

I. Il linguaggio MATLAB

1. Introduzione

La necessità di disporre di linguaggi di programmazione utilizzabili anche da utenti non esperti è stata sentita sin dalla nascita dell'Informatica. Ad esempio, nelle università e nei centri di ricerca si sono affiancati ai tipici linguaggi scientifici (es. FORTRAN) alcuni linguaggi nati per rendere più semplice l'approccio al calcolatore, mediante sintassi semplificate e funzioni di interfacciamento tra utente e 'macchina' che rendessero immediate tali operazioni (il linguaggio BASIC ne è un tipico esempio). Con l'evoluzione della tecnologia informatica, hardware e software, è nato e si è sviluppato un nuovo approccio alla programmazione che semplifica ulteriormente il lavoro dell'utente del calcolatore: la **programmazione visuale**. Attualmente è la tecnica di programmazione generalmente utilizzata da tutti gli applicativi disponibili sul mercato. In questo tipo di programmazione l'utente, al posto di scrivere una sequenza di istruzioni, può ricorrere a delle interfacce grafiche appositamente sviluppate che gli consentono di 'creare' il programma inserendo degli 'oggetti' già precostituiti (recuperabili da menu o mediante icone) i quali possono essere configurati di volta in volta per funzionare come operatori in grado di svolgere le più svariate operazioni: esecuzione di calcoli, elaborazioni di stringhe di caratteri, esecuzione di grafici, operazioni di input/output, ecc. Una volta che l'ambiente operativo è stato creato e configurato, l'applicativo genera automaticamente il programma che consente di ottenere l'ambiente voluto in grado di svolgere le funzioni definite dall'utente. In pratica è come se l'utente potesse descrivere il funzionamento del programma in modo grafico (visuale). A questo punto l'applicativo, sulla base della 'descrizione' dell'utente genera, senza intervento umano, il programma desiderato.

Il linguaggio MATLAB riunisce in sé le caratteristiche dei linguaggi di programmazione standard e di quelli ad approccio visuale. È, possiamo dire, un linguaggio 'intermedio' nel senso che consente la costruzione di programmi scritti dall'utente, istruzione per istruzione, con la possibilità però di utilizzare potenti interfacce visuali per l'input e l'output. Inoltre, costituisce la base e l'ambiente di riferimento di **Simulink**, un linguaggio di simulazione totalmente visuale che si integra perfettamente in MATLAB costituendone una vera e propria estensione.

Il linguaggio MATLAB possiede alcune caratteristiche che lo rendono di semplice utilizzo e di grande versatilità. La facilità d'uso è immediatamente percepita dall'utente che può, inserendo semplici comandi, ottenere risultati elementari come quelli di una calcolatrice, o realizzare elaborazioni altamente sofisticate tipiche dei sistemi informatici dotati di grande potenza di calcolo. Vedremo in seguito come il linguaggio in oggetto renda perfettamente coerenti tra loro questi due comportamenti assai differenti. Inoltre, grazie alla sua versatilità, verrà utilizzato in questo corso nell'ambito dell'analisi economica in particolare nello studio dei processi di ottimizzazione e programmazione matematica in ambito statico. Il tutto sarà possibile mediante l'utilizzo di '*utensili informatici*', denominati **ToolBoxes**, che sono stati sviluppati appositamente per varie applicazioni ed estendono notevolmente la potenza del linguaggio.

Questo linguaggio è stato inoltre progettato per interfacciarsi con altri linguaggi di programmazione standard come il C e il **Fortran** o con linguaggi speciali sviluppati, come **Simulink**, per la *simulazione*, o **StateFlow**, per il *progetto degli automi finiti*. L'integrazione tra i linguaggi avviene mediante il supporto dell'ambiente MATLAB che fornisce i collegamenti necessari affinché i vari linguaggi cooperino tra loro in modo coerente ed efficace.

2. Caratteristiche dell'ambiente MATLAB

Il programma MATLAB di MathWorks fu sviluppato originariamente per essere un "**laboratorio delle matrici**" (*MATrix LABoratory*). Il MATLAB di oggi ha capacità che superano di gran lunga quelle del MATLAB originale ed è un sistema interattivo e un linguaggio di

programmazione per calcoli scientifici e tecnici in generale. Il suo elemento base è una matrice. Poiché i comandi di MATLAB sono simili al modo in cui esprimiamo i passaggi di un calcolo in matematica, scrivere soluzioni informatiche in MATLAB è molto più veloce che non scriverle utilizzando un linguaggio di alto livello come il C o il Fortran.

Il **MATLAB** all'interno di uno stesso '*ambiente software*' integra calcoli, visualizzazioni e programmazione al fine di fornire soluzioni a problemi espressi mediante la notazione matematica standard.

Gli usi tipici sono:

- *matematica e calcolo*;
- *sviluppo di algoritmi*;
- *acquisizione dati*;
- *modellizzazione, simulazione e supporto nella creazione di prototipi*;
- *analisi di dati e visualizzazioni*;
- *generazione di grafici scientifici ed ingegneristici*.

MATLAB è un sistema interattivo i cui dati base sono costituiti da *vettori* e *matrici* (array). Questo fatto, come vedremo, consente di scrivere programmi in modo semplice e in tempi decisamente inferiori rispetto a quelli necessari nel caso di utilizzo di linguaggi di tipo scalare come il C ANSI ed il Fortran. Attualmente il '*motore*' di MATLAB incorpora librerie che racchiudono i risultati dell'evoluzione del moderno *calcolo matriciale*.

Oltre ad essere uno strumento standard nell'università, MATLAB viene utilizzato anche nell'industria come strumento di ricerca, sviluppo ed analisi. Mediante numerose applicazioni sviluppate per problemi specifici, i **Toolboxes**, è possibile risolvere particolari classi di problemi connessi a tecnologie specialistiche.

Il sistema MATLAB consiste di cinque parti principali:

1. **Ambiente di sviluppo**: è un insieme di tools ('*utensili*') e di facilitazioni che aiutano nella programmazione e nell'uso delle funzioni e dei file di MATLAB. Molti di questi tools sono di tipo visuale e vengono chiamati *interfacce grafiche di utente* (GUI) e permettono di estendere l'interattività e nel contempo di semplificare enormemente la programmazione. L'ambiente è completato da un insieme di finestre di lavoro chiamato **MATLAB desktop**. Esso comprende il **Command Window** (finestra di lavoro principale), il **Command History** (elenco dei comandi precedenti), l'**Editor** (per scrivere i file) ed il **Debugger** (per correggere gli errori di programmazione), il **Browser** per l'help, ed il **Workspace** che visualizza il contenuto della memoria di lavoro;
2. la **libreria delle funzioni matematiche**: comprende una vasta collezione di funzioni ed algoritmi di calcolo;
3. il **linguaggio MATLAB**: è un linguaggio ad alto livello di tipo vettoriale con istruzioni per il controllo del flusso di elaborazione, per la creazione ed utilizzo di funzioni e strutture dati e per le operazioni di input/output.
4. la **Grafica**: MATLAB permette la visualizzazione bidimensionale e tridimensionale di dati, l'elaborazione ed animazione di immagini in maniera assai semplice;
5. le **interfacce alle applicazioni** (Application Program Interface o API): consentono di scrivere programmi in C ed in Fortran che interagiscono con MATLAB.

Inoltre tutta la documentazione di MATLAB è disponibile in linea mediante un apposito **help browser** che consente di accedere ad esempi applicativi delle funzioni

Il desktop è l'interfaccia principale per lavorare con MATLAB. Quando si attiva l'applicazione, mediante l'icona , appare il desktop, contenente le interfacce grafiche per lavorare con i file, le variabili e le applicazioni associate con MATLAB. La fig. 1 illustra i principali elementi del desktop.

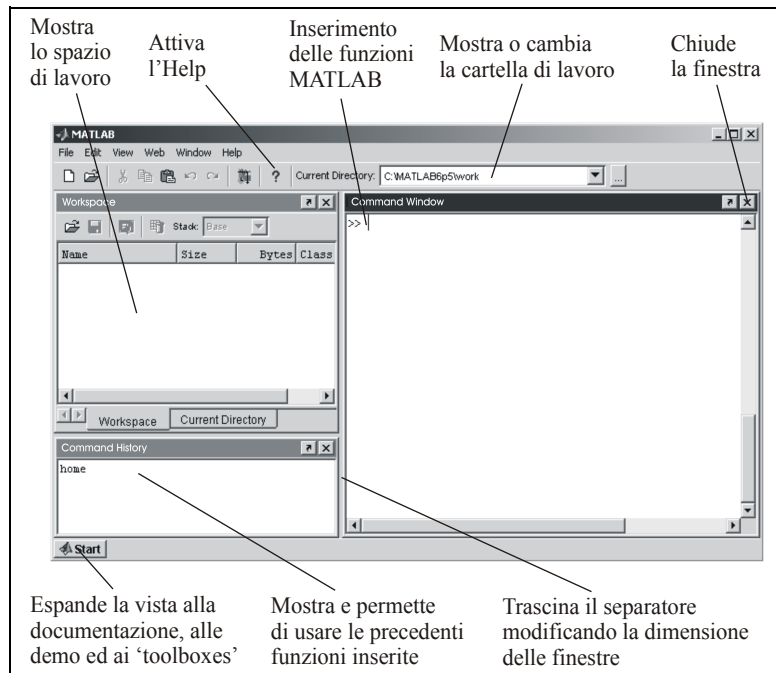


Fig. 1 Principali elementi del desktop.

Esaminiamo più in dettaglio tali elementi:

- la **barra principale**, visualizzata in fig. 2, fornisce un modo rapido di accesso alle operazioni usate più frequentemente;

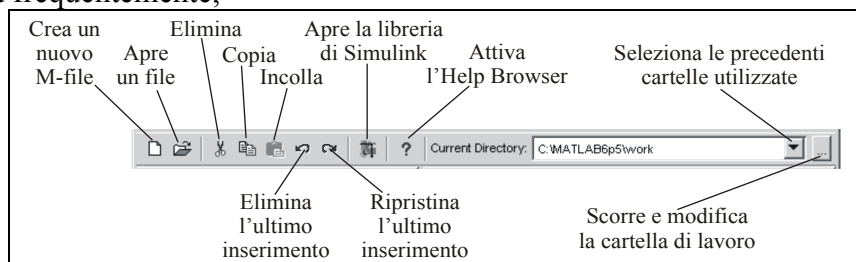


Fig. 2 La barra principale (**Desktop Toolbar**).

- il **Command Window** è usato per inserire le variabili ed eseguire le funzioni ed i cosiddetti M-files (di cui si parlerà successivamente). Nella fig. 3 è mostrata l'esecuzione di una funzione di MATLAB, la ***magic*(n)** che costruisce un quadrato magico di ordine n (i quadrati magici di ordine n , che vengono costruiti con i primi n^2 numeri interi, hanno la proprietà di fornire lo stesso valore sommando gli elementi di una qualsiasi riga o colonna e delle diagonali principale e secondaria);

Nota. È possibile eseguire programmi esterni dalla finestra dei comandi. Il punto esclamativo ! indica che il resto della stringa di input è un comando del sistema operativo.

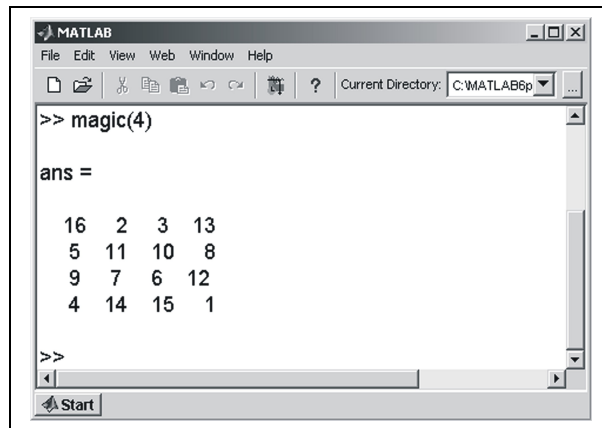


Fig. 3 Costruzione di un quadrato magico di ordine 4 mediante la funzione **magic(4)**.

- il **Command History** è una sorta di contenitore dei comandi precedentemente eseguiti i quali possono essere copiati o eseguiti nuovamente.
- il pulsante di **Start** fornisce un facile accesso agli strumenti, agli esempi (demo) ed alla documentazione (fig. 4);

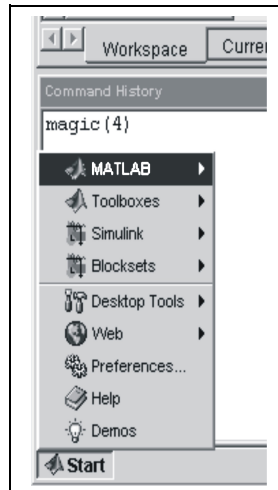


Fig. 4 Espansione del pulsante di start.

- l'**Help Browser** (attivabile con il pulsante **?**) è un vero Web browser, integrato in MATLAB, che serve per ricercare ed esaminare tutta la documentazione e gli esempi (vedi fig. 5).

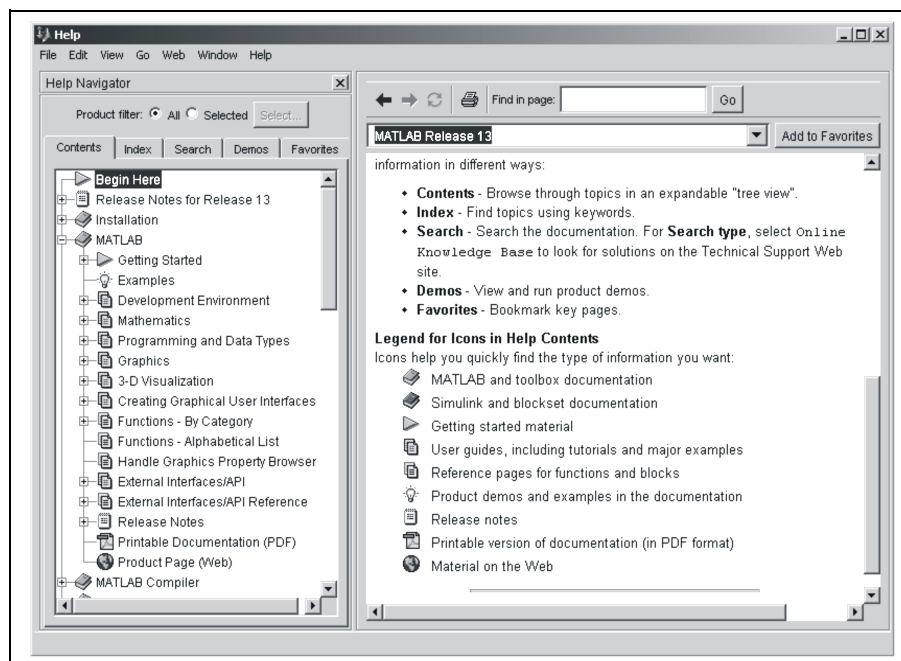


Fig. 5 Help Browser

- lo **Workspace** di MATLAB consiste in un insieme di variabili (denominate array) costruite e memorizzate durante la sessione attiva di lavoro. Di ogni variabile vengono mostrate il nome, il tipo e la dimensione. È possibile aggiungere o cancellare variabili dallo spazio di lavoro. La fig. 6 illustra alcune tipologie di variabili ed i simboli utilizzati nella finestra per rappresentarli;

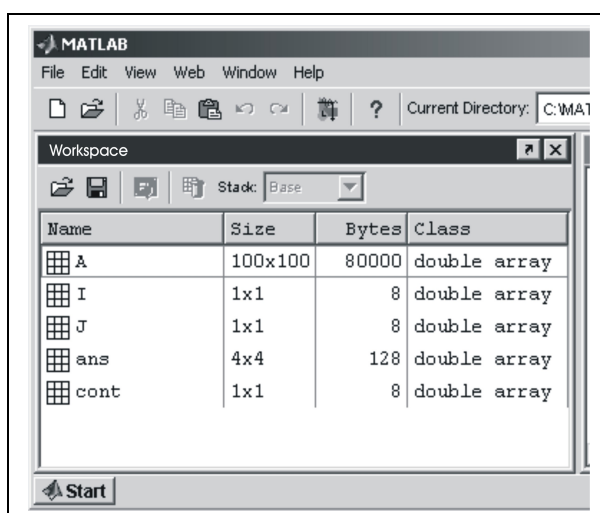


Fig. 6 Uno spazio di lavoro di una generica sessione di MATLAB. Per esempio la matrice **A** di 100 righe per 100 colonne (10000 elementi) occupa 80000 byte di memoria poiché ogni elemento della stessa, essendo 'double', occupa 8 byte.

- l'**Editor/Debugger** consente di creare e correggere, mediante una opportuna interfaccia grafica, gli M-files che sono programmi scritti in linguaggio MATLAB: l'esempio di fig. 7 mostra un M-file che definisce una funzione creata dall'utente. Una volta che la funzione è

stata eseguita, è possibile, ponendo il cursore su una qualsiasi variabile, vedere il valore corrente della variabile stessa;

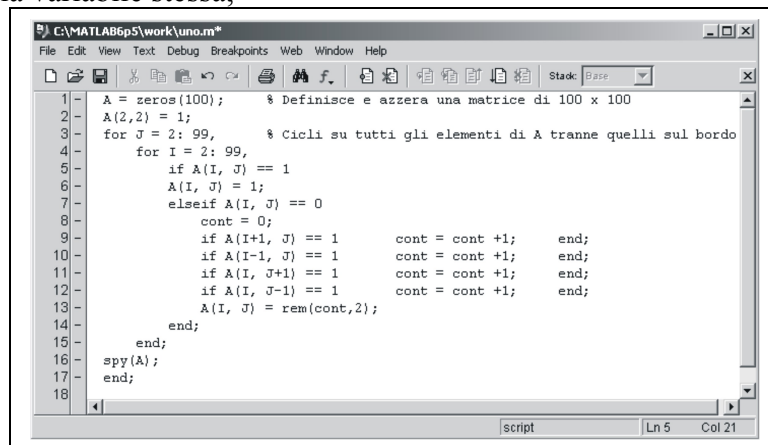


Fig. 7 La finestra di Editor/Debugger.

- il **Profiler** è un'interfaccia grafica che ci permette di testare le prestazioni delle funzioni ed in generale degli M-files, dandoci per esempio i tempi di elaborazione delle singole linee di codice MATLAB inserite nei files stessi.
- l'**importazione** e l'**esportazione di dati**, ovvero, rispettivamente, il recupero di dati creati da altre applicazioni ed il loro inserimento nello spazio di lavoro di MATLAB ed il salvataggio di tutto o di una parte del **Workspace** su file, sono ottenuti attivando direttamente dal menu **File** i comandi **Import Data...** e **Save Workspace As...**

3. Le operazioni fondamentali

Il simbolo **>>** (chiamato prompt), che compare nel **Command Window**, indica che MATLAB è pronto ad accettare un input. Quando lo vediamo apparire possiamo compiere le seguenti operazioni:

- *inserire delle variabili;*
- *far eseguire una funzione.*

Una cosa da notare è che il **programma è sensibile alle maiuscole/minuscole** ed **anche agli spazi** per cui, ad esempio, la funzione **plot**, che visualizza dei grafici, non può essere usata chiamandola **Plot**, mentre le variabili **alfa** ed **Alfa** sono due variabili diverse. Nei paragrafi successivi cominceremo ad esaminare in dettaglio queste operazioni almeno per quanto riguarda le più elementari.

L'inserimento di una **matrice** è il modo migliore per cominciare ad utilizzare MATLAB.

Una matrice è una serie di numeri sistemati su una griglia rettangolare di righe e colonne. Così, un singolo punto può essere considerato una matrice con una sola riga e una sola colonna, una coordinata x-y può essere considerata come una matrice con una riga e due colonne e una serie di quattro coordinate x-y-z può essere considerata come una matrice con quattro righe e tre colonne. Una matrice può essere inserita in vari modi: inserimento di una lista esplicita di elementi, caricamento da un file esterno, generazione diretta mediante una delle funzioni predefinite, creazione da un M-file.

Le convenzioni sintattiche utilizzate sono le seguenti:

- separazione tra elementi di una riga mediante una virgola (,) o uno spazio;
- uso del punto e virgola (;) per indicare la fine di una riga;
- parentesi quadre per racchiudere la definizione della matrice.

Consideriamo alcuni esempi:

$$A = [3.5] \quad B = [1.5 \quad 3.1]$$

$$C = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Notate che i dati della matrice sono scritti tra parentesi quadre. Quando una matrice ha una sola riga e una sola colonna la possiamo anche chiamare scalare. Allo stesso modo, una matrice che ha una sola riga o una sola colonna la possiamo chiamare vettore riga o vettore colonna. Quando utilizziamo una matrice, abbiamo bisogno di un modo per indicare i singoli elementi, o numeri, della matrice. Un metodo semplice per specificare un elemento nella matrice utilizza il numero della riga o della colonna. Per esempio, se ci riferiamo al valore nella riga 4 e colonna 3 della matrice C dell'esempio precedente, non c'è ambiguità: noi ci stiamo riferendo al valore 2. Utilizziamo allora i numeri di riga e colonna come **pedici**; così, C4,3 è uguale a 2. Per indicare l'intera matrice utilizziamo il nome senza alcun pedice, come C, o utilizziamo parentesi quadre attorno a un riferimento a un elemento individuale che utilizza lettere invece dei numeri per i pedici, come in [Ci,j]. Nella notazione matematica formale, il nome della matrice è normalmente in lettere maiuscole, con i riferimenti dei pedici in lettere minuscole, come in C, C4,3 e [Ci,j]. Tuttavia, poiché MATLAB fa distinzione fra maiuscole e minuscole, C e c rappresentano matrici diverse. Perciò, nei vostri programmi MATLAB dovreste utilizzare *coerentemente lettere maiuscole o minuscole nei riferimenti a una specifica matrice*. La dimensione di una matrice è specificata dal numero di righe e colonne. Così, utilizzando il precedente esempio, C è una matrice con quattro righe e tre colonne, o una matrice 4×3. Se una matrice contiene m righe e n colonne, il numero di valori della matrice è il prodotto di m per n; così, C contiene 12 valori. Se una matrice ha lo stesso numero di righe e di colonne, allora viene chiamata **matrice quadrata**.

Nei programmi MATLAB, noi assegniamo nomi agli scalari, ai vettori e alle matrici che utilizziamo. Ai nomi di queste variabili si applicano le seguenti regole.

- I nomi di variabile devono iniziare con una lettera.
- I nomi di variabili possono contenere lettere, numeri e il carattere underscore (_).
- I nomi delle variabili possono avere qualsiasi lunghezza, ma devono essere univoci nei primi 19 caratteri.

Poiché MATLAB discrimina fra lettere minuscole e maiuscole, i nomi Time, TIME e time rappresentano variabili diverse.

Presentiamo ora **quattro metodi** per inizializzare le matrici in MATLAB. Il primo metodo elenca esplicitamente i valori, il secondo legge i dati da un file, il terzo utilizza l'operatore due punti e il quarto legge i dati dalla tastiera.

a) Elenchi espliciti Il metodo più semplice per definire una matrice è quello di utilizzare un elenco di numeri, come mostrato nell'esempio seguente, che definisce le matrici A, B e C che abbiamo utilizzato nell'esempio precedente:

```
A = [3.5];  
B = [1.5, 3.1];  
C = [-1, 0, 0; 1, 1, 0; 1, -1, 0; 0, 0, 2];
```

Queste istruzioni sono esempi di istruzioni di assegnamento, che consistono di un nome di variabile seguito da un segno di uguale e dai valori da assegnare alla variabile stessa. I valori sono racchiusi fra parentesi quadre in ordine di riga. Alcuni punti e virgola separano le righe e i valori delle righe

possono essere separati da virgole o da spazi. Un valore può contenere un segno più o meno e un punto decimale, ma non può contenere una virgola, come in 32,010.

Quando definiamo una matrice, MATLAB stamperà immediatamente i valori sullo schermo, a meno che non sopprimiamo la stampa mettendo un punto e virgola dopo la definizione. Nei nostri esempi, quando state ancora imparando a definire le matrici, è utile vedere i valori della matrice. Quindi, potete voler omettere il punto e virgola dopo la definizione di una matrice fino a quando non siete sicuri che sapete definire le matrici correttamente. I comandi **who** e **whos** sono molto utili quando utilizzate MATLAB. Il comando *who* elenca le matrici che avete definito, mentre il comando *whos* specifica anche le loro dimensioni.

Una matrice può anche essere definita elencando ogni riga su una riga separata, come nella seguente serie di istruzioni MATLAB:

```
C =[-1,0,0  
1,1,0  
1,-1,0  
0,0,2 ];
```

Se ci sono troppi numeri su una riga della matrice per stare su una riga dello schermo, potete continuare l'istruzione sulla riga seguente, ma sono necessari una virgola e tre puntini (un'ellissi) per indicare la continuazione. Per esempio, se vogliamo definire il vettore F con dieci valori, possiamo utilizzare entrambe le istruzioni seguenti:

```
F=[1,52,64,197,42,-42,55,82,22,109 ];  
F=[1,52,64,197,42,-42,...  
55,82,22,109 ];
```

MATLAB vi permette anche di definire una matrice utilizzando un'altra matrice che è già stata definita. Per esempio, considerate le seguenti istruzioni:

```
B=[1.5,3.1 ];  
S=[3.0 B ];
```

Questi comandi sono equivalenti al seguente:

```
S=[3.0 1.5 3.1 ];
```

Va posta anche attenzione al fatto che la sottomatrice inserita in una matrice deve essere un vettore riga o avere lo stesso numero di righe oppure di colonne della matrice in cui si inserisce.

Possiamo anche cambiare i valori in una matrice o aggiungere valori in più utilizzando i riferimenti a posizioni specifiche. Così, il comando

```
S(2)=-1.0;
```

cambia il secondo valore della matrice S da 1.5 a -1.0.

Potete anche estendere una matrice definendo nuovi elementi. Se eseguiamo il comando

```
S(4)=5.5;
```

la matrice S avrà quattro valori invece di tre. Se eseguiamo il comando

$S(8)=9.5;$

allora la matrice S avrà otto valori, e i valori di S(5), S(6) e S(7) sono automaticamente fissati uguali a zero, poiché per essi non è stato fornito alcun valore.

b) File di dati. Le matrici possono anche essere inizializzate a partire da informazioni che sono state memorizzate in un file di dati. MATLAB può leggere due tipi di file di dati: i file MAT e i file ASCII. I file MAT, contengono dati memorizzati in un efficiente metodo binario, mentre un file ASCII contiene le informazioni memorizzate in caratteri ASCII. I file MAT sono preferibili per i dati che devono essere generati e utilizzati solo dai programmi MATLAB, perché l'immagazzinamento è molto efficiente.

I file ASCII sono utili se i dati devono essere condivisi (importati o esportati) con altri programmi diversi da quelli di MATLAB. Un file ASCII che deve essere utilizzato con un programma MATLAB deve avere solo informazioni numeriche e ogni riga del file deve contenere lo stesso numero di valori numerici. Il file può essere generato utilizzando il vostro word processor preferito, ma può essere generato anche utilizzando un programma scritto in un linguaggio come il C. Inoltre, si può creare il file dall'interno di un programma MATLAB utilizzando una delle seguenti forme del comando *save* (dove *<fname>* è il nome del file e *<vlist>* è un elenco delle variabili che si devono salvare).

<i>save <fname> <vlist> -ascii</i>	Utilizza un formato di testo a 8 bit.
<i>save <fname> <vlist> -ascii -double</i>	Utilizza un formato di testo a 16 bit.
<i>save <fname> <vlist> -ascii -double -tabs</i>	Delimita gli elementi con tabulazioni.

Esempio

Per esempio, per creare la matrice Z e salvarla nel file *data_2.dat* in un formato di testo a 8 caratteri, utilizzate i seguenti comandi:

```
Z=[ 5 3 5; 6 2 3 ];
```

```
save data_2.dat Z -ascii;
```

Questi comandi fanno sì che ogni riga della matrice Z sia scritta su una riga diversa del file di dati. L'estensione .mat non viene aggiunta a un file ASCII; tuttavia, come abbiamo mostrato in questo esempio, raccomandiamo che i file ASCII abbiano l'estensione .dat in modo che sia facile distinguerli dai file MAT.

Se guardate il file *data_2.dat* con un word processor come WordPad o Emacs, esso dovrebbe apparire come:

```
5.0000000e+000 3.0000000e+000 5.0000000e+000
6.0000000e+000 2.0000000e+000 3.0000000e+000
```

Supponiamo che un file ASCII chiamato *data_3.dat* contenga una serie di valori che rappresentano il tempo e le corrispondenti distanze di un corridore dalla linea di partenza di una gara. Ogni tempo e il suo corrispondente valore di distanza sono su una riga separata del file di dati. Così, le prime poche righe del file potrebbero avere la forma seguente:

```
0.0 0.0
0.1 3.5
0.2 6.8
```

Il comando `load` seguito dal nome del file leggerà le informazioni in una matrice che ha lo stesso nome del file.

Esempio

Per esempio, le istruzioni seguenti leggeranno i dati dal file `data_3.dat` e li immetteranno nella matrice `data_3`, che ha tre righe e due colonne. Potete poi digitare il nome della variabile per vederne il contenuto.

```
load data_3.dat
```

```
data3
```

Dovrebbe apparire sullo schermo il seguente output:

```
data_3 =  
0 0  
0.1000 3.5000  
0.2000 6.8000
```

c) Operatore due punti L'operatore due punti è un operatore molto potente per creare nuove matrici. Per esempio, l'operatore due punti può essere utilizzato per creare vettori a partire da una matrice. Quando un segno di due punti viene utilizzato in un riferimento di una matrice al posto di uno specifico indice, esso rappresenta l'intera riga o l'intera colonna. Per esempio, utilizzando la matrice `data_3` che è stata letta dal file nella trattazione precedente, i seguenti comandi memorizzeranno la prima colonna di `data_3` nel vettore colonna `x` e la seconda nel vettore colonna `y`:

```
x=data_3(:,1);  
y=data_3(:,2);
```

L'operatore due punti può anche essere utilizzato per generare nuove matrici. Se un segno di due punti viene utilizzato per separare due interi, l'operatore genera tutti gli interi tra due specificati.

Per esempio, la notazione seguente genera un vettore chiamato `H` che contiene i numeri da 1 a 8:

```
H=1:8;
```

Se i due punti sono utilizzati per separare tre numeri, allora l'operatore due punti genera i valori tra il primo e il terzo numero, utilizzando il secondo come incremento.

Per esempio, la notazione seguente genera un vettore riga di nome *time* che contiene i numeri da 0.0 a 5.0, con un incremento pari a 0.5:

```
time = 0.0:0.5:5.0;
```

L'incremento può anche essere negativo, come mostrato nell'esempio seguente, che genera i numeri 10, 9, 8,...0 nel vettore riga *values*:

```
values =10:-1:0;
```

L'operatore due punti può anche essere utilizzato per selezionare una sottomatrice da un'altra matrice.

Per esempio, supponendo che `C` sia la matrice seguente:

$$C = \begin{bmatrix} -1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & 2 \end{bmatrix}$$

Se eseguiamo i seguenti comandi:

```
C_partial_1 = C(:,2:3);
C_partial_2 = C(3:4,1:2);
```

abbiamo definito le matrici seguenti:

$$C_partial_1 = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ -1 & 0 \\ 0 & 2 \end{bmatrix} \quad C_partial_2 = \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix}$$

Se la notazione due punti definisce una matrice con un indice non valido, come in C(5:6,:) allora viene mostrato un messaggio di errore.

In MATLAB è valido anche avere una matrice vuota. Per esempio, le istruzioni seguenti generano una **matrice vuota**:

```
a = [];
b = 4:-1:5;
```

Notate che una matrice vuota è diversa da una matrice che contiene soltanto il valore zero.

L'utilizzo dell'espressione C(:) è equivalente a una lunga matrice colonna che contiene la prima colonna di C seguita dalla seconda colonna e così via, fino alla fine della matrice.

Un operatore che è molto utile con le matrici è l'operatore **trasposizione**. La matrice trasposta della matrice A viene indicata con A' e rappresenta una nuova matrice in cui le righe di A diventano le colonne di A'.

Per esempio, supponiamo di aver generato due vettori x e y e di voler stampare i valori come x(1) e y(1) sulla stessa riga, x(2) e y(2) su un'altra riga, e così via. Un metodo semplice per fare questo è:

```
x = 0:4;
y = 5:5:25;
[x' y']
```

L'output generato da queste istruzioni è:

```
0 5
1 10
2 15
3 20
4 25
```

d) Input dell'utente I valori di una matrice possono anche essere immessi mediante la tastiera utilizzando il comando `input` che mostra una stringa di testo e poi attende un valore. Il valore immesso dall'utente viene memorizzato nella variabile specificata. Se vengono immessi più valori, essi devono essere racchiusi fra parentesi quadre. Se l'utente preme INVIO senza aver immesso alcun valore, allora viene generata una matrice vuota. Se il comando non finisce con un punto e virgola, allora i valori immessi vengono immediatamente stampati di nuovo.

Considerate il seguente comando:

```
z=input('Inserire i valori per z tra parentesi:');
```

Quando viene eseguito questo comando, viene mostrata sullo schermo del terminale la stringa di testo *Inserire i valori per z tra parentesi:*. L'utente può allora immettere un'espressione come `[5.1 6.3 -18.0]` che specifica i valori per `z`. Poiché questo comando `input` termina con un punto e virgola, il valore di `z` non viene stampato al termine della sua esecuzione.

4. La stampa di matrici

Ci sono molti modi per stampare il contenuto di una matrice. Il metodo più semplice è quello di immettere il nome della matrice alla riga dei comandi. Il nome verrà immediatamente ripetuto e i valori saranno stampati a iniziare dalla riga seguente. Ci sono anche numerosi altri comandi che possono essere utilizzati per stampare le matrici con un maggiore controllo sulla forma dell'output.

Formato display. Quando vengono stampati gli elementi di una matrice, i numeri interi sono sempre stampati come interi. I valori con una parte decimale vengono stampati utilizzando un formato di default (chiamato anche formato corto) che mostra quattro decimali. MATLAB vi permette di specificare formati diversi (vedere la Tabella 1), che mostrano più cifre significative. Per esempio, per specificare che vogliamo che i valori siano stampati con 14 cifre decimali, possiamo utilizzare il comando *format long*. Il formato può essere ripristinato con il comando *format short*. Quando invece viene utilizzato il comando *format bank* allora MATLAB mostra solo due cifre decimali.

Comando MATLAB	Display	Esempio
<code>format short</code>	default	15.2345
<code>format long</code>	14 decimali	15.23453333333333
<code>format short e</code>	4 decimali	1.5235e+01
<code>format long e</code>	15 decimali	1.523453333333333e+01
<code>format bank</code>	2 decimali	15.23
<code>format +</code>	+, -, spazio vuoto	+

Tabella 1

Quando un valore è molto piccolo o molto grande, la notazione decimale non è sempre soddisfacente. Per esempio, consideriamo il valore 602.300.000.000.000.000.000. Ovviamente, abbiamo bisogno di una notazione più maneggevole per valori molto grandi come questo numero o molto piccoli come 0.0000000031. La **notazione scientifica** esprime un valore come un numero fra 1 e 10 moltiplicato per una potenza di 10. Nella notazione scientifica, il valore precedente diventa 6.023×10^{23} . In questa forma il valore può essere diviso in una mantissa (6.023) e un esponente (23). In MATLAB, i valori nella notazione scientifica vengono stampati con la lettera *e* che separa la mantissa e l'esponente, come in 6.023e+23. Se vogliamo che MATLAB stampi i numeri in notazione scientifica con 4 cifre decimali, possiamo utilizzare il comando *format short e*. Per

specificare una notazione scientifica con 16 cifre significative, utilizziamo il comando *format long e*. Possiamo anche inserire un valore in una matrice utilizzando la notazione scientifica, ma è importante omettere gli spazi tra la mantissa e l'esponente, perché MATLAB interpreterebbe '6.023 e+23' come due valori (6.023 ed e+23), mentre 6.023e+23 verrebbe interpretato correttamente.

Un altro comando di formato è *format +*. Quando una matrice viene stampata con questo valore, di ogni valore vengono visualizzati solo i segni più o meno. Se un valore è positivo, allora viene stampato un segno più, se il valore è zero viene stampato uno spazio e se è negativo viene stampato un segno meno. Questo formato permette di vedere una grande matrice in termini dei suoi segni.

Diversamente, non saremmo in grado di vederla facilmente, perché su una riga ci potrebbero essere troppi valori per stare su una sola riga.

Per formati lunghi e corti, viene applicato all'intera matrice un comune fattore di scala, se gli elementi diventano molto grandi. Questo fattore di scala viene stampato insieme ai valori scalati.

Infine, il comando *format compact* sopprime molti dei ritorni a capo che appaiono fra le matrici e permette di vedere sullo schermo più righe di informazione insieme. Il comando *format loose* serve a ripristinare il metodo di stampa meno compatto.

Esempio. Il codice sotto indicato definisce mediante la funzione **format** il tipo di rappresentazione dei numeri da utilizzare. In particolare il formato short indica la rappresentazione del numero con quattro cifre decimali. Inoltre crea le variabili vettoriali **x** di 10 elementi e **logs** che contiene il valore di **x** e del suo logaritmo corrispondente ottenuto con la funzione **log**.

```
>> format short; x=(1:10)'; logs=[x log(x)]
```

logs =

1.0000	0
2.0000	0.6931
3.0000	1.0986
4.0000	1.3863
5.0000	1.6094
6.0000	1.7918
7.0000	1.9459
8.0000	2.0794
9.0000	2.1972
10.0000	2.3026

Spiegazione:

- la variabile **x=(1:10)'** definisce una matrice di una sola colonna (denominata anche vettore colonna) i cui elementi vanno dal valore 1 al valore 10 con incremento di 1; la variabile **logs=[x log(x)]** è una matrice che ha due colonne. La prima è la **x**, la seconda il valore della funzione logaritmica **log(x)** per i valori corrispondenti.

Stampare testi e matrici Il comando **disp** può essere utilizzato per stampare il testo racchiuso fra apici singoli. Può anche essere utilizzato per stampare il contenuto di una matrice senza stampare il nome della matrice stessa. Così, se uno scalare di nome **temp** contiene dei valori di temperatura in gradi Celsius, potremmo stampare il valore su una riga e l'unità di misura sulla successiva con questi comandi:

```
disp(temp); disp('gradi Celsius')
```

Se il valore di **temp** è 78, l'output sarà il seguente:

78

gradi Celsius

Notate che i due comandi `disp` sono stati immessi sulla stessa riga, in modo tale da farli eseguire insieme.

Output formattato Il comando `fprintf` dà un maggiore controllo sull'output di quanto non faccia il comando `disp`. Oltre a stampare sia il testo che i valori delle matrici, potete anche specificare il formato da utilizzare per stampare i valori e potete esplicitare le andate a capo. Se siete dei programmatori C, vi troverete a vostro agio con la sintassi di questo comando. Con poche eccezioni, infatti, il comando `fprintf` di MATLAB utilizza le stesse specifiche di formato della funzione C `fprintf()`. Verranno descritte solo alcune opzioni di formato.

La forma generale di questo comando è:

`fprintf(format-string,var, . . .)`

La stringa di formato contiene le specifiche sulla formattazione dell'output. La stringa può contenere anche del testo letterale e delle specifiche di formato che definiscono come devono essere stampati i valori numerici. Il resto degli argomenti è un elenco di nomi delle matrici da stampare. Gli specificatori di formato hanno la seguente forma:

`%[flags] [width] [.precision] type` (Gli spazi e le parentesi non fanno parte della sintassi)

Ogni specificatore inizia con il carattere `%`. I parametri `flags`, `width` e `precision` sono facoltativi. Esempi di **campi di tipo** sono invece:

`%e` stampa in formato esponenziale (notazione scientifica)

`%f` stampa in virgola fissa

`%g` stampa utilizzando `%e` o `%f`, a seconda di quale è più corto

Le stringhe vengono mostrate letteralmente. Il carattere `%` può essere stampato utilizzando `%%`. Le tabulazioni e le andate a capo si possono ottenere rispettivamente con `\t` e `\n`.

Esempio

Come esempio, immettete i seguenti comandi:

```
temp = 98.6;  
fprintf('La temperatura è %f gradi F.\n', temp);
```

Dovreste ottenere il seguente output:

La temperatura è 98.600000 gradi F.

Il parametro *width* controlla il minimo numero di caratteri stampati e deve essere un intero non negativo. Se i caratteri complessivi sono in numero inferiore al parametro *width* vengono anteposti tanti spazi quanti necessitano a stampare complessivamente il numero di caratteri individuato dal parametro *width*. Il parametro *precision* è preceduto da un punto e specifica il numero di cifre decimali da stampare dopo il punto, per i tipi esponenziale e a virgola fissa. Notate nel precedente esempio che la precisione di default è di sei cifre. Il campo *precision* ha significati diversi con altri tipi di dati.

Esempio

Per esempio, immettete la seguente istruzione:

```
fprintf('La temperatura è %8.1f gradi .\n', temp);
```

Otterrete questo output:

La temperatura è 98.6 gradi. (ci sono 4 spazi davanti a 98.6 oltre quello immediatamente dopo è)

Il campo flag viene utilizzato per determinare i caratteri di riempimento e il formato del segno. Un parametro zero (0) con un valore in virgola fissa indica che i caratteri iniziali verranno riempiti con zeri, invece che con spazi. Un flag – allinea a sinistra l'argomento convertito nel suo campo e un flag + fa sempre stampare il segno del valore (+ o –). Si possono inserire anche più flag uno di seguito all'altro.

Esempio

Per esempio, immettete la seguente istruzione:

```
fprintf('La temperatura è %08.1f gradi .\n', temp);
```

Otterrete questo output:

La temperatura è 000098.6 gradi.