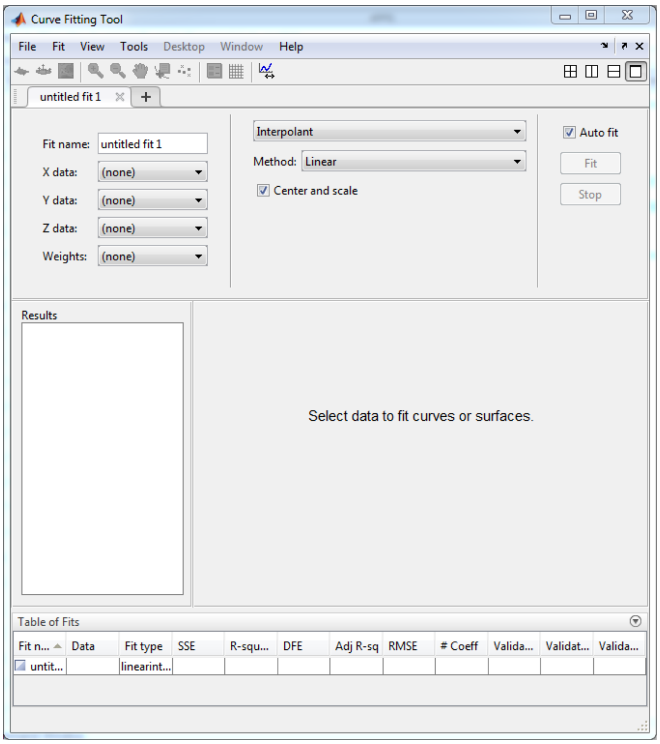

Figura 5.11 Finestra di configurazione del tool *Curve Fitting*. Può essere aperta dal menu APPS di Matlab o con il comando `cftool`.

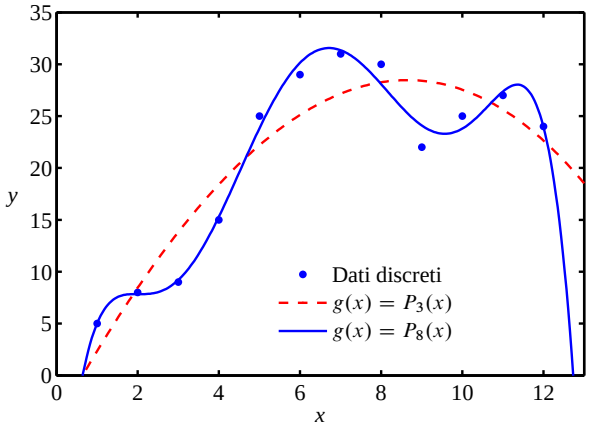

Figura 5.12 Esempi di interpolazione approssimata con polinomi di terzo ed ottavo grado.

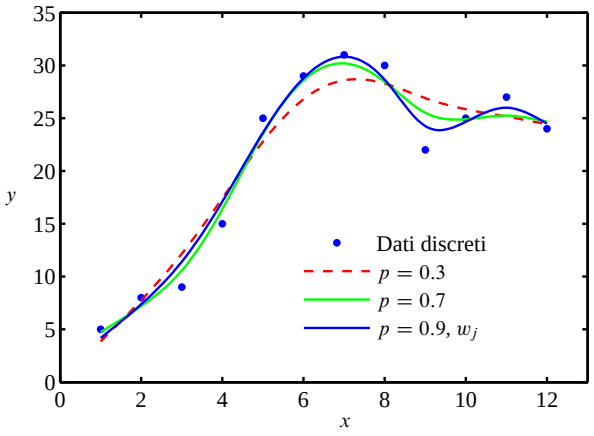

Figura 5.13 Esempi di interpolazione approssimata con *spline smoothing*.

Listato 5.8 Esempio d'uso delle funzioni `csaps` e `fnval`.

```
% n coppie (x,y)
x = 1:12; % ← definisce le x_j, con n = 12
y=[5 8 9 15 25 29 31 30 22 25 27 24]; % ← le y_j

% intervallo delle x infittito
xi=1:0.1:12;

pp1 = csaps(x,y, .3); % ← p = 0.3
pp2 = csaps(x,y, .7); % ← p = 0.7
pp3 = csaps(x,y, .9, ...                % ← p = 0.9
   [], ...
 [0.1*ones(1,6) 1 1 1 1 1 1]); % ← w_j variabili

y4 = fnval(xi,pp1);
y5 = fnval(xi,pp2);
y6 = fnval(xi,pp3);

plot(x,y,'o',xi,y4,'--r',xi,y5,'-g',xi,y6,'-b')
```

per mezzo del parametro p. Questa quantità è detta *smoothing parameter* e permette all'utente di regolare il comportamento della curva finale giungendo ad un compromesso tra due esigenze contrastanti in presenza di dati affetti da rumore: da una parte quella di avere una g_{ss} più regolare possibile, dall'altra quella di avere una curva passante quanto più vicino possibile ai dati. Come si vede dall'esame attento della (5.17), per $p = 1$ si otterrà una *spline* di cubiche che interpola più o meno esattamente i punti di supporto. Per p che decresce, via via fino ad avvicinarsi a zero si avrà un'interpolante approssimata che tende ad una *retta di regressione ai minimi quadrati*. Esempi di interpolazione approssimanta con *smoothing spline* sono riportati nella figura 5.13 e nel listato 5.8.

5.2 Integrazione numerica

Data una funzione f di una variabile, continua in un intervallo $[a, b]$, l'integrazione definita di f, estesa a tale intervallo, consiste nel determinare il numero

$$I(f) = \int_a^b f(x)\,\mathrm{d}x \tag{5.18}$$

Dall'Analisi matematica è noto che l'integrale (5.18) è il risultato di un processo di limite. Ad esempio, per la funzione riportata nella figura 5.14, esso si può interpretare come la somma delle aree dei rettangoli che sottendono il grafico di $f(x)$ al tendere ad infinito del numero delle suddivisioni dell'intervallo di integrazione.

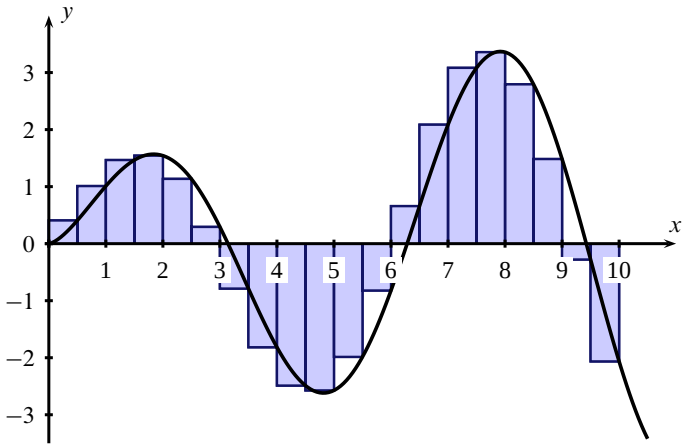

Figura 5.14 Quadratura della funzione $f(x) = 1.2\sqrt{x}\sin(x)$ su un intervallo $[0, 10]$.

Nel Calcolo numerico vengono studiate le cosiddette *formule di quadratura*, che permettono di ottenere un valore approssimato $\tilde{I}$ dell'integrale (5.18) a partire da un numero n di valori di f. Una formula di quadratura ha l'espressione generale seguente

$$\tilde{I}(f; a, b) = \sum_{j=1}^{n} w_j\, f(x_j) \tag{5.19}$$

dove, al solito, le x_j sono i *nodi* ed i coefficienti w_j sono i cosiddetti *pesi*. Si assuma inizialmente, per semplicità, che i nodi siano equispaziati, cioè che le differenze $x_{j+1} - x_j = h = (b-a)/(n-1)$.

Una prima formula di quadratura notevole è la cosiddetta *formula del punto medio*:

$$\tilde{I} = h \sum_{j=2}^{n} f\left(\frac{x_{j-1} + x_j}{2}\right) \tag{5.20}$$

in cui il valore approssimato dell'integrale è dato dalla somma dei rettangoli di base h ed altezza, rispettivamente, i valori della funzione valutata nei punti medi dei singoli sotto-intervalli in cui è suddiviso $[a, b]$. Si può far vedere che con la (5.20) si commette un errore $E = I(f) - \tilde{I}$ proporzionale ad h^2.

Una seconda formula di quadratura, molto nota, è la *formula dei trapezi*:

$$\tilde{I} = h\left(\frac{f(a)}{2} + \sum_{j=2}^{n-1} f(x_j) + \frac{f(b)}{2}\right) \tag{5.21}$$

per la quale, ancora, l'errore di approssimazione è proporzionale ad h^2. Questa formula prende il nome dal fatto che $\tilde{I}$ è ottenuto sommando i trapezi rettangoli aventi per basi $f(x_j)$ ed $f(x_{j-1})$ ed altezza h. Sviluppando la sommatoria per $j = 2, \ldots, n$ si ottiene la formula (5.21). Essa è semplice da ricordare in quanto è pari, a meno del fattore moltiplicativo h, alla somma dei valori che la f assume nei nodi interni e della metà dei valori assunti nei nodi estremi.

Infine, una terza formula di quadratura, più accurata delle due precedenti, è la *formula (composita) di Cavalieri-Simpson*:

$$\tilde{I} = \frac{h}{6}\left(f(a) + 2\sum_{j=2}^{n-1} f(x_j) + 4\sum_{j=2}^{n} f\left(\frac{x_{j-1} + x_j}{2}\right) + f(b)\right) \tag{5.22}$$

il cui errore di approssimazione è proporzionale ad h^4. La (5.22) può essere vista come una combinazione delle due precedenti attraverso opportuni coefficienti.

Esistono altre formule di quadratura, come ad esempio le formule di Gauss e di Newton-Cotes, ma quelle appena richiamate sono sufficienti a dare un'idea al lettore del problema numerico che è necessario risolvere per conoscere l'integrale definito di una funzione.

Nelle applicazioni si potranno verificare situazioni differenti, a seconda del particolare problema in esame, che rendono più o meno complicato il calcolo di un integrale definito. Ad esempio ci saranno casi in cui la f è una funzione analitica, facilmente calcolabile nell'intervallo assegnato. In altri casi la f sarà una funzione tabellare, dunque nota per punti. In quest'ultima circostanza i nodi potranno non essere necessariamente equispaziati e bisognerà utilizzare delle formule di quadratura che generalizzano quelle appena viste al caso di suddivisione non uniforme dell'intervallo di integrazione. Queste ultime sono spesso implementate in concomitanza di strategie di adattamento del passo a seconda della rapidità di variazione della funzione integranda, al fine di ridurre l'errore di approssimazione.

In Matlab esiste la funzione `integral` con la quale l'utente può applicare una formula di quadratura di Cavalieri-Simpson con passo variabile. Un esempio d'uso di questa funzione è riportato nei listati 5.9 e 5.10. In quest'ultimo, in particolare, è stata definita una funzione **`myfun`**. La definizione è stata opportunamente salvata nella cartella di lavoro, nel file `myfun.m`. Le variabili `x` ed `y` vengono assegnate nello *script* chiamante, listato 5.9, e dichiarate sia in quest'ultimo che nella funzione definita dall'utente come variabili globali, mediante il comando `global`. La funzione **`myfun`** accetta dunque un solo parametro e restituisce il valore interpolato tramite una *spline* di cubiche, assegnata la funzione tabellare che lega `x` ad `y`.

Dal momento che l'algoritmo implementato da una *function* di Matlab come `integral` è indipendente dalla particolare funzione matematica alla quale esso si applica, la sintassi di `integral` prevede il passaggio come primo argomento di una *function*, in particolare di un *puntatore a function*. I rimanenti due argomenti sono i due estremi di integrazione ed

Listato 5.9 Esempio d'uso della funzione `integral` e della funzione **`myfun`** definita dall'utente.

```
global x y
% n coppie (x,y)
x = 1:12; % ← definisce le x_j, con n = 12
y=[5 8 9 15 25 29 31 30 22 25 27 24]; % ← le y_j
% intervallo delle x infittito
xi=1:0.1:12;
y3=interp1(x,y,xi,'spline');

% definisce la funzione Ĩ(g_spline;0,x)
for j=1:(length(xi)-1)
    si(j) = integral(@myfun,xi(1),xi(j+1));
end
plot(x,y,'o',xi,y3,'-', ...
     xi(2:length(xi)),0.08*si,'-')
```

Listato 5.10 Funzione **`myfun`** il cui puntatore è passato alla funzione `integral`.

```
function r = myfun(x0)
global x y
r = interp1(x,y,x0,'spline');
```

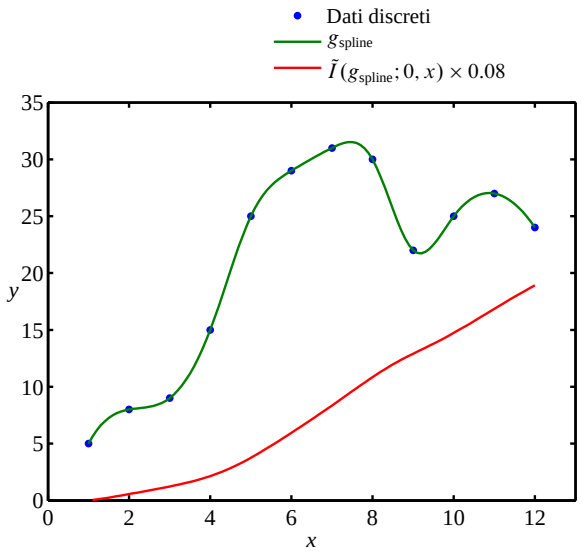

Figura 5.15 Esempio di integrazione definita con formula di Cavalieri-Simpson.

il valore restituito sarà l'integrale approssimato. Il puntatore alla funzione viene passato tramite la sintassi `@myfun`, cioè anteponendo il carattere '`@`' al nome della *function*. La figura 5.15 mostra il risultato dell'esempio in questione. Lo *script* definisce una funzione

$$\tilde{I}(g_{\text{spline}};0,x)=\int_0^x g_{\text{spline}}(t)\,\mathrm{d}t \tag{5.23}$$

dove g_{spline} è la funzione approssimante definita in **`myfun`**. Per esigenze di rappresentazione delle due curve nella figura 5.15, le ordinate del diagramma della (5.23) sono opportunamente scalate.

L'esempio appena discusso mostra uno dei possibili metodi pratici di calcolo di un integrale definito di una funzione nota per punti. Si lascia per esercizio al lettore quello di verificare come cambiano i valori dell'integrale se nella chiamata ad `interp1` vengono passate le opzioni `'linear'` o `'pchip'` al posto di `'spline'`.

Naturalmente, una funzione come `integral` può essere utilizzata per calcolare integrali definiti di funzioni in genere. Ad esempio, l'integrale

$$\int_0^{\pi}\sin x\,\mathrm{d}x=-\cos x\Big|_0^{\pi}=-\cos\pi-(-\cos 0)=2$$

il cui valore è noto esattamente poiché della funzione integranda si può svolgere analiticamente l'integrale indefinito, può essere calcolato numericamente come segue:

```
>> format long
>> integral(@sin,0,pi)
ans =
    1.99999999639843
```

Si lascia per esercizio al lettore il calcolo dell'integrale

$$\int_0^1 \frac{1}{x^2 - x + 1}\,\mathrm{d}x = \frac{2\sqrt{3}}{3}\arctan\left(\frac{\sqrt{3}}{3}(2x-1)\right)\Bigg|_0^1 = 1.209199576$$

per il quale si suggerisce di definire una funzione `myIntFun` come segue:

```
function y = myIntFun(x)
y = 1./(x.^2-x+1); % notare l'uso di .^
```

e di passarne il puntatore `@myIntFun` alla funzione `integral` insieme agli estremi di integrazione.

Si osserva infine che Matlab mette a disposizione anche la funzione `trapz`. Ad esempio, una chiamata del tipo `trapz(x,y)` restituisce l'integrale della funzione tabellare che associa gli elementi dei due vettori `x` ed `y`, esteso all'intervallo di estremi `x(1)` ed `x(length(x))`. Per approfondimenti su questa funzione si rimanda all'help di Matlab.

5.3 Derivazione numerica

Spesso nelle applicazioni si ha a disposizione una funzione nota per punti e si ha la necessità, allo stesso tempo, di dover considerare le derivate di una ipotetica funzione $f(x)$ che assuma valori y_j in corrispondenza dei nodi x_j.

Per il calcolo di tali derivate ci si potrebbe ricondurre naturalmente a quanto richiamato a proposito dell'interpolazione esatta al paragrafo 5.1. Se si assume un certo modello di rappresentazione della $f(x)$, corrispondente alla funzione analitica interpolante, sarà possibile calcolare agevolmente le derivate nei punti desiderati. Ovviamente, questo tipo di approccio va seguito con una certa attenzione ed il modello va scelto in base all'ordine delle derivate richieste. Ad esempio, se si desidera calcolare la derivata prima e seconda in corrispondenza dei nodi x_j, allora il modello di funzione interpolante dovrà essere tale da garantirne l'esistenza: dunque una *spline* di cubiche come funzione interpolante è certamente la scelta più indicata; meno indicata è la scelta di una interpolante cubica a tratti di Hermite poiché essa garantisce solo la continuità delle derivate prime. Va osservato che spesso nei problemi di dimensionamento in cui è necessario calcolare delle derivate, interessano solo le derivate prime.

Spesso, a partire da un assegnato insieme $\{(x_j, y_j)\}_{j=1,\ldots,n}$, ovvero da una funzione tabellare f_{tab}, interessa semplicemente costruire un'altra funzione tabellare f'_{tab} che ai medesimi nodi x_j faccia corrispondere delle y'_j, cioè i valori approssimati della derivata $f'(x)$ di una ipotetica funzione interpolante esatta $f(x)$. Tale azione, in un gergo un po' grossolano ma ingegneristicamente accettabile, viene detta a volte *derivata numerica* della tabella f_{tab}. La derivata numerica viene calcolata tipicamente attraverso le cosiddette *formule alle differenze finite* che si studiano nel Calcolo numerico.

Il metodo delle differenze finite nasce come metodo numerico di risoluzione di equazioni differenziali. Quando è applicato alla derivazione numerica di una funzione nota per punti esso consiste nell'approssimare il generico valore $f'(x_j)$ — per conoscere il quale sarebbe necessario poter valutare la f con continuità in un intorno di x_j, quindi in un numero infinito di punti — con una formula algebrica che richiede di valutare la f in un numero *finito* di punti. Si passa cioè dall'operazione di limite di un rapporto incrementale,

come vorrebbe la definizione matematica di derivata di una funzione continua, a quella di semplice rapporto incrementale. Di qui il termine "differenze finite". Le formule alle differenze finite, applicate in particolare ad una funzione tabellare, richiederanno di valutare la f nel nodo considerato e nei nodi adiacenti. Pertanto nelle formule algebriche compariranno i valori tabellari y_j, y_{j-1}, y_{j+1}, eccetera. L'insieme di nodi per i quali prelevare i valori della tabella in una data formula alle differenze finite viene detto *stencil* o *schema di discretizzazione*.

In questi richiami non si scenderà nei dettagli teorici legati alle formule alle differenze finite, per i quali si rimanda ai testi di analisi numerica, ad esempio [74, 75, 76]. Basta ricordare qui che esse si ricavano, secondo un approccio classico, sviluppando la funzione $f(x)$ in serie di Taylor in un intorno del punto x_j e troncando opportunamente tale sviluppo. Nello studio e nel tentativo di rendere piccolo l'errore di approssimazione $E = y'_j - f'(x_j)$ vengono anche costruite delle formule di ordine superiore combinando linearmente un certo numero di formule di ordine inferiore relative a diversi intorni del nodo considerato.

Un risultato classico che fornisce la derivata prima in un generico nodo x_j è il seguente:

$$y'_j = \frac{y_{j+1} - y_{j-1}}{2h} \tag{5.24}$$

Lo schema della formula (5.24) è valido per nodi uniformemente distribuiti nell'intervallo $[x_1, x_n]$, con $h = (x_1 - x_n)/(n-1)$, ed è noto come *formula centrale* del secondo ordine. Infatti si può far vedere che l'errore di approssimazione è proporzionale ad h^2. Per nodi non uniformemente distribuiti è possibile far vedere che la formula centrale del secondo ordine analoga alla (5.24) è la seguente:

$$y'_j = \frac{1}{(x_{j+1} - x_j) + (x_j - x_{j-1})} \left[\frac{x_j - x_{j-1}}{x_{j+1} - x_j} (y_{j+1} - y_j) + \frac{x_{j+1} - x_j}{x_j - x_{j-1}} (y_j - y_{j-1}) \right] \tag{5.25}$$

Un'altra formula molto nota, anch'essa del secondo ordine, è quella che fornisce il valore approssimato della derivata seconda:

$$y''_j = \frac{y_{j+1} - 2y_j + y_{j-1}}{h^2} \tag{5.26}$$

Si osservi che le formule precedenti sono ovviamente valide solo per nodi interni all'intervallo $[x_1, x_n]$, ovvero per $j = 2, \ldots, n-1$. Per calcolare le derivate agli estremi dell'intervallo è necessario utilizzare degli schemi detti *one-sided* o anche formule alle differenze "in avanti" (*forward*) e formule alle differenze "all'indietro" (*backward*). Una formula *forward* del primo ordine, con errore proporzionale ad h, è la seguente:

$$y'_j = \frac{y_{j+1} - y_j}{h} \tag{5.27}$$

e va utilizzata per $j = 1$ per conoscere il valore y'_1. L'analoga formula del secondo ordine è data invece dalla:

$$y'_j = \frac{-3y_j + 4y_{j+1} - 2y_{j+2}}{2h} \tag{5.28}$$

Come si può notare dalle formule richiamate fino a qui, le formule di ordine superiore hanno sempre uno *stencil* più esteso rispetto alle analoghe formule di ordine più basso.

Le formule *backward*, rispettivamente, del primo e del secondo ordine sono date dalla

$$y'_j = \frac{y_j - y_{j-1}}{h} \tag{5.29}$$

e dalla

$$y'_j = \frac{y_{j-2} - 4y_{j-1} + 3y_j}{2h} \tag{5.30}$$

e vanno utilizzate per $j = n$ per conoscere y'_n.

Si tralascia di riportare a questo punto le formule analoghe alle (5.27)-(5.30) per nodi non uniformemente distribuiti, dal momento che ciò esulerebbe dallo scopo di questi richiami.

Per gli utenti di Matlab il metodo più snello ed efficiente per effettuare derivazioni numeriche di tabelle è quello di basarsi sulla funzione nativa `gradient`. Quando la chiamata a questa *function* è effettuata passando due vettori riga o colonna di uguali dimensioni, ad esempio `y` ed `x`, `gradient(y,x)` restituisce la derivata numerica della tabella `y` con derivate calcolate in corrispondenza dei nodi `x`. Le formule alle differenze centrali per i nodi interni e *one-sided* nei nodi estremi implementate in `gradient` sono tali da garantire un grado accuratezza globale del secondo ordine. Un esempio d'uso di tale funzione costruito intorno alla solita funzione definita per punti degli esempi precedenti è dato dalla figura 5.16, ottenuta con i comandi implementati nel listato 5.11.

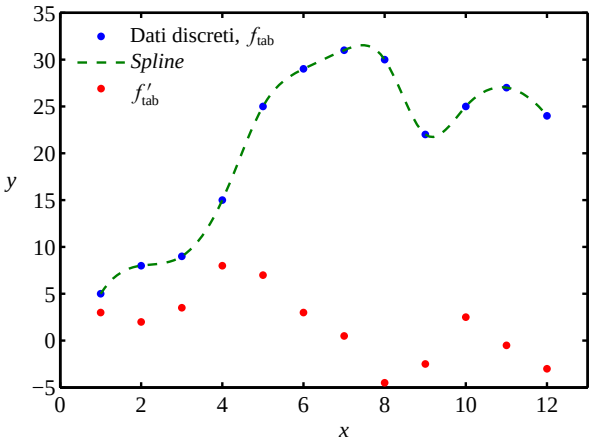

Figura 5.16 Esempio di derivazione numerica di una funzione tabellare f_{tab}. Il risultato è una tabella f'_{tab} di uguali dimensioni.

Listato 5.11 Esempio d'uso della funzione `gradient`.

```
% n coppie (x,y)
x = 1:12; % ← definisce le x_j, con n = 12
y=[5 8 9 15 25 29 31 30 22 25 27 24]; % ← le y_j

% intervallo delle x infittito
xi=1:0.1:12;

y3=interp1(x,y,xi,'spline');

% derivata numerica
dy1 = gradient(y,x);

plot(x,y,'o',xi,y3,'-',x,dy1)
```

Nella figura 5.16 si è riportata la derivata numerica come funzione tabellare, dunque come insieme di pallini così come andrebbe fatto per i dati originali. Un diagramma che interpolasse i punti (x_j, y'_j) riporterebbe un'operazione in più, che sarebbe probabilmente fuorviante. Se si volessero unire i punti in questione si dovrebbe derivare analiticamente e riportare in grafico una funzione interpolante polinomiale a tratti di grado opportuno, cioè dipendente dal grado di accuratezza delle formule alle differenze finite che hanno permesso di determinare la f'_{tab}.

Si osservi che esiste la possibilità di programmare a basso livello delle funzioni che implementano le diverse formule alle differenze per mezzo della funzione nativa `diff`.

In questi casi, soprattutto per ragioni di efficienza computazionale e per incoraggiare il riuso di codice già consolidato, si incoraggia l'uso di `gradient`. Per approfondimenti è bene comunque consultare attentamente l'*help* ed i riferimenti incrociati citati al termine di ciascuna voce.

5.4 Soluzione numerica di sistemi di equazioni differenziali ordinarie

5.4.1 Problemi di valori iniziali e integrazione numerica

Sia data una funzione $\boldsymbol{f}$ delle variabili scalari $t, y_1, \ldots, y_m$. In generale sia la $\boldsymbol{f}$ una funzione vettoriale ad m componenti: $\boldsymbol{f} = [f_1, \ldots, f_m]^{\mathrm{T}}$. Un sistema di equazioni differenziali ordinarie del primo ordine, nell'incognita vettoriale $\boldsymbol{y} = [y_1, \ldots, y_m]^{\mathrm{T}} = \boldsymbol{y}(t)$, funzione della variabile indipendente t, si scrive nella forma

$$\boldsymbol{y}' = \boldsymbol{f}(t, \boldsymbol{y}) \tag{5.31}$$

dove l'operatore $(\cdot)' \equiv \mathrm{d}(\cdot)/\mathrm{d}t$ indica la derivazione rispetto alla t.

È noto dall'Analisi matematica che una qualsiasi equazione differenziale ordinaria, di qualsiasi ordine, può sempre essere ridotta ad un sistema di equazioni differenziali del primo ordine del tipo (5.31). Ad esempio l'equazione

$$\frac{\mathrm{d}^2 g}{\mathrm{d}t^2} + q(t)\frac{\mathrm{d}g}{\mathrm{d}t} = r(t) \tag{5.32}$$

nell'incognita scalare $g(t)$, per assegnate funzioni $q(t)$ ed $r(t)$, equivale al sistema

$$\begin{cases} \dfrac{\mathrm{d}y_1}{\mathrm{d}t} = y_2(t) \\ \dfrac{\mathrm{d}y_2}{\mathrm{d}t} = r(t) - q(t)\, y_2(t) \end{cases} \tag{5.33}$$

nella funzione vettoriale incognita $\boldsymbol{y}(t) = [y_1(t), y_2(t)]^{\mathrm{T}}$. L'incognita scalare originaria compare nella posizione $y_1(t) = g(t)$ mentre $y_2(t)$ è una nuova variabile dipendente, spesso chiamata 'variabile ausiliaria'.

Il generico problema retto da equazioni differenziali ordinarie di ordine superiore viene quindi sempre ricondotto allo studio di un sistema di m equazioni accoppiate del primo ordine nelle funzioni incognite y_i, $i = 1, \ldots, m$, avente la forma

$$\frac{\mathrm{d}y_i}{\mathrm{d}t} = f_i(t, y_1, \ldots, y_m) \qquad i = 1, \ldots, m \tag{5.34}$$

dove le funzioni f_i a secondo membro sono assegnate. Il sistema (5.34) insieme con la condizione iniziale nel punto t_0

$$\boldsymbol{y}(t_0) = \boldsymbol{y}_0 \tag{5.35}$$

costituisce un *problema di valori iniziali* che ammette una soluzione unica, sotto opportune ipotesi di regolarità della funzione $\boldsymbol{f}$. Un problema ben posto è anche quello costituito

dalla (5.34) e dalle condizioni al contorno:

$$\begin{aligned} y_i(a) &= y_{i,A} \qquad i = 1, \dots m' \\ y_i(b) &= y_{i,B} \qquad i = m'+1, \dots m \end{aligned} \tag{5.36}$$

dove, ad esempio, alcune condizioni ($m' < m$) sono assegnate per un valore di $t = a$ e le rimanenti per un valore di $t = b$. In tal caso si parla di *problema di valori al contorno* nell'intervallo delle $t \in [a, b]$. In questi richiami si parlerà esclusivamente della risoluzione di un problema di valori iniziali.

I metodi numerici di risoluzione del problema (5.34)-(5.35), detti anche metodi di integrazione 'al passo', determinano in maniera approssimata i valori della $\boldsymbol{y}$ in corrispondenza di un numero discreto di valori della t

$$t_0 \, , \, t_1 \, , \, \dots \, , \, t_n \, , \, \dots \, , \, t_N \tag{5.37}$$

dove n rappresenta l'indice contatore ed $h_n = t_{n+1} - t_n$ il generico passo di integrazione. Detta $\tilde{\boldsymbol{y}}(t)$ la soluzione esatta del problema di valori iniziali, si dirà

$$\boldsymbol{y}_0 \, , \, \boldsymbol{y}_1 \, , \, \dots \, , \, \boldsymbol{y}_n \, , \, \dots \, , \, \boldsymbol{y}_N \tag{5.38}$$

la sequenza di valori calcolati numericamente che approssima quella dei valori esatti

$$\tilde{\boldsymbol{y}}(t_0) \, , \, \tilde{\boldsymbol{y}}(t_1) \, , \, \dots \, , \, \tilde{\boldsymbol{y}}(t_n) \, , \, \dots \, , \, \tilde{\boldsymbol{y}}(t_N) \tag{5.39}$$

Per semplicità si supporrà qui che il passo di integrazione sia costante e pari ad $h = \left(t_{\text{Fin}} - t_{\text{Iniz}}\right)/N$, con N il numero totale di passi di integrazione.

Qualsiasi procedura numerica di integrazione al passo segue la stessa semplice idea di base: viste le formule (5.34), i differenziali $\mathrm{d}y_i$ e $\mathrm{d}t$ vengono riscritti come incrementi finiti Δy_i e Δt; moltiplicando ciascuno dei secondi membri delle (5.34) per Δt si ottengono delle formule algebriche che, 'passo passo', cioè di Δt in Δt a partire da $\boldsymbol{y}(t_0)$, forniscono la variazione $\Delta \boldsymbol{y}$ della variabile dipendente in termini delle funzioni f_i valutate via via nei vari punti $(t_n, \boldsymbol{y}_n)$. Nel limite per $\Delta t \to 0$ la soluzione numerica, campionata in un numero sempre maggiore di punti, tenderà alla soluzione esatta del problema differenziale di partenza. L'implementazione di una simile procedura corrisponde al ben noto *metodo* di integrazione *di Eulero*.

5.4.2 Schemi di integrazione multistadio

La formula di integrazione corrispondente al metodo di Eulero, che fa 'avanzare' una soluzione, nota al generico passo n, dal punto t_n a quello successivo $t_{n+1} \equiv t_n + h$ è la seguente:

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + \Delta \boldsymbol{y}_n \stackrel{\text{def}}{=} \boldsymbol{y}_n + h \, \boldsymbol{f}\left(t_n, y_n\right) \tag{5.40}$$

mutuata dalla ben nota formula di Taylor

$$\tilde{\boldsymbol{y}}(t_{n+1}) = \tilde{\boldsymbol{y}}(t_n) + \left.\frac{\mathrm{d}\tilde{\boldsymbol{y}}}{\mathrm{d}t}\right|_{t_n, \tilde{\boldsymbol{y}}(t_n)} h + O(h^2) \tag{5.41}$$

in cui sono troncati i termini infinitesimi di ordine superiore al primo.

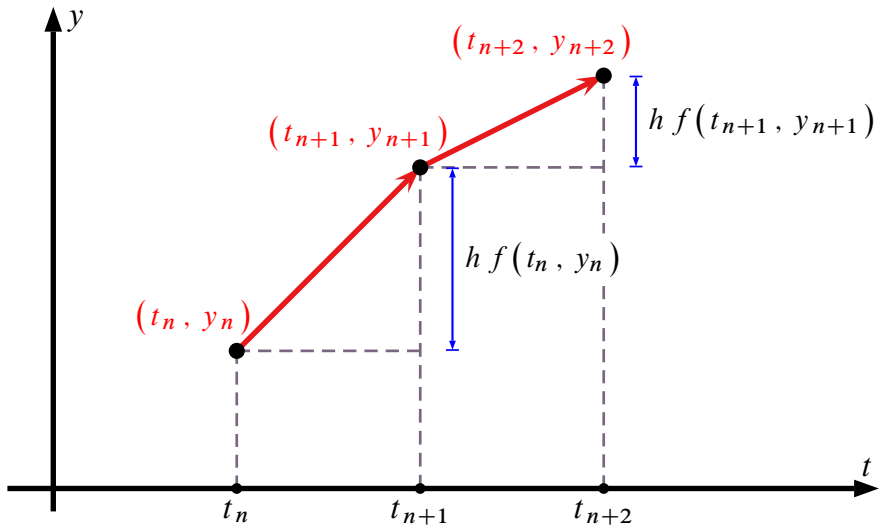

Figura 5.17 Metodo di Eulero. Con la derivata f valutata una volta nel punto iniziale di ogni intervallo $[t_n, t_{n+1}]$ si estrapola direttamente il valore successivo della funzione incognita in corrispondenza dell'altro estremo.

La (5.40) è una tipica formula 'non simmetrica', nel senso che sfrutta delle informazioni, cioè il valore $\boldsymbol{y}_n$ e la derivata $\boldsymbol{f}(t_n, y_n)$, valutate solo al primo estremo dell'intervallo $[t_n, t_{n+1}]$. Ciò rende la 'predizione' del valore $\boldsymbol{y}_{n+1}$ accurata a meno di un termine di correzione di $O(h^2)$. Per convenzione un metodo di integrazione al passo, oppure una formula di integrazione, basati sul calcolo degli incrementi $\Delta \boldsymbol{y}_n$ si dicono di ordine r se l'*errore di troncamento* $\boldsymbol{E}$ presente nella formula di avanzamento generale

$$\tilde{\boldsymbol{y}}(t_{n+1}) = \tilde{\boldsymbol{y}}(t_n) + \Delta \boldsymbol{y}_n + \boldsymbol{E} \tag{5.42}$$

è di $O(h^{r+1})$. Nel caso della (5.40) si ha $\Delta \boldsymbol{y}_n = h\,\boldsymbol{f}(t_n, y_n)$ ed $E = O(h^2)$. Infatti la formula di Eulero (5.40) è nota come formula di integrazione al passo del primo ordine.

Il metodo di Eulero *non* è raccomandabile per un uso pratico, per due motivi principali: (*i*) esso non è accurato, a parità di passo di integrazione, quanto altri metodi di altrettanto semplice implementazione, come il metodo di Runge-Kutta di cui si parlerà qui di seguito; (*ii*) il metodo di Eulero può risultare spesso instabile quando alcune componenti della funzione $\boldsymbol{f}$ sono sensibilmente variabili.

Ora si consideri comunque una formula del tipo (5.40), e si definisca un incremento 'di prova' della $\boldsymbol{y}$

$$(\Delta \boldsymbol{y})_0 = h\boldsymbol{f}(t_n, \boldsymbol{y}_n) \stackrel{\text{def}}{=} h\boldsymbol{f}_0$$

calcolato a partire dai valori noti al passo corrente. Si otterrà un nuovo possibile valore della $\boldsymbol{y}$ pari a

$$\boldsymbol{y}_n + (\Delta \boldsymbol{y})_0 = \boldsymbol{y}_n + h\boldsymbol{f}_0$$

associabile a t_{n+1}, attraverso la valutazione della funzione $\boldsymbol{f}$ in corrispondenza di t_n ed $\boldsymbol{y}_n$. Si veda a tal proposito la figura 5.18. A questo punto si reiteri l'applicazione della formula precedente, valutando stavolta la $\boldsymbol{f}$ in corrispondenza dei valori intermedi $t_n + \frac{1}{2}h$ ed $\boldsymbol{y}_n + \frac{1}{2}(\Delta \boldsymbol{y})_0$, di cui il secondo dipende dal risultato della prima applicazione della

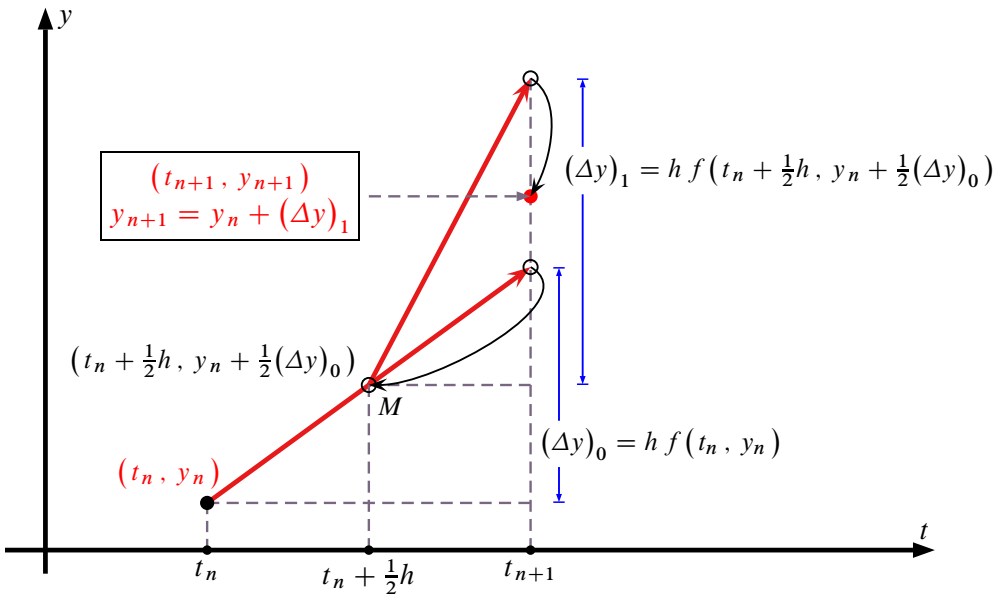

Figura 5.18 Formula di Runge-Kutta del secondo ordine o *Midpoint method*. Con la derivata iniziale si ottiene un punto intermedio M. In esso si valuta la derivata f e l'incremento $(\Delta y)_1$ per l'avanzamento al passo successivo a partire da quello corrente $(t_n\,,\,y_n)$.

formula stessa. Si otterrà un nuovo incremento

$$(\Delta \boldsymbol{y})_1 = h\,\boldsymbol{f}\left(t_n + \frac{1}{2}h\,,\,\boldsymbol{y}_n + \frac{1}{2}(\Delta \boldsymbol{y})_0\right) \stackrel{\text{def}}{=} h\boldsymbol{f}_1$$

che potrà essere considerato un valore più accurato di $h\boldsymbol{f}_0$. Si può far vedere [74] che $(\Delta \boldsymbol{y})_1$ approssima l'incremento $\tilde{\boldsymbol{y}}(t_{n+1}) - \tilde{\boldsymbol{y}}(t_n)$ a meno di un termine di $O(h^3)$. La procedura appena descritta, che si riassume nella forma

$$\begin{aligned}
(\Delta \boldsymbol{y})_0 &= h\,\boldsymbol{f}(t_n\,,\,\boldsymbol{y}_n) \\
(\Delta \boldsymbol{y})_1 &= h\,\boldsymbol{f}\left(t_n + \frac{1}{2}h\,,\,\boldsymbol{y}_n + \frac{1}{2}(\Delta \boldsymbol{y})_0\right) \\
\boldsymbol{y}_{n+1} &= \boldsymbol{y}_n + (\Delta \boldsymbol{y})_1
\end{aligned} \tag{5.43}$$

corrisponde ad un metodo di integrazione del secondo ordine ed è nota come *midpoint method* o *metodo di Runge-Kutta del secondo ordine*. Si veda l'interpretazione grafica della figura 5.18. Le (5.43) mostrano che, combinando due formule del primo ordine, cioè costruendo un valore definitivo della $\boldsymbol{y}_{n+1}$ in due stadi come somma

$$\boldsymbol{y}_{n+1} = \boldsymbol{y}_n + 0 \cdot (\Delta \boldsymbol{y})_0 + 1 \cdot (\Delta \boldsymbol{y})_1 \tag{5.44}$$

e valutando differentemente la funzione $\boldsymbol{f}$ in ciascuno stadio, si ottiene la cancellazione dal termine di errore del contributo di $O(h^2)$.

Esistono peraltro diversi modi di valutare la $\boldsymbol{f}(t, \boldsymbol{y})$ un numero $N_S + 1$ di volte all'interno dell'intervallo $[t_n, t_{n+1}]$ secondo formule del primo ordine come la (5.40) e di combinarne opportunamente il risultato per controllare l'errore di troncamento. Tali combinazioni costituiscono delle sequenze che generalizzano la (5.43) e si esprimono

nella forma

$$\begin{aligned}
(\Delta y)_0 &= h\, f\left(t_n\,,\; y_n\right)\\
(\Delta y)_1 &= h\, f\left(t_n + \alpha_1 h\,,\; y_n + \alpha_1 (\Delta y)_0\right)\\
(\Delta y)_2 &= h\, f\left(t_n + \alpha_2 h\,,\; y_n + \alpha_2 (\Delta y)_1\right)\\
&\dots \qquad \dots\\
(\Delta y)_{N_S} &= h\, f\left(t_n + \alpha_{N_S} h\,,\; y_n + \alpha_{N_S} (\Delta y)_{N_S-1}\right)\\
y_{n+1} &= y_n + \beta_0 (\Delta y)_0 + \beta_1 (\Delta y)_1 + \beta_2 (\Delta y)_2 + \ldots + \beta_{N_S} (\Delta y)_{N_S}
\end{aligned} \tag{5.45}$$

in cui ciascuna valutazione della f rappresenta il cosidetto *stadio*, i numeri β_k sono coefficienti di peso degli incrementi $(\Delta y)_k$ ed i coefficienti α_k esprimono il punto in cui valutare il corrispondente stadio nell'intervallo $[t_n, t_{n+1}]$. Le (5.45) esprimono la forma generale di una classe di schemi numerici di integrazione al passo detti *schemi multistadio*.

È possibile dimostrare che, al crescere del numero degli stadi, con opportune combinazioni dei coefficienti α_k e β_k, si possono via via eliminare dall'errore di approssimazione

$$\boldsymbol{E} = \tilde{\boldsymbol{y}}(t_{n+1}) - \tilde{\boldsymbol{y}}(t_n) - \sum_{k=0}^{N_S} \beta_k (\Delta \boldsymbol{y})_k \qquad \left(\text{dove} \quad \sum_{k=0}^{N_S} \beta_k = 1\right) \tag{5.46}$$

i contributi di $O(h^2)$, $O(h^3)$, $O(h^4)$, eccetera. Questa è l'idea in base alla quale è stato sviluppato il metodo di Runge-Kutta.

5.4.3 Il metodo di Runge-Kutta

La versione più diffusa di questo metodo di integrazione al passo corrisponde alla cosiddetta *formula di Runge-Kutta del quarto ordine*. Tale procedura viene scritta classicamente nella forma

$$\begin{aligned}
(\Delta y)_0 &= h\, f\left(t_n\,,\; y_n\right)\\
(\Delta y)_1 &= h\, f\left(t_n + \frac{1}{2}h\,,\; y_n + \frac{1}{2}(\Delta y)_0\right)\\
(\Delta y)_2 &= h\, f\left(t_n + \frac{1}{2}h\,,\; y_n + \frac{1}{2}(\Delta y)_1\right)\\
(\Delta y)_3 &= h\, f\left(t_n + h\,,\; y_n + (\Delta y)_2\right)\\
y_{n+1} &= y_n + \frac{1}{6}(\Delta y)_0 + \frac{1}{3}(\Delta y)_1 + \frac{1}{3}(\Delta y)_2 + \frac{1}{6}(\Delta y)_3
\end{aligned} \tag{5.47}$$

Il metodo di Runge-Kutta espresso dalle (5.47), noto anche come *4-stage stepping scheme*, è la particolarizzazione delle (5.45) per $N_S = 3$ ed è del quarto ordine perché si dimostra che l'errore di troncamento e di $O(h^5)$. I coefficienti $\beta_0 = \beta_3 = \frac{1}{6}$, $\beta_1 = \beta_2 = \frac{1}{3}$, $\alpha_1 = \alpha_2 = \frac{1}{2}$ ed $\alpha_3 = 1$ sono i *coefficienti della formula di Runge-Kutta del quarto ordine*.

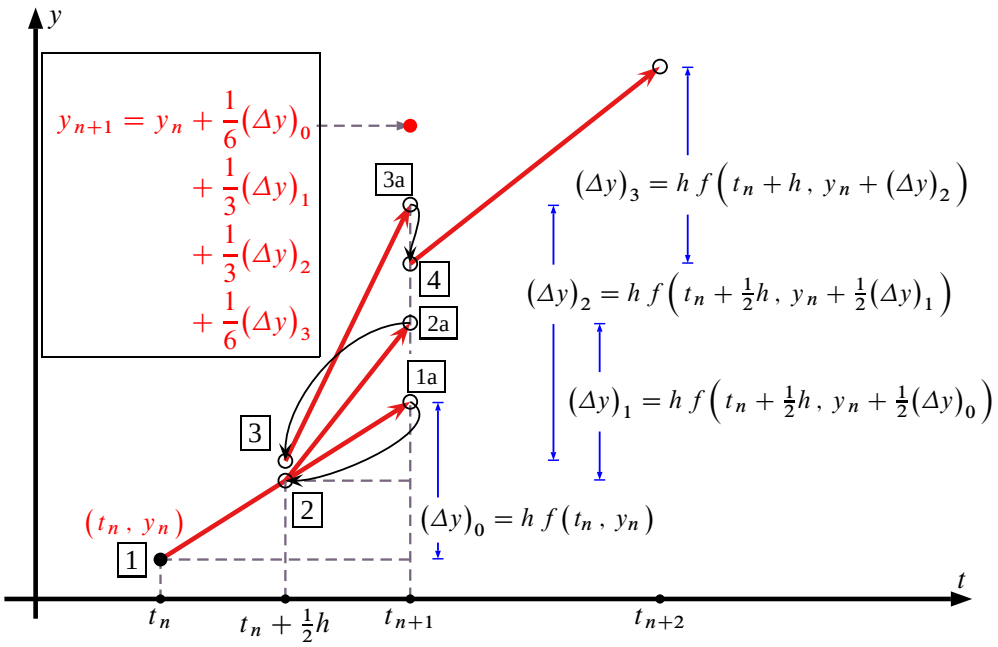

Figura 5.19 Metodo di Runge-Kutta del quarto ordine. La derivata è valutata quattro volte, una volta al punto iniziale (1), due volte in due punti intermedi di prova (2) e (3) ed infine al punto di prova finale (4). Dalla combinazione di tali valori si ottiene il valore successivo dell'incognita y.

Le (5.47) richiedono quattro valutazioni della funzione $\boldsymbol{f}$ per ciascun passo h come mostrato nella figura 5.19. Nella grande maggioranza dei casi la sequenza (5.47) è superiore alla formulazione del secondo ordine (5.43). Si vedano, ad esempio, le *Numerical Recipes* [73].

Concludiamo riassumendo i passi necessari all'applicazione pratica del metodo di Runge-Kutta ad un sistema di equazioni.

1. In primo luogo occorrerà, per ciascuna delle equazioni che compongono il sistema da risolvere, calcolare in sequenza i valori $(\Delta \boldsymbol{y})_0$, $(\Delta \boldsymbol{y})_1$, $(\Delta \boldsymbol{y})_2$ e $(\Delta \boldsymbol{y})_3$ dalle (5.47) con $n = 0$ e per t_0 ed $\boldsymbol{y}_0$ assegnati inizialmente.

2. Sempre per ciascuna equazione del sistema, occorrerà costruire il valore di $\boldsymbol{y}_1$ relativo a $t_0 + h$ secondo l'ultima delle (5.47), utilizzando a questo scopo i 4 valori intermedi $(\Delta \boldsymbol{y})_k$ precedentemente conservati.

3. Il processo potrà quindi essere iterato, sostituendo $t_0 + h$ a t_0 ed $\boldsymbol{y}_1$ a $\boldsymbol{y}_0$, ricominciando dal punto 1 fino a coprire l'intervallo desiderato delle t.

5.4.4 Esempio applicativo in Matlab

Come esempio di applicazione del metodo di integrazione di Runge-Kutta viene proposto in questo paragrafo un problema di valori iniziali da risolvere numericamente, di cui è nota la soluzione analitica.

Il linguaggio Matlab possiede una funzione predefinita, `ode45` (`ode` sta per *Ordinary Differential Equations*), che permette un'agevole soluzione di questo tipo di problemi evitando all'utente la necessità di un'implementazione a basso livello dell'algoritmo di integrazione espresso dalle formule (5.47).

Il problema di valori iniziali è il seguente:

$$\begin{aligned} \frac{\mathrm{d}^2 y}{\mathrm{d}t^2} + 3\cos^2 t - 2 = 0 \\ y(0) = \frac{\mathrm{d}y}{\mathrm{d}t}(0) = 0 \end{aligned} \tag{5.48}$$

Si può facilmente verificare che la soluzione analitica del problema (5.48) è costituita dalla funzione:

$$y(t) = \frac{1}{4}t^2 + \frac{3}{8}\cos(2t) - \frac{3}{8} \tag{5.49}$$

Per risolvere il problema assegnato è necessario implementare una funzione Matlab, che chiameremo **myfunction2x1** per ricordare che l'output è un vettore colonna 2×1, e salvarla in un *M-file* avente lo stesso nome nella cartella di lavoro scelta dall'utente. Un esempio di file **myfunction2x1**.m è il seguente:

```
function dydt = myfunction2x1(t,y)
% MYFUNCTION2X1 : funzione di 3 variabili, (t,y1,y2), a due componenti
% da valutare nella risoluzione dei problemi di valori iniziali associati
% all'equazione
% (1)               y'' + 3 cos^2(t) - 2 = 0
%
% La (1) si riscrive in forma di sistema di equazioni del primo ordine
% (2a)              y1' = y2
% (2b)              y2' = -3cos^2(t) + 2
%

% Alloca ed inizializza un vettore colonna 2x1
dydt = zeros(2,1);

% Definisce il vettore dydt, variabile di output
dydt(1) = y(2);             % implementa la dipendenza (2a)
dydt(2) = ...               % implementa la dipendenza (2b)
    - 3*(cos(t))^2 + 2;
```

Nella sequenza di comandi riportata più avanti viene fissato inizialmente un numero di passi di integrazione e conservato nella variabile `NSTEP`. Un valore finale della variabile indipendente pari a `6.28` viene conservato nella variabile `tFin`. Successivamente viene assegnato un vettore riga `t` con il comando `linspace`, che assegna ai suoi `NSTEP` elementi dei valori equispaziati nell'intervallo $[0, 2\pi]$. La funzione `ode45` viene poi invocata dopo aver fissato il vettore delle condizioni iniziali `y0`. Il grafico della soluzione esatta e della soluzione numerica del problema proposto è riportato infine in figura 5.20 ed è ottenuto con il comando `plot`.

```
>> NSTEP = 40; tFin = 2*pi;
>> t = linspace(0,tFin,NSTEP)
>> y0 = [0 0];
>> [Tt,Y] = ode45(@myfunction2x1,t,y0);
>> Yexact = 0.25*Tt.^2+(3./8.)*cos(2.*Tt) -(3./8.);
>> plot(Tt,Y(:,1),'*',Tt,Yexact,'-')
>> xlabel('t'); ylabel('y'); legend('Runge-Kutta','Soluzione esatta');
```

Si noti che nella chiamata alla funzione `ode45` viene passato quello che nel gergo di Matlab è un *function handle*, cioè il *puntatore a funzione* `@myfunction2x1` che si ottiene anteponendo il carattere speciale '`@`' al nome della funzione che definisce il problema

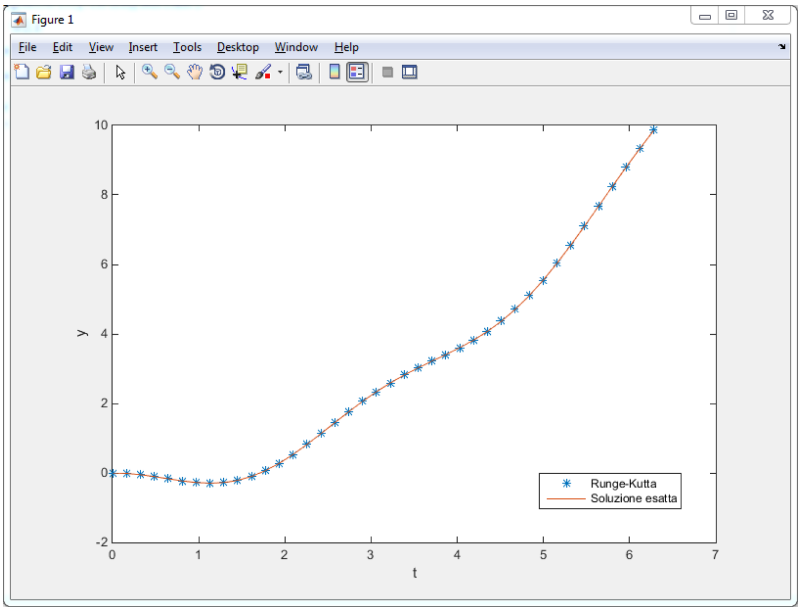

Figura 5.20 Finestra del plot di Matlab in cui la soluzione numerica del problema di valori iniziali (5.48) con metodo di Runge-Kutta del quarto ordine (`ode45`) è confrontata con la soluzione esatta (5.49) nell'intervallo $[0, 2\pi]$.

differenziale studiato. La funzione `ode45` restituisce un vettore `Tt`, in questo caso identico a `t`, ed una matrice `Y` di `NSTEP` righe e 2 colonne. Ciascuna delle righe rappresenta la soluzione numerica ottenuta in corrispondenza di ciascuno dei valori discreti della variabile indipendente contenuti in `Tt`. Il motivo per cui viene restituito un vettore come `Tt` nasce dalla possibilità di invocare funzioni come `ode45` abilitando l'opzione di passo di integrazione variabile in caso di forte variabilità locale della funzione che implementa la derivata. Si rimanda all'*help* o al riferimento [69] per maggiori dettagli sull'utilizzo delle funzioni `ode45`, `ode23`, `ode113`, `ode15s`, `ode23s`, `ode23t`, `ode23tb`.

Bibliografia

[1] W. R. Hamilton, *Lectures on Quaternions*, Hodeges & Smith, 1853.

[2] O. Rodrigues, "Des lois géometriques qui régissent les désplacements d'un système solide dans l'espace, et de la variation des coordonnée provenant de ses désplacements considerées indépendamment des causes qui peuvent les produire", *Journal des Mathématiques Pures et Appliquées*, vol. 5, 1840.

[3] E. Salamin, "Application of Quaternions to Computation with Rotations", Working paper, Stanford AI Lab, 1979.

[4] A. P. Yefremov, "Quaternions: Algebra, Geometry and Physical Theories", *Hypercomplex Numbers in Geometry and Physics*, vol. 1, 2004.

[5] Schwab A. L., "Quaternions, Finite Rotations and Euler Parameters", Course notes on Applied Multibody Dynamics, Delft University of Technology, Laboratory for Engineering Mechanics, 2003.
http://tam.cornell.edu/~{}als93/quaternion.pdf.

[6] AIAA/ANSI, *Recommended Practice for Atmospheric and Space Flight Vehicle Coordinate Systems*. R-004-1992, 1992.

[7] G. H. Bryan, *Stability in Aviation: An Introduction to Dynamical Stability as Applied to the Motions of Aeroplanes*. Macmillan and Co., Limited, London, 1911.

[8] D. J. Diston, *Computational Modelling of the Aircraft and the Environment. Volume 1, Platform Kinematics and Synthetic Environment*. John Wiley & Sons, Inc., 2009.

[9] W. F. Phillips, *Mechanics of Flight*. John Wiley & Sons, Inc., 2004.

[10] W. F. Phillips, "Phugoid Approximation for Conventional Airplanes", Journal of Aircraft, Vol. 37, No. 1, January-February 2000.

[11] W. F. Phillips, "Improved Closed-Form Approximation for Dutch-Roll", Journal of Aircraft, Vol. 37, No. 1, May-June 2000.

[12] R. Stengel, *Flight Dynamics*. Princeton University Press, Princeton, 2004.

[13] M. R. Napolitano, *Aircraft Dynamics: From Modeling to Simulation*. John Wiley, 2012.

[14] D. K. Schmidt, *Modern Flight Dynamics*. McGraw-Hill, 2010.

[15] B. Stevens, F. Lewis, *Aircraft Control and Simulation*. John Wiley & Sons, Inc., 1992.

[16] D. Stinton, *The Anatomy of the Airplane* (2nd edition). American Institute of Aeronautics and Astronautics, 1998.

[17] B. Etkin, *Dynamics of Flight, Stability and Control*. John Wiley & Sons, New York, 1982.

[18] M. Calcara, *Elementi di dinamica del velivolo*. Edizioni CUEN, Napoli, 1988.

[19] L. V. Schmidt, *Introduction to Aircraft Flight Dynamics*. AIAA Education Series, 1998.

[20] W. J. Duncan, *Control and Stability of Aircraft*. Cambridge University Press, Cambridge, 1952.

[21] R. Jategaonkar, *Flight Vehicle System Identification: A Time Domain Methodology*. Progress in Astronautics and Aeronautics Series, 2006.

[22] C. D. Perkins, R. E. Hage, *Aircraft Performance, Stability and Control*. John Wiley & Sons, New York, 1949.

[23] J. R. Wright, J.. E. Cooper, *Introduction to Aircraft Aeroelasticity and Loads*. John Wiley & Sons, Inc., 2007.

[24] V. Losito, *Fondamenti di Aeronautica Generale*. Accademia Aeronautica, Napoli, 1994.

[25] E. Torenbeek, H. Wittenberg, *Flight Physics*. Springer, Heidelberg, 2009.

[26] P. H. Zipfel, *Modeling and Simulation of Aerospace Vehicle Dynamics*. Second Edition. AIAA Education Series, American Institute of Aeronautics and Astronautics, Reston, VA. 2007.

[27] J. D. Mattingly, *Elements of Propulsion: Gas Turbines and Rockets*. AIAA Education Series, American Institute of Aeronautics and Astronautics, Reston, VA. 2006.

[28] K. Hünecke, *Jet Engines. Fundamentals of Theory, Design and Operation*. Motorbooks International, 1997.

[29] A. Linke-Diesinger, *Systems of Commercial Turbofan Engines*. Springer-Verlag, Berlin Heidelberg, 2008.

[30] F. R. Garza, E. A. Morelli, "A Collection of Nonlinear Aircraft Simulations with MATLAB". NASA-TM-2003-212145, January 2003.

[31] Voce WGS84 su *Wikipedia*:
http://en.wikipedia.org/wiki/World_Geodetic_System

[32] Anonimo, *Department of Defense World Geodetic System 1984. Its Definition and Relationship with Local Geodetic Systems*. NIMA TR8350.2, Third Edition, Amendment 2. National Imagery and Mapping Agency, US Department of Defense, 2004.

[33] J. Roskam, *Airplane Flight Dynamics and Automatic Flight Controls*. DARcorporation, 2001.

[34] H. T. Schlichting, E. A. Truckenbrodt, *Aerodynamics of the Aeroplane*. McGraw Hill Higher Education, 2nd edition, 1979.

[35] M. M. Munk, "The aerodynamic forces on airship hulls". NACA-TR-184, 1924.

[36] A. Silverstein, S. Katzoff, "Aerodynamic characteristics of horizontal tail surfaces". NACA-TR-688, 1940.

[37] R. I. Sears, "Wind-tunnel data on the aerodynamic characteristics of airplane control surfaces". NACA-WR-L-663, 1943.

[38] E. Garner, "Wind-tunnel investigation of control-surface characteristics XX: plain and balanced flaps on an NACA 0009 rectangular semispan tail surface". NACA-WR-L-186, 1944.

[39] J. D. Brewer, M. J. Queijo, "Wind-tunnel investigation of the effect of tab balance on tab and control-surface characteristics". NACA-TN-1403, 1947.

[40] S. M. Crandall, H. E. Murray, "Analysis of available data on the effects of tabs on control-surface hinge moments". NACA-TN-1049, 1946.

[41] B. W. McCormick, *Aerodynamics, Aeronautics, and Flight Mechanics*. John Wiley & Sons, 1979.

[42] B. N. Pamadi, *Performance, Stability, Dynamics and Control of Airplanes*. AIAA Education Series, 1998.

[43] A. Tewari, *Atmospheric and Space Flight Dynamics. Modelling and Simulation with Matlab and Simulink*. Birkhäuser, Berlin, 2007.

[44] D. Howe, *Aircraft Loading and Structural Layout*. AIAA Education Series, 2004.

[45] P. Morelli, *Static Stability and Control of Sailplanes*. Levrotto & Bella, Torino, 1976.

[46] L. Prandtl, O. G. Tietjens, *Fundamentals of Hydro and Aeromechanics*. Dover, 1957.

[47] R. K. Heffley, W. F. Jewell, "Aircraft Handling Qualities Data". NASA-CR-2144, December 1972.

[48] H. P. Stough III, J. M. Patton Jr, S. M. SliWa, "Flight Investigation of the Effect of Tail Configuration on Stall, Spin, and Recovery Characteristics of a Low-Wing General Aviation Research Airplane". NASA-TP-1987-2644, February 1987.

[49] J. D. Anderson, *Fundamentals of Aerodynamics*. McGraw-Hill, 3rd edition, New York, 2001.

[50] J. J. Bertin, *Aerodynamics for Engineers*. Prentice-Hall, 4th edition, Upper Saddle River, NJ, 2002.

[51] J. Katz, A. Plotkin, *Low-Speed Aerodynamics*. Cambridge University Press, 2nd edition, Cambridge, England, U.K., 2001.

[52] D. E. Hoak, *et al.*, "The USAF Stability and Control Datcom". Air Force Wright Aeronautical Laboratories, TR-83-3048, 1960 (Revised 1978).

[53] R. T. Jones, "A Note on the Stability and Control of Tailless Airplanes". NACA Report 837, 1941.

[54] D. P. Coiro, F. Nicolosi, A. De Marco, N. Genito, S. Figliolia, "Design of a Low Cost Easy-to-Fly STOL Ultralight Aircraft in Composite Material". *Acta Polytecnica*, Vol. 45 no. 4, 2005, pp. 73-80; ISSN 1210-2709.

[55] F. Nicolosi, A. De Marco, P. Della Vecchia, "Flight Tests, Performances and Flight Certification of a Twin-Engine Light Aircraft". *Journal of Aircraft*, Vol 48, No. 1, January-February 2011.

[56] F. Nicolosi, A. De Marco, P. Della Vecchia, "Parameter Estimation and Flying Qualities of a Twin-Engine CS23/FAR23 Certified Light Aircraft". AIAA-2010-7947, AIAA Atmospheric Flight Mechanics Conference, Toronto, 2010.

[57] B. Etkin, *Dynamics of Atmospheric Flight*, Dover Publications, 2005.

[58] L. Mangiacasale, *Flight Mechanics of a μ-Airplane*, Edizioni Libreria CLUP, Milano, 1998.

[59] G. Mengali, *Elementi di Dinamica del Volo con Matlab*, Edizioni ETS, Pisa, 2001.

[60] R. Nelson, *Flight Stability and Automatic Control*, McGraw-Hill, 1989.

[61] Y. Li, M. Nahon, "Modeling and simulations of airship dynamics", *Journal of Guidance, Controls and Dynamics*, Vol 30, No. 6, November-December 2007.

[62] Y. Fan, F. H. Lutze, E. M. Cliff, "Time-Optimal Lateral Maneuvers of an Aircraft", *Journal of Guidance, Controls and Dynamics*, Vol 18, No. 5, September-October 1995.

[63] J. N. Nielsen, *Missile Aerodynamics*, AIAA, Cambridge, MA, 1988.

[64] T. I. Fossen, *Guidance and Control of Ocean's Vehicles*, Whiley, New York, 1998.

[65] J. N. Newman, *Marine Hydrodynamics*, MIT Press, Cambridge, MA, 1977.

[66] E. L. Duke, R. F. Antoniewicz, K. D. Krambeer, "Derivation and Definition of a Linear Aircraft Model". Technical Report NASA Reference Publication RP-1207, Research Engineering, NASA Ames Research Center and NASA Dryden Flight Research Facility, 1988.

[67] G. A. Stagg, *An Unsteady Aerodynamic Model for Use in the High Angle of Attack Regime*. MS thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1998.

[68] Y. Fan, *Identification of an Unsteady Aerodynamic Model up to High Angle of Attack Regime*. PhD thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1997.

[69] *MATLAB Users' Guide*. The Mathworks, 2003 ed edizioni successive.
http://www.mathworks.com/
http://www.mathworks.com/access/helpdesk/help/techdoc/matlab.html

[70] V. Comincioli, *Analisi numerica: metodi, modelli, applicazioni*. McGraw-Hill, 1990, seconda edizione 1995.

[71] E. Kreyszig, *Advanced Engineering Mathematics*. John Wiley & Sons, seventh edition, 1993.

[72] C. de Boor, *A Practical Guide to Splines*. Springer-Verlag, 1978.

[73] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, *Numerical Recipes in Fortran: The Art of Scientific Computing*. Cambridge University Press, 1992.

[74] G. Dahlquist, A. Bjorck, *Numerical Methods. Volume I: Fundamentals of Numerical Discretization*. John Wiley & Sons, 1988.

[75] R. D. Richtmyer, K. W. Morton, *Difference Methods for Initial Value Problems*. Wiley-Interscience, 1967.

[76] C. Hirsch, *Numerical Computation of Internal and External Flows*. John Wiley & Sons, 1994.

[77] R. D. Finck, "USAF Stability and Control Datcom". AFWAL-TR-83-3048, October 1960, Revised 1978.

[78] S. R. Vukelich, J. E. Williams, "The USAF Stability and Control Digital Datcom". AFFDL-TR-79-3032, Volume I, April 1979, Updated by Public Domain Aeronautical Software 1999.

[79] W. B. Blake, "Prediction of Fighter Aircraft Dynamic Derivatives Using Digital Datcom". AIAA-85-4070, AIAA Applied Aerodynamics Conference, Colorado Springs, Colorado, 1985.

[80] Autori Vari, Distribuzione ufficiale di Digital Datcom,
sito internet:
http://wpage.unina.it/agodemar/DSV-DQV/Digital-Datcom-Package.zip

[81] B. Galbraith, "Digital Datcom+", Holy Cows, Inc.,
sito internet: http://www.holycows.net/datcom/