Manualetto di Matlab®

Silvia Falletta

1 Comandi d'avvio

Per avviare MATLAB in ambiente Windows è sufficiente selezionare con il mouse l'icona corrispondente. In ambiente MsDos o in ambiente Unix basta digitare `matlab` e premere il tasto di invio (o enter, return, ...). Il simbolo `>>` che compare, è il prompt di MATLAB. Per eseguire un comando digitato occorre premere il tasto di invio. Per terminare la sessione di lavoro occorre digitare il comando `exit` oppure `quit`.

Tabella 1. Alcuni comandi per gestire una sessione di lavoro.

Comando	Significato
`help`	per visualizzare tutti gli argomenti presenti
`help arg`	per visualizzare informazioni su arg
`doc arg`	per visualizzare dettagliate informazioni su arg
`clc`	per cancellare il contenuto della finestra di lavoro
`;`	per non visualizzare il risultato di un'istruzione
`...`	per continuare a scrivere un'istruzione nella riga successiva
`who`	per visualizzare le variabili poste in memoria
`whos`	per visualizzare informazioni sulle variabili poste in memoria
`clear`	per cancellare tutte le variabili dalla memoria
`clear var1 var2`	per cancellare le variabili `var1` e `var2` dalla memoria

2 Le variabili in Matlab

I nomi delle variabili possono essere lunghi al massimo 32 caratteri. I caratteri utilizzabili sono le lettere (maiuscole e minuscole), i numeri e il carattere "_" (underscore). Un nome di variabile deve cominciare con un carattere alfabetico (a-z, A-Z). MATLAB distingue tra lettere maiuscole e minuscole (ad esempio i nomi `a1` ed `A1` rappresentano variabili diverse). La variabile si crea automaticamente nel momento in cui si assegna ad essa un valore o il risultato di un'espressione. L'assegnazione avviene mediante il simbolo `=` secondo la seguente sintassi

```
>> nome_variabile=espressione
```

Se la variabile che si vuole creare è di tipo stringa occorre racchiudere `espressione` tra una coppia di apici. Nella tabella 2 abbiamo riportato alcune variabili scalari predefinite.

MATLAB lavora con sedici cifre significative. Tuttavia, in output una variabile intera viene visualizzata generalmente in un formato privo di punto decimale, mentre una variabile reale (non intera) viene visualizzata solo con quattro cifre decimali. Se si vuole modificare il formato di output si può utilizzare uno dei comandi della tabella 3. Per visualizzare tutte le sedici cifre impiegate da MATLAB è necessario attivare il comando `format long e`.

Nella tabella 4 abbiamo riportato le principali operazioni eseguibili sulle variabili scalari. Oltre alle operazioni di base, in MATLAB sono presenti anche le funzioni predefinite riportate nella tabella 5.

Gli elementi di un vettore vanno digitati tra parentesi quadre; gli elementi di un vettore riga vanno separati con uno spazio oppure una virgola, quelli di un vettore colonna con un punto e virgola

Tabella 2. Alcune variabili predefinite in MATLAB.

Variabile	Significato
`ans`	variabile temporanea che contiene il risultato più recente
`i,j`	unità immaginaria
`pi`	π, 3.14159265...
`eps`	*epsilon* di macchina
`realmax`	massimo numero di macchina positivo
`realmin`	minimo numero di macchina positivo
`Inf`	∞, ossia un numero maggiore di `realmax` oppure il risultato di 1/0
`NaN`	Not a Number (per esempio, il risultato di 0/0)

Tabella 3. Alcuni possibili formati di output in MATLAB.

Formato	Azione
`format`	formato di default, equivalente a `format short`
`format short`	rappresentazione fixed-point con 4 cifre decimali
`format long`	rappresentazione fixed-point con 14 cifre decimali
`format short e`	rappresentazione floating-point con 4 cifre decimali
`format long e`	rappresentazione floating-point con 15 cifre decimali
`format rat`	rappresentazione sottoforma di frazione irriducibile

Tabella 4. Operazioni fra variabili scalari.

Operazione	Significato
`+`	addizione
`-`	sottrazione
`*`	moltiplicazione
`/`	divisione
`^`	elevamento a potenza

oppure premendo il tasto di invio dopo l'introduzione di ogni elemento. In MATLAB non è possibile utilizzare indici nulli o negativi per le componenti di un vettore. Il comando `x(i)` individua l'i-esimo elemento del vettore `x`, `x(end)` individua l'ultimo elemento del vettore `x` e `length(x)` determina la lunghezza del vettore `x`. Nella tabella 6 abbiamo riportato alcuni comandi per generare e manipolare vettori.

Nella tabella 7 abbiamo riportato alcuni comandi che operando sul vettore $x = (x_i)_{i=1,\dots,n}$ consentono di calcolare particolari grandezze ad esso relative.
La funzione `diag` può essere utilizzata anche nella forma `diag(x,k)`, con k intero positivo o negativo; in questo caso essa genera una matrice quadrata di dimensione $n+|k|$ con tutti gli elementi uguali a zero tranne quelli della k-esima diagonale sopra ($k > 0$) oppure sotto ($k < 0$) la diagonale principale, che coincidono con gli elementi del vettore x.

Gli elementi di una matrice vanno digitati tra parentesi quadre, procedendo per righe e terminando ciascuna riga con un punto e virgola oppure premendo il tasto di invio. Il comando `A(i,j)` individua l'elemento di posto (i, j), `size(A)` genera un vettore riga contenente il numero di righe e il numero di colonne della matrice `A` e `length(A)` applicato ad una matrice, anziché a un vettore, equivale a calcolare `max(size(A))`.

Nella tabella 8 abbiamo riportato alcune funzioni MATLAB che consentono di generare particolari matrici quadrate e/o rettangolari. Osserviamo inoltre che nei comandi della tabella 8 quando $m \equiv n$ è sufficiente digitare solo il parametro n.

Tabella 5. Alcune funzioni predefinite in MATLAB.

Funzione	**Significato**
`sin`	seno
`cos`	coseno
`asin`	arcoseno
`acos`	arcocoseno
`tan`	tangente
`atan`	arcotangente
`exp`	esponenziale
`log`	logaritmo naturale
`log2`	logaritmo in base 2
`log10`	logaritmo in base 10
`sqrt`	radice quadrata
`abs`	valore assoluto o modulo
`real`	parte reale
`imag`	parte immaginaria
`sign`	funzione segno
`factorial`	fattoriale
`round`	arrotonda all'intero più vicino
`floor`	arrotonda per difetto all'intero più vicino
`ceil`	arrotonda per eccesso all'intero più vicino
`chop(x,t)`	arrotonda `x` a `t` cifre significative

Tabella 6. Alcuni comandi per generare e manipolare vettori.

Comando	**Azione**
`x'`	genera il vettore trasposto di `x`
`x=[]`	genera il vettore vuoto `x`
`sort(x)`	riordina in ordine crescente le componenti del vettore `x`
`x=[a:h:b]`	genera il vettore riga $x = (x_i)_{i=1,\dots,m+1}$ ove $x_i = a + (i-1)h$ e m è la parte intera di $(b-a)/h$
`x=linspace(a,b,n)`	genera il vettore riga $x = (x_i)_{i=1,\dots,n}$ ove $x_i = a + (i-1)h$ e $h = (b-a)/(n-1)$
`x=logspace(a,b,n)`	genera il vettore riga $x = (10^{x_i})_{i=1,\dots,n}$ ove $x_i = a + (i-1)h$ e $h = (b-a)/(n-1)$
`x(r)`	estrae le componenti del vettore `x` i cui indici sono specificati in `r`
`x(r)=z`	assegna alle componenti del vettore `x` (i cui indici sono specificati in `r`) i valori definiti in z rispettivamente
`x(r)=[]`	rimuove le componenti del vettore `x` (i cui indici sono specificati in `r`)
`x([i j])=x([j i])`	scambia le componenti i e j del vettore `x`

Nella tabella 9 abbiamo riportato alcuni comandi che operando sulla matrice $A = (a_{ij})_{i,j=1,\dots,n}$ consentono di calcolare particolari grandezze ad essa relative.
Osserviamo che le funzioni della tabella 9 si possono applicare anche a matrici rettangolari. Inoltre, le funzioni `sum`, `max` e `min` possono essere utilizzate anche nella forma `sum(A,k)`, `max(A,[],k)` e `min(A,[],k)`, con $k = 1, 2$. Per $k = 1$ agiscono in maniera analoga a quella specificata nella tabella 9. Per $k = 2$ generano un vettore colonna $x = (x_i)_{i=1,\dots,n}$, con $x_i = \sum_{j=1}^{n} a_{ij}$, $x_i = \max_j a_{ij}$ e $x_i = \min_j a_{ij}$, rispettivamente. La funzione `diag` può essere utilizzata anche nella forma `diag(A,k)`, con `k` intero positivo o negativo; in questo caso essa genera un vettore colonna coincidente con la k-

Tabella 7. Alcune funzioni predefinite in MATLAB agenti su un vettore `x`.

Comando	Azione
`a=sum(x)`	genera lo scalare $a = \sum_{i=1}^{n} x_i$
`a=prod(x)`	genera lo scalare $a = \prod_{i=1}^{n} x_i$
`a=max(x)`	genera lo scalare $a = \max_i x_i$
`a=min(x)`	genera lo scalare $a = \min_i x_i$
`a=norm(x)`	genera lo scalare $a = \|\|x\|\|_2$
`a=norm(x,1)`	genera lo scalare $a = \|\|x\|\|_1$
`a=norm(x,inf)`	genera lo scalare $a = \|\|x\|\|_\infty$
`A=diag(x)`	genera la matrice diagonale $A = (a_{ij})_{i,j=1,\ldots,n}$, con $a_{ii} = x_i$

Tabella 8. Alcuni comandi per generare e manipolare matrici.

Comando	Azione
`A=[]`	genera la matrice vuota A
`A'`	genera la matrice trasposta di A
`A=eye(n)`	genera la matrice identità $A = (a_{ij})_{i,j=1,\ldots,n}$, con $a_{ij} = \delta_{ij}$
`A=zeros(n,m)`	genera la matrice $A = (a_{ij})_{i=1,\ldots,n,j=1,\ldots,m}$, con $a_{ij} = 0$
`A=ones(n,m)`	genera la matrice $A = (a_{ij})_{i=1,\ldots,n,j=1,\ldots,m}$, con $a_{ij} = 1$
`A=rand(n,m)`	genera la matrice $A = (a_{ij})_{i=1,\ldots,n,j=1,\ldots,m}$, con $0 < a_{ij} < 1$ pseudo-casuali
`A=hilb(n)`	genera la matrice di Hilbert $A = (a_{ij})_{i,j=1,\ldots,n}$, con $a_{ij} = 1/(i+j-1)$
`A=vander(x)`	genera la matrice di Vandermonde $A = (a_{ij})_{i,j=1,\ldots,n}$, con $a_{ij} = x_i^{n-j}$
`A(r,c)`	estrae gli elementi di `A` appartenenti all'intersezione delle righe e delle colonne specificate in `r` e in `c` rispettivamente
`A(r,c)=C`	assegna agli elementi di `A` (i cui indici di riga e di colonna sono specificati in `r` e in `c`) i valori definiti in C rispettivamente
`A(r,c)=[]`	rimuove gli elementi di `A` (i cui indici di riga e di colonna sono specificati in `r` e in `c`)
`A([i j],c)=A([j i],c)`	scambia gli elementi delle righe `i` e `j` di `A` appartenenti alle colonne specificate in `c`
`A(r,[i j])=A(r,[j i])`	scambia gli elementi delle colonne `i` e `j` di `A` appartenenti alle righe specificate in `r`

esima diagonale sopra ($k > 0$) oppure sotto ($k < 0$) la diagonale principale. Anche la funzione `tril` (`triu`) può essere utilizzata nella forma `tril(A,k)` (`triu(A,k)`), con `k` intero positivo o negativo; in questo caso essa estrae la parte triangolare inferiore (superiore) a partire dalla k-esima diagonale sopra ($k > 0$) oppure sotto ($k < 0$) la diagonale principale.

Le operazioni elementari, che si eseguono tra scalari, si estendono (quando ben definite) in modo del tutto naturale ai vettori e alle matrici, con l'eccezione delle operazioni di divisione e di elevamento a potenza. L'operazione `*` esegue il prodotto righe per colonne. Nel caso di matrici quadrate il comando `A^k` con `k` intero positivo fornisce il prodotto (righe per colonne) della matrice A per se stessa k volte; `A^(-1)` genera l'inversa della matrice A, ammesso che A sia non singolare.

Oltre alle operazioni classiche di somma e di prodotto fra vettori, o più in generale fra matrici, in MATLAB si possono considerare le cosiddette operazioni puntuali, che agiscono direttamente sui singoli elementi. Tali operazioni si definiscono premettendo un punto al simbolo che identifica l'operazione.

Dati i vettori riga (colonna) $x = (x_i)_{i=1,\ldots,n}$, $y = (y_i)_{i=1,\ldots,n}$, le matrici $A = (a_{ij})_{i,j=1,\ldots,n}$, $B = (b_{ij})_{i,j=1,\ldots,n}$ e il numero reale positivo e, nella tabella 10 abbiamo riportato le possibili operazioni

Tabella 9. Alcune funzioni predefinite in MATLAB agenti su una matrice `A`.

Comando	Azione
`a=norm(A)`	genera lo scalare $a = \|\|A\|\|_2$
`a=norm(A,1)`	genera lo scalare $a = \|\|A\|\|_1$
`a=norm(A,inf)`	genera lo scalare $a = \|\|A\|\|_\infty$
`x=sum(A)`	genera il vettore riga $x = (x_j)_{j=1,\dots,n}$, con $x_j = \sum_{i=1}^{n} a_{ij}$
`x=max(A)`	genera il vettore riga $x = (x_j)_{j=1,\dots,n}$, con $x_j = \max_i a_{ij}$
`x=min(A)`	genera il vettore riga $x = (x_j)_{j=1,\dots,n}$, con $x_j = \min_i a_{ij}$
`x=diag(A)`	genera il vettore colonna $x = (x_i)_{i=1,\dots,n}$, con $x_i = a_{ii}$
`B=abs(A)`	genera la matrice $B = (b_{ij})_{i,j=1,\dots,n}$, con $b_{ij} = \|a_{ij}\|$
`B=tril(A)`	genera la matrice triangolare inferiore $B = (b_{ij})_{i,j=1,\dots,n}$, con $b_{ij} = a_{ij}$, $i = 1, \dots, n$, $1 \le j \le i$
`B=triu(A)`	genera la matrice triangolare superiore $B = (b_{ij})_{i,j=1,\dots,n}$, con $b_{ij} = a_{ij}$, $i = 1, \dots, n$, $i \le j \le n$

puntuali.

Tabella 10. Operazioni puntuali in MATLAB.

Operazione	Azione
`z=x.*y`	genera il vettore riga (colonna) $z = \{z_i\}_{i=1,\dots,n}$, con $z_i = x_i * y_i$
`z=x./y`	genera il vettore riga (colonna) $z = \{z_i\}_{i=1,\dots,n}$, con $z_i = x_i / y_i$
`z=x.^y`	genera il vettore riga (colonna) $z = \{z_i\}_{i=1,\dots,n}$, con $z_i = x_i^{y_i}$
`z=x.^e`	genera il vettore riga (colonna) $z = \{z_i\}_{i=1,\dots,n}$, con $z_i = x_i^{e}$
`C=A.*B`	genera la matrice $C = (c_{ij})_{i,j=1,\dots,n}$, con $c_{ij} = a_{ij} * b_{ij}$
`C=A./B`	genera la matrice $C = (c_{ij})_{i,j=1,\dots,n}$, con $c_{ij} = a_{ij} / b_{ij}$
`C=A.^B`	genera la matrice $C = (c_{ij})_{i,j=1,\dots,n}$, con $c_{ij} = a_{ij}^{b_{ij}}$
`C=A.^e`	genera la matrice $C = (c_{ij})_{i,j=1,\dots,n}$, con $c_{ij} = a_{ij}^{e}$

3 La grafica in Matlab

Per disegnare una funzione f della variabile x si utilizza la funzione `fplot` secondo la sintassi

```
fplot('f',[xmin xmax])
```

ove `f` è l'espressione della funzione che si vuole rappresentare e `[xmin xmax]` è un vettore che ha per componenti gli estremi dell'intervallo di rappresentazione sull'asse x. Se si desidera stabilire anche un intervallo di rappresentazione sull'asse y occorre fornire il vettore `[xmin xmax ymin ymax]` come secondo argomento della funzione `fplot` .

Una procedura alternativa per disegnare una funzione $f(x)$ consiste nel definire un vettore `x` di punti dell'asse x, generare il vettore `y` contenente le valutazioni della funzione f nei punti precisati in `x`, e quindi utilizzare il comando `plot` secondo la sintassi

```
plot(x,y)
```

Entrambi i comandi `plot` e `fplot` consentono di personalizzare il grafico mediante la scelta del colore della linea, del simbolo da mettere in corrispondenza dei punti (x_i, y_i) e del tipo di linea (continua, punteggiata, tratteggiata, ...). Nella tabella 11 abbiamo riportato alcune delle possibili opzioni che si possono utilizzare. Inoltre, nella tabella 12 abbiamo riportato alcuni comandi

Tabella 11. Alcune possibili opzioni per i comandi `plot` e `fplot`.

Colore	Significato	Simbolo	Significato	Linea	Significato
`w`	bianco	`.`	punto	`-`	linea continua
`y`	giallo	`o`	circoletto	`:`	linea punteggiata
`r`	rosso	`x`	per	`-.`	linea tratto-punto
`g`	verde	`+`	più	`--`	linea tratteggiata
`b`	blu	`*`	asterisco		
`k`	nero	`s`	quadratino		

Tabella 12. Alcuni possibili comandi per commentare un grafico.

Comando	Azione
`title`	inserisce un titolo nel grafico
`xlabel`	inserisce un nome per l'asse x
`ylabel`	inserisce un nome per l'asse y
`grid`	inserisce una griglia sugli assi x ed y
`legend`	inserisce una legenda per identificare rappresentazioni diverse
`text`	inserisce una stringa di testo in una specificata posizione
`gtext`	inserisce una stringa di testo in una posizione individuata tramite mouse

disponibili per commentare un grafico.

Se si desidera considerare una scala logaritmica sull'asse delle x, o sull'asse delle y, o su entrambi gli assi occorre utilizzare al posto del comando `plot` il comando `semilogx`, `semilogy`, `loglog` rispettivamente. L'uso di questi comandi è del tutto analogo a quello del comando `plot`.

Ogni qual volta si digita il comando `plot`, MATLAB cancella il grafico precedente e traccia quello nuovo. Per disegnare più grafici nella stessa finestra grafica si può digitare il comando `hold on` prima di disegnare il grafico che si vuole sovrapporre a quello già tracciato oppure utilizzare il comando `plot` nella forma

```
plot(x_1,y_1,'-',x_2,y_2,':')
```

Per disegnare grafici diversi in una stessa finestra grafica ma in sottofinestre separate si utilizza il comando `subplot` secondo la sintassi

```
subplot(righe,colonne,sottofinestra)
```

dove `righe` e `colonne` indicano il numero di righe e di colonne rispettivamente della matrice di sottofinestre della finestra grafica principale, e `sottofinestra` indica il numero della sottofinestra che si desidera attivare per disegnare un grafico all'interno di essa. Le sottofinestre vengono numerate a partire da sinistra verso destra e dall'alto verso il basso. Si possono mantenere aperte più finestre grafiche: per attivare la finestra grafica numero `n` si utilizza il comando `figure(n)`, per chiudere la suddetta finestra si digita il comando `close(n)`, per chiudere tutte le finestre attive si digita `close all`.

4 Programmi Matlab

Un file contenente istruzioni MATLAB viene generalmente chiamato *m-file*. Tale nome deriva dal fatto che esso deve essere salvato con l'estensione "`.m`". Il nome di un *m-file* può essere definito mediante lettere e/o numeri; è inoltre consentito anche il carattere `_` (underscore). In un *m-file* i comandi vanno digitati su righe differenti oppure sulla stessa riga purché siano separati da un punto

e virgola o da una virgola. Per introdurre un commento all'interno dell'*m-file* si deve utilizzare il carattere %.

Gli *m-file* possono essere di due tipi: *script* oppure *function*.

Gli *m-file* di tipo *script* sono definiti semplicemente da una sequenza di comandi MATLAB. Per eseguire un *m-file* di tipo *script* occorre selezionare la directory in cui l'*m-file* è stato salvato e, quindi, digitare il nome dell'*m-file* (senza estensione) al prompt di MATLAB. Gli *script* file non prevedono un passaggio di parametri di input ed output. Inoltre, le variabili definite in uno *script* sono variabili che rimangono nella memoria della sessione di lavoro, come se fossero state definite direttamente al prompt.

Gli *m-file* di tipo *function* devono cominciare necessariamente con la seguente struttura

```
function [y_1,y_2,...,y_n]=nome_function(x_1,x_2,...,x_m)
```

ove `y_1,y_2,...,y_n` sono i parametri di output e `x_1,x_2,...,x_m` sono i parametri di input. La stringa `nome_function` è il nome della *function*; quest'ultimo deve coincidere con il nome dell'*m-file* (esclusa, ovviamente, l'estensione `.m`) in cui è stata salvata la *function*. Per eseguire una *function* dal prompt, o anche all'interno di uno *script* o di un'altra *function*, occorre digitare

```
[y_1,y_2,...,y_n]=nome_function(x_1,x_2,...,x_m)
```

oppure

```
nome_function(x_1,x_2,...,x_m)
```

In questo caso la *function* restituisce solo il primo parametro di output che viene salvato nella variabile `ans`.

Osserviamo che è necessario assegnare un valore a ciascun parametro di output; in caso contrario MATLAB restituisce il seguente messaggio

```
??? One or more output arguments not assigned during call to 'nome_function'
```

A differenza degli *script*, gli *m-file* di tipo *function* prevedono in ingresso dei parametri di input e in uscita dei parametri di output e le variabili di servizio utilizzate durante l'esecuzione di una *function* vengono trattate come variabili locali e vengono automaticamente cancellate dalla memoria di MATLAB al termine dell'esecuzione della *function*.

4.1 Programmare in Matlab: costrutti sintattici

MATLAB come linguaggio di programmazione dispone delle strutture sintattiche tradizionali.

Nella tabella 13 abbiamo riportato gli operatori relazionali presenti in MATLAB.

Tabella 13. Operatori relazionali in MATLAB.

Operatore	Significato
<	minore
>	maggiore
<=	minore o uguale
>=	maggiore o uguale
==	uguale
∼=	non uguale

Con gli operatori relazionali è possibile effettuare confronti tra espressioni. Poiché MATLAB non dispone di variabili di tipo logico, assegna un valore numerico al risultato di un confronto. In

particolare, 0 rappresenta il valore falso e un qualsiasi numero diverso da zero, generalmente 1, rappresenta il valore vero.

Gli operatori relazionali sopra descritti possono essere combinati tra loro mediante gli operatori logici riportati nella tabella 14. Nella tabella 15 abbiamo riportato il risultato degli operatori logici

Tabella 14. Operatori logici di MATLAB.

Operatore	**Significato**
`&`	and
`|`	or
$\sim$	not
`xor`	or esclusivo

quando agiscono su due condizioni `a` e `b`.

Tabella 15. Azione degli operatori logici su due condizioni `a` e `b`.

`a`	`b`	`a & b`	`a | b`	$\sim$`a`	`xor(a,b)`
0	0	0	0	1	0
1	0	0	1	0	1
0	1	0	1	1	1
1	1	1	1	0	0

Di seguito elenchiamo alcune strutture di programmazione elementari disponibili in MATLAB.

- **ciclo incondizionato controllato da un contatore**

```
for indice = espressione
    blocco di istruzioni
end
```

  dove `indice` è una quantità che assume i valori definiti da `espressione` alla destra del segno di uguaglianza.

- **ciclo condizionato**

```
while condizione
    blocco di istruzioni
end
```

  dove `condizione` è un'espressione che MATLAB valuta e interpreta come vera se assume un valore diverso da zero, come falsa se assume il valore zero.

- **strutture condizionali**

```
if condizione_1
   blocco di istruzioni
elseif condizione_2
   blocco di istruzioni

     ⋮
```

```
else
   blocco di istruzioni
end
```

dove il primo `blocco di istruzioni` verrà eseguito solo se `condizione_1` risulta essere vera, il secondo solo se `condizione_1` risulta essere falsa e `condizione_2` vera e così via. Il blocco che segue `else` verrà eseguito soltanto se nessuna delle precedenti condizioni risulta essere vera. Le istruzioni `elseif` e `else` si possono omettere se non sono necessarie.

Il comando `return` consente di terminare l'esecuzione di un programma prima che si raggiunga l'ultima istruzione.

MATLAB dispone inoltre del comando `break`, che consente di uscire in maniera forzata da un ciclo. Quando tale comando viene eseguito MATLAB salta direttamente all'istruzione `end`, con cui termina il ciclo.

Per valutare l'efficienza di un programma in termini di tempo d'esecuzione espresso in secondi, si possono utilizzare i comandi `tic` e `toc`. Essi consentono di conoscere il numero dei secondi richiesto da un determinato `calcolo` e si utilizzano secondo la seguente sintassi

```
tic
    calcolo;
toc
```

`tic` attiva il timer, `toc` lo arresta e restituisce l'"`elapsed_time`", ovvero il tempo (in secondi) trascorso dal momento in cui `tic` è stato attivato.

5 Principali function Matlab per problemi di Calcolo Numerico

Tabella 16. Algebra lineare.

Function	**Scopo**
`lu`	genera la fattorizzazione di Gauss con pivoting parziale
`chol`	genera la fattorizzazione di Choleski
`qr`	genera la fattorizzazione QR
`x=A\b`	risolve il sistema lineare $Ax = b$
`cond(A)`	calcola il numero di condizionamento spettrale (in norma 2) di A
`cond(A,1)`	calcola il numero di condizionamento in norma 1 di A
`cond(A,inf)`	calcola il numero di condizionamento in norma ∞ di A
`rcond(A)`	calcola il reciproco del numero di condizionamento in norma 1 di A
`rank(A)`	calcola il rango di A
`det(A)`	calcola il determinante di A
`inv(A)`	calcola l'inversa di A
`eig`	calcola gli autovalori e gli autovettori di A

Tabella 17. Polinomi, funzioni e approssimazione.

Function	**Scopo**
`polyval`	valuta un polinomio
`f=inline('espressione','x_1',...,'x_n')`	definisce la funzione $f(x_1, \ldots, x_n) = espressione$
`y=f(x_1,...,x_n)`	valuta la funzione $y = f(x_1, \ldots, x_n)$ definita mediante `inline`
`y=feval(f,x_1,...,x_n)`	valuta la funzione $y = f(x_1, \ldots, x_n)$ definita mediante `inline` oppure mediante una function
`polyfit`	calcola i coefficienti del polinomio interpolante oppure approssimante nel senso dei minimi quadrati
`spline`	valuta una spline cubica interpolante

Tabella 18. Equazioni e sistemi di equazioni non lineari.

Function	**Scopo**
`fzero`	calcola gli zeri di una funzione non lineare
`roots`	calcola gli zeri di un polinomio
`fsolve`	risolve un sistema di equazioni non lineari

Tabella 19. Calcolo di integrali.

Function	**Metodo**
`quad`	formula di Simpson adattativa
`quadl`	formula di Gauss-Lobatto adattativa

Tabella 20. Equazioni e sistemi di equazioni differenziali ordinarie.

Function	**Metodo**
`ode45`	Runge-Kutta esplicito di ordine 4 e 5
`ode113`	Adams-Moulton di ordine variabile
`ode23`	Runge-Kutta esplicito di ordine 2 e 3
`ode23t`	trapezi
`ode15s`	multistep lineare implicito di ordine variabile
`ode23s`	Runge-Kutta implicito di ordine 2