

II. Introduzione alla programmazione in MATLAB

1. Le espressioni

In generale le variabili possono essere definite mediante una espressione. I blocchi costitutivi delle espressioni sono:

- **variabili;**
- **numeri;**
- **operatori;**
- **funzioni.**

Le **variabili** non richiedono nessun tipo di dichiarazione o dimensionamento come avviene ad esempio nel linguaggio C. Quando MATLAB incontra un nuovo nome di variabile, automaticamente alloca l'appropriata quantità di memoria; se la variabile già esiste, il suo contenuto viene riscritto con il nuovo valore. I nomi delle variabili devono cominciare con una lettera e possono contenere fino a 31 caratteri.

I **numeri** usano la notazione decimale convenzionale anglosassone: il punto separa le cifre decimali da quelle intere. La notazione scientifica specifica dopo la lettera **e** il numero del fattore di potenza del dieci da moltiplicare. I numeri immaginari usano il suffisso **i** o **j**. Qualche esempio di numeri legali:

3	-99	0.0001
9.6397238	1.60210e-20	02252e23
5i	-3.14159j	3 - 4e2i

Tutti i numeri sono memorizzati secondo il formato IEEE floating-point standard con 16 cifre significative nel range tra 10^{-308} e 10^{+308} .

Gli **operatori** sono quelli utilizzati normalmente nei linguaggi di programmazione. In tabella 1 ve ne sono riportati alcuni, che saranno esaminati in dettaglio nei prossimi paragrafi.

+	Addizione
-	Sottrazione
*	Moltiplicazione
/	Divisione
^	Potenza

Tab. 1 Alcuni operatori matematici

Le **funzioni** sono quelle fornite da MATLAB e comprendono sia le funzioni matematiche elementari (valore assoluto, radice quadrata, esponenziale, ecc.),

<code>abs(x)</code>	Absolute value $ x $
<code>sign(x)</code>	Sign, returns -1 if $x < 0$, 0 if $x = 0$, 1 if $x > 0$
<code>exp(x)</code>	Exponential e^x
<code>log(x)</code>	Natural logarithm $\ln x$
<code>log10(x)</code>	Common (base 10) logarithm $\log_{10} x$
<code>sqrt(x)</code>	Square root $\sqrt{x}$
<code>rem(x,y)</code>	Remainder of x/y . For example, <code>rem(100,21)</code> is 16. Also called the modulus function.

sia quelle della matematica avanzata (funzioni Gamma, di Bessel, ecc.). Mediante il comando **help elfun** è possibile visualizzare l'elenco delle funzioni matematiche elementari, mentre con il comando **help specfun** sono elencate quelle avanzate. Alcune costanti particolari sono definite direttamente in MATLAB:

- **pi** 3.14159265...
- **i o j** unità immaginaria
- **realmin** numero più piccolo rappresentabile in memoria
- **realmax** numero più grande rappresentabile in memoria
- **Inf** infinito
- **NaN** non è un numero

Esempi di espressioni:

```
>> x = 1 / -sqrt(4^2+4)      % Usa la funzione radice quadrata
x =
-0.2236

>> round(abs(-3/4)) % La funzione round arrotonda il risultato all'intero più
                    % vicino: 0.75 viene arrotondato a 1
ans =
1
```

2. Gli operatori

Gli **operatori** rivestono un ruolo molto importante come elementi fondamentali di ogni linguaggio di programmazione. È infatti dalla quantità e qualità degli operatori definiti nel linguaggio che noi possiamo valutarne la 'potenza' in termini di efficienza e di comodità nell'uso del linguaggio stesso.

L'insieme degli operatori presenti in MATLAB è veramente completo e permette di agire anche su dati complessi come i vettori e le matrici. Gli operatori di MATLAB si suddividono in tre categorie:

- **Operatori aritmetici:** realizzano i calcoli numerici, per esempio addizionando due numeri od elevando una variabile al quadrato;
- **Operatori relazionali:** comparano operandi dal punto di vista quantitativo, utilizzando costrutti di tipo "minore di" o "non uguale a".
- **Operatori logici:** usano gli operatori *AND*, *OR* e *NOT*.

Inoltre, tutti gli operatori, che possono essere combinati tra loro, sottostanno a delle regole denominate

- **Precedenze degli operatori:** indicano la priorità degli operatori nell'eseguire le operazioni.

- Operatori aritmetici

Dato che MATLAB è un linguaggio a carattere vettoriale gli operatori aritmetici sono concepiti per agire sulle matrici ed anche sulle variabili scalari, che è sempre possibile considerare come matrici di una riga e una colonna.

Tali operatori agiscono sulle matrici sia secondo le regole dell'algebra lineare (o algebra delle matrici) sia elemento per elemento. Questo secondo tipo di funzionamento viene indicato premettendo il punto (.) all'operatore.

La tab. 2 mostra l'elenco completo degli operatori aritmetici.

Operatore	Descrizione
+	Addizione
-	Sottrazione
.*	Moltiplicazione
./	Divisione destra
.\	Divisione sinistra
+	Più unario
-	Meno unario
:	Operatore due punti
.^	Potenza
.'	Trasposta
,	Complessa coniugata della trasposta
*	Moltiplicazione di matrice
/	Divisione destra di matrice
\	Divisione sinistra di matrice
^	Potenza di matrice

Tab. 2 Tabella degli operatori aritmetici.

Descrizione degli operatori

Abbiamo già visto che in generale gli operatori aritmetici agiscono su matrici (es. **A** e **B**). La sintassi è

A + B

A - B

A * B A .* B

A / B A ./ B

A \ B A .\ B

A ^ B A .^ B

A' A .'

Descriviamo in dettaglio il funzionamento degli operatori:

- +** **Addizione o più unario:** A+B addiziona A e B che devono avere la stessa dimensione (ovvero numero di righe e colonne) a meno che una delle due sia uno scalare. Uno scalare può essere sempre sommato a una matrice di qualsiasi dimensione.
- **Sottrazione o meno unario:** A-B sottrae B da A. A e B devono avere la stessa dimensione a meno che una delle due sia uno scalare.
- *** **Moltiplicazione matriciale:** $C = A*B$ è il prodotto lineare algebrico delle matrici A e B. Più precisamente

$$C(i, j) = \sum_{k=1}^n A(i, k)B(k, j)$$

Il numero di colonne di A deve essere uguale al numero di righe di B.

Uno scalare può essere sempre moltiplicato a una matrice di qualsiasi dimensione.

ESEMPIO 1

Costruiamo due matrici A e B di dimensioni diverse mediante il semplice assegnamento e moltiplichiamole tra loro per mezzo dell'operatore moltiplicazione matriciale. Per inciso, il numero di colonne di A deve essere uguale al numero di righe di B e la matrice prodotto possiede lo stesso numero di righe di A e lo stesso numero di colonne di B.

```
>> A = [1 2 3 4; 5 6 7 8]          % Matrice A(2,4): 2 righe e 4 colonne.
A =
     1     2     3     4
     5     6     7     8

>> B = [1 1 4; 2 2 3; 3 3 2; 4 4 1] % Matrice B(4,3): 4 righe e 3 colonne.
B =
     1     1     4
     2     2     3
     3     3     2
     4     4     1

>> C = A * B                      % Matrice C(2,3): 2 righe e 3 colonne.
C =
    30    30    20
    70    70    60
```

ESEMPIO 2

Questo esempio mostra come moltiplicare tutti gli elementi della matrice A per una costante uguale a 3.

```
>> A = [1 2 3 4; 5 6 7 8];
>> A * 3 % Ogni elemento di A è moltiplicato per 3.
ans =
     3     6     9    12
    15    18    21    24
```

.* ***Moltiplicazione scalare:*** A .* B è il prodotto elemento per elemento di A e B. In altre parole, vengono moltiplicati gli elementi corrispondenti delle due matrici ovvero $A(i,j) * B(i,j)$; si deduce che A e B devono avere la stessa dimensione a meno che una delle due sia uno scalare.

ESEMPIO 11

```
>> A = [1 2 3 4; 5 6 7 8]          % Definizione matrice A.
A =
     1     2     3     4
```

```

5   6   7   8
>> B = [-2 -2 1 1; 1 1 -1 -1]    % Definizione matrice B.
B =
    -2    -2     1     1
     1     1    -1    -1

>> A .* B    % Vengono moltiplicati gli elementi corrispondenti di A e di B.
ans =
    -2    -4     3     4
     5     6    -7    -8

```

/ **Divisione matriciale destra:** Se A è una matrice quadrata non singolare ($\det(A) \neq 0$ ovvero A è invertibile), B/A è la stessa operazione indicata nell'algebra lineare $B * \mathbf{inv}(A)$ (**inv** significa matrice inversa). Più precisamente $B/A = (A.' \setminus B.')'$. Vedi \. Nel caso in cui A sia quadrata ma singolare B/A è la matrice con le stesse dimensioni di B i cui elementi sono tutti uguali a Inf.

./ **Divisione scalare destra:** $B ./ A$ è la matrice con elementi $B(i,j) / A(i,j)$, dove all'espressione $0/0$ viene restituito il valore Nan e a quella $n/0$, con $n \neq 0$, il valore Inf.

\ **Divisione matriciale sinistra:** Se A è una matrice quadrata non singolare, $A \setminus B$ è la stessa che si ottiene facendo $\mathbf{inv}(A) * B$, eccetto per il modo in cui il calcolatore effettua l'elaborazione. Se A è una matrice n per n e B è un vettore colonna di n componenti, allora $X = A \setminus B$ è la soluzione dell'equazione matriciale $AX = B$ (ovvero del sistema lineare associato) calcolata con il metodo della eliminazione Gaussiana. Nel caso in cui A sia quadrata ma singolare $A \setminus B$ è la matrice con le stesse dimensioni di B i cui elementi sono tutti uguali a Inf.

ESEMPIO 3

Si vuole risolvere un sistema lineare di quattro equazioni in quattro incognite. Detta A la matrice dei coefficienti che contiene i coefficienti che moltiplicano le incognite del sistema, X il vettore colonna delle incognite e B il vettore colonna contenente i termini noti delle equazioni del sistema, è possibile rappresentare il sistema stesso come $A * X = B$. Prendiamo ad esempio il seguente sistema:

$$\begin{cases} 2x_1 + 3x_2 - 12x_3 - 7x_4 = 0 \\ -x_1 + 2x_2 + 10x_3 - 2x_4 = -3 \\ 4x_1 - 3x_2 + 5x_3 - 4x_4 = 10 \\ x_1 + x_2 + 9x_3 - 4x_4 = 6 \end{cases}$$

Possiamo determinare la soluzione (se il sistema l'ammette) utilizzando proprio la divisione matriciale sinistra.

```

>> A = [2 3 -12 -7;-1 2 10 -2; 4 -3 5 -4; 1 1 9 -4]    % Matrice dei coefficienti.
A =
     2     3    -12    -7
    -1     2    10    -2
     4    -3     5    -4
     1     1     9    -4

>> B = [0 -3 10 6]' % Vettore colonna dei termini noti.

```

```

B =
    0
   -3
   10
    6

>> X = A \ B % Vettore soluzione del sistema
X =
   27.3469
   18.0714
    1.4388
   13.0918

>> A * X      % Verifica della soluzione: si ottiene esattamente il termine noto.
ans =
   -0.0000
   -3.0000
   10.0000
    6.0000

```

. ***Divisione scalare sinistra:*** $A \setminus B$ è la matrice con elementi $B(i,j) / A(i,j)$, dove all'espressione $0/0$ viene restituito il valore Nan e a quella $n/0$, con $n \neq 0$, il valore Inf.

^ ***Potenza di matrice:*** Se X è una matrice quadrata, X^p è X alla p, se p è uno scalare.

.^ ***Potenza scalare di matrice:*** $A.^B$ è la matrice con elementi $A(i,j)$ alla $B(i,j)$. Ovvero $A(i,j)^{B(i,j)}$. A e B devono avere la stessa dimensione a meno che una delle due sia uno scalare.

' ***Matrice trasposta:*** A' è la trasposta algebrica di A, ovvero la matrice che ha come righe le colonne di A e come colonne le righe di A. Per matrici complesse diventa la trasposta della complessa coniugata.

.' ***Matrice trasposta senza coniugazione:*** A.' è la trasposta algebrica di A. Per matrici complesse non coinvolge la coniugazione.

- Operatori relazionali

Gli operatori relazionali, indicati in tab. 3, effettuano in generale una comparazione elemento per elemento tra matrici. Essi restituiscono una matrice logica con gli elementi posti a 1 se la relazione tra elementi è vera, viceversa posti a 0 se la relazione è falsa.

Operatore	Descrizione
<	Minore di
<=	Minore o uguale di
>	Maggiore di
>=	Maggiore o uguale di
==	Uguale a
~=	Non uguale a

Tab. 3 Tabella degli operatori relazionali.

La sintassi è alquanto semplice:

```
A < B
A <= B
A > B
A >= B
A == B
A ~= B
```

Nota. Gli operatori <, >, <=, e >= usano solo la parte reale dei loro operandi per testare il confronto. Gli operatori == e ~= testano sia la parte reale che quella immaginaria.

ESEMPIO 4

Le seguenti istruzioni producono una matrice logica

```
X = 5; X >= [1 2 3; 4 5 6; 7 8 10]
ans =
     1     1     1
     1     1     0
     0     0     0
```

La variabile scalare X viene confrontata con tutti gli elementi della matrice [1 2 3; 4 5 6; 7 8 10]. Il confronto genera la matrice logica che contiene gli 1 dove il test dà risultato Vero, mentre contiene gli 0 nel caso di risultato Falso.

- Operatori logici

MATLAB dispone di tre tipi di operatori logici che agiscono su:

- elementi di vettori logici;
- bits corrispondenti di numeri interi o vettori;
- scalari o espressioni logiche.

Assumono la simbologia di tab. 4:

Simbolo	Operatore
&	AND
	OR
~	NOT
xor	XOR

Tab. 4 Tabella degli operatori logici.

Attivando l'Help di MATLAB è possibile recuperare tutte le informazioni per l'utilizzo nei vari casi previsti.

L'esempio seguente considera due vettori sui quali vengono applicati i diversi operatori che agiscono sugli elementi corrispondenti dei due vettori:

```
>> A = [0 1 1 0 1]; B = [1 1 0 0 1];
>> A & B % And tra i vettori A e B
ans =     0     1     0     0     1
```

```

>> A | B                % Or tra i vettori A e B
ans = 1 1 1 0 1
>> ~A                   % Not A.
ans = 1 0 0 1 0

>> xor(A,B)             % Xor tra i vettori A e B
ans = 1 0 1 0 0

```

Come si nota, il funzionamento è lo stesso che si avrebbe se ogni elemento dei due vettori fosse un singolo bit di un numero binario.

- Precedenze degli operatori

La tab. 5 mostra le precedenze degli operatori quando vengono utilizzati per costruire le espressioni. I livelli di precedenza determinano in quale ordine MATLAB deve valutare le combinazioni di operatori. A parità di livello la valutazione viene fatta da sinistra verso destra.

Parentesi ()
Trasposta (.'), potenza (.^), complessa coniugata della trasposta ('), potenza di matrice (^)
Segno più unario (+), segno meno unario (-), negazione logica (~)
Moltiplicazione (.*), divisione destra (./), divisione sinistra (.\), moltiplicazione di matrici (*), divisione destra di matrici (/), divisione sinistra di matrici (\)
Addizione (+), sottrazione (-)
Operatore due punti (:)
Minore di (<), minore o uguale di (<=), maggiore di (>), maggiore o uguale di (>=), uguale a (==), non uguale a (~=)
Element-wise AND (&)
Element-wise OR ()
Short-circuit AND (&&)
Short-circuit OR ()

Tab. 5 Tabella delle precedenze degli operatori.

3. Operazioni con gli scalari e le matrici

I calcoli aritmetici sono specificati da matrici e costanti combinati con le operazioni aritmetiche. In questo paragrafo, tratteremo prima le operazioni che coinvolgono solo gli scalari e poi le estenderemo alle operazioni elemento dopo elemento.

Operazioni sugli scalari

Le operazioni aritmetiche fra due scalari sono mostrate nella Tabella 6. Le espressioni contenenti scalari e le operazioni scalari possono essere valutate e memorizzate in una variabile, come nella seguente istruzione, che specifica che i valori di a e b devono essere sommati, e la somma memorizzata in x :

```
x = a + b;
```

Questa istruzione di assegnamento dovrebbe voler dire come una precisazione che il valore in a deve essere sommato al valore in b e la somma memorizzata in x . Se interpretate l'istruzione di assegnamento in questo modo, non dovrete trovare strana la seguente istruzione MATLAB:

```
count =count +1;
```

Chiaramente questa istruzione non è una espressione algebrica valida, ma in MATLAB dice che 1 deve essere sommato al valore in count e il risultato deve essere di nuovo memorizzato in count . Perciò, è equivalente a specificare che il valore in count deve essere incrementato di uno.

È importante capire che una variabile può memorizzare un solo valore per volta. Per esempio, supponiamo di eseguire in sequenza le due istruzioni MATLAB seguenti:

```
time =0.0;  
time =5.0;
```

Il valore 0.0 viene memorizzato nella variabile time quando viene eseguita la prima istruzione, e poi rimpiazzato da 0.5 quando viene eseguita la seconda.

Quando immettete un'espressione *senza specificare la variabile a cui assegnare il risultato*, esso viene automaticamente memorizzato in una variabile chiamata **ans**. Ogni volta che viene memorizzato un nuovo valore in **ans** il *precedente viene perso*.

Operazione	Forma algebrica	MATLAB
Addizione	$a + b$	$a + b$
Sottrazione	$a - b$	$a - b$
Moltiplicazione	$a \times b$	$a * b$
Divisione	$\frac{a}{b}$	a / b
Elevamento a potenza	a^b	$a ^b$

Tab. 6 Operazioni aritmetiche fra due scalari

Come introdotto in precedenza, numerose operazioni possono essere combinate in una singola espressione aritmetica, di conseguenza è importante conoscere in quale ordine le operazioni verranno svolte. È possibile schematizzare l'ordine di precedenza degli operatori aritmetici in MATLAB nella seguente maniera:

Precedenza Operazione

- 1 Parentesi, prima le più interne.
- 2 Elevamento a potenza, da sinistra a destra.
- 3 Moltiplicazione e divisione, da sinistra a destra.
- 4 Addizione e sottrazione, da sinistra a destra.

Notate che questa precedenza segue la normale precedenza algebrica.

Supponiamo di voler calcolare l'area di un trapezio, e che le variabili *base_1* e *base_2* contengano la lunghezza della base maggiore e minore e che *height* contenga quella dell'altezza. L'area del trapezoide può essere calcolata utilizzando le seguenti istruzioni MATLAB:

```
area =0.5*height*(base_1 + base_2);
```

Supponiamo ora di omettere le parentesi nell'espressione:

```
area =0.5*height * base_1 + base_2;
```

Questa istruzione verrebbe eseguita come se fosse:

```
area =(0.5*height * base_1) +base_2;
```

Notate che sebbene sia stata calcolata la risposta sbagliata, non ci sono messaggi di errore che ci avvertano. Perciò, è importante prestare molta attenzione quando si convertono le equazioni in istruzioni di MATLAB. Aggiungere parentesi in più è un modo molto sicuro per assicurarsi che i calcoli vengano svolti nell'ordine voluto. Se un'espressione è lunga, rompetela in più istruzioni. Per esempio, considerate la seguente equazione:

$$f = \frac{x^3 - 2x^2 + x - 6.3}{x^2 + 0.05005x - 3.14}$$

Il valore di f potrebbe essere calcolato utilizzando le istruzioni MATLAB:

```
numerator = x^3 - 2*x^2 + x - 6.3;  
denominator = x^2 + 0.05005*x - 3.14;  
f = numerator/denominator;
```

È meglio utilizzare numerose istruzioni che sono facili da capire, piuttosto che una sola che richieda un'attenta considerazione per capire l'ordine delle operazioni.

Limitazioni di calcolo

Le variabili memorizzate in un computer possono assumere valori entro un grande intervallo. Per la maggior parte dei computer, l'intervallo dei valori si estende da 10^{-308} a 10^{308} , che dovrebbe essere abbastanza da soddisfare la maggior parte dei calcoli. Tuttavia, è possibile che il risultato di un'espressione sia al di fuori di questo intervallo. Per esempio, supponiamo di eseguire i seguenti comandi:

```
X = 2.5e200;  
y = 1.0e200;  
z = x*y;
```

Se supponiamo che l'intervallo dei valori sia da 10^{-308} a 10^{308} , allora il valori di x e y sono all'interno dell'intervallo permesso. Tuttavia, il valore di z è $2.5e400$, e questo valore supera l'intervallo. Questo errore è chiamato **overflow** dell'esponente, perché l'esponente del risultato di un'operazione aritmetica è troppo grande per essere memorizzato nella memoria del computer. In MATLAB, il risultato di un overflow di esponente è ∞ .

L'**underflow** dell'esponente è un errore simile, causato dal fatto che il risultato di un'operazione aritmetica è troppo piccolo per essere memorizzato nella memoria del computer. Utilizzando lo stesso intervallo permesso, otteniamo un underflow di esponente con i seguenti comandi:

```
x = 2.5e-200;  
y = 1.0e200;  
z = x/y;
```

Di nuovo, i valori di x e y sono nell'intervallo permesso, ma il valore di z sarebbe $2.5e-400$. Poiché l'esponente è più piccolo del minimo permesso, abbiamo causato un errore di underflow dell'esponente. In MATLAB, il risultato di questo errore è zero.

È noto che la **divisione per zero** è un'operazione non legittima. Se un'espressione in MATLAB risulta in una divisione per zero, il risultato della divisione è ∞ . MATLAB stamperà un messaggio di avvertimento, e continuerà i calcoli.

Operazioni sui vettori

Un'**operazione su un vettore** viene eseguita elemento per elemento. Per esempio, se A e B sono due vettori riga con cinque elementi, un modo per generare un nuovo vettore riga C con i valori che sono il prodotto dei corrispondenti valori di A e B è:

```
C(1) = A(1)*B(1);
C(2) = A(2)*B(2);
C(3) = A(3)*B(3);
C(4) = A(4)*B(4);
C(5) = A(5)*B(5);
```

Questi sono essenzialmente comandi scalari, perché ogni comando moltiplica un singolo valore per un altro singolo valore e memorizza il risultato in un terzo singolo valore. Per indicare che vogliamo fare una moltiplicazione elemento per elemento tra due vettori con le stesse dimensioni, possiamo utilizzare un asterisco preceduto da un punto. Così, le istruzioni precedenti possono essere sostituite con la seguente:

```
C = A.*B;
```

Dimenticare il punto prima dell'asterisco è un'omissione seria, perché allora l'istruzione specifica un'operazione tra matrici e non un'operazione elemento per elemento.

Per l'addizione e la sottrazione, le operazioni fra matrici e quelle fra vettori sono uguali. Tuttavia, l'operazione vettoriale per la moltiplicazione, la divisione e l'elevamento a potenza differiscono dalle corrispondenti operazioni per le matrici, così abbiamo bisogno di aggiungere un punto per specificare l'operazione fra vettori.

Queste regole sono riassunte nella Tabella 7.

Operazione	Forma algebrica	MATLAB
Addizione	$a + b$	$a + b$
Sottrazione	$a - b$	$a - b$
Moltiplicazione	$a \times b$	$a.*b$
Divisione	$\frac{a}{b}$	$a./b$
Elevamento a potenza	a^b	$a.^b$

Tab. 7 Operazione elemento per elemento

Le operazioni elemento per elemento, o operazioni vettoriali, si applicano non solo a vettori delle stesse dimensioni, ma anche a operazioni fra scalari e non scalari. Tuttavia, la moltiplicazione di una matrice per uno scalare può essere scritta in entrambi i modi. Così, le due istruzioni in ogni serie di istruzioni seguenti sono equivalenti per la matrice A.

```
B = 3*A;
B = 3.*A;
C = A/5;
C = A./5;
```

Le risultanti matrici B e C avranno le stesse dimensioni di A.

Per illustrare le operazioni per i vettori, consideriamo i due seguenti vettori riga:

$$A = [2 \quad 5 \quad 6] \quad B = [2 \quad 3 \quad 5]$$

Se calcoliamo il vettore prodotto di A e B utilizzando la seguente istruzione:

$$C = A.*B;$$

allora C conterrà i seguenti valori:

$$C = [4 \quad 15 \quad 30]$$

Il comando di divisione tra vettori:

$$C = A./B;$$

genererà un nuovo vettore in cui ogni elemento di A è diviso per il corrispondente valore di B. Così, C conterrà i seguenti valori:

$$C = [1 \quad 1.6667 \quad 1.2]$$

Possiamo anche utilizzare una base scalare e un vettore esponente, come in:

$$C = 3.0.^A;$$

che genera un vettore con questi valori:

$$C = [9 \quad 243 \quad 729]$$

Questo vettore potrebbe essere calcolato anche con l'istruzione:

$$C = (3).^A;$$

Se non siete sicuri di aver scritto l'espressione corretta, verificatela sempre con semplici esempi, come quelli che abbiamo appena utilizzato.

Gli esempi precedenti utilizzavano vettori, ma le stesse regole si applicano alle matrici, come mostrato dalle istruzioni seguenti:

$$d = [1:5; -1:-1:-5];$$

$$p = d.*5;$$

$$q = d.^3;$$

I valori di queste matrici sono:

$$d = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ -1 & -2 & -3 & -4 & -5 \end{bmatrix}$$

$$p = \begin{bmatrix} 5 & 10 & 15 & 20 & 25 \\ -5 & -10 & -15 & -20 & -25 \end{bmatrix}$$

$$q = \begin{bmatrix} 1 & 8 & 27 & 64 & 125 \\ -1 & -8 & -27 & -64 & -125 \end{bmatrix}$$

Valori speciali e matrici speciali

MATLAB contiene un grande numero di costanti predefinite, valori speciali e matrici speciali che sono disponibili per i nostri programmi. La maggior parte di questi valori e matrici speciali vengono generati da MATLAB utilizzando le funzioni. Una funzione MATLAB utilizza tipicamente input chiamati **argomenti** per calcolare una matrice, sebbene alcune funzioni non richiedano nessun argomento. Passiamo ora in rassegna alcuni esempi di funzioni di MATLAB.

Valori speciali

I valori scalari che sono disponibili per l'utilizzo in programmi MATLAB sono descritti nel seguente elenco.

pi	Rappresenta π .
i, j	Rappresentano $\sqrt{-1}$.
Inf	Rappresenta l'infinito, che tipicamente si presenta come risultato della divisione per zero. Quando viene calcolato questo risultato verrà stampato un messaggio di errore e; se visualizzate una matrice che contiene questo valore, esso verrà rappresentato come ∞ .
NaN	Rappresenta il "risultato indefinito" e tipicamente si trova quando un'espressione è indefinita, come nella divisione di zero per zero.
clock	Rappresenta l'ora corrente in un vettore riga a sei elementi che contiene l'anno, il mese, il giorno, l'ora, i minuti e i secondi.
date	Rappresenta la data corrente in una stringa di caratteri, come per esempio 20-Jun-96.
eps	Rappresenta la precisione utilizzata dal computer per i calcoli in virgola mobile. Questa precisione è la più piccola quantità di cui due valori possono differire.
ans	Rappresenta un valore calcolato da un'espressione ma che non è stato salvato in una variabile con un nome.

Matrici speciali

MATLAB contiene un gruppo di funzioni che generano matrici speciali, ne presentiamo alcune.

Matrice di zeri La funzione *zeros* genera una matrice che contiene solo zeri. Se l'argomento della funzione è uno scalare, come in *zeros(6)*, la funzione genererà una matrice quadrata utilizzando l'argomento sia come numero di righe che di colonne. Se la funzione ha due argomenti scalari, come in *zeros(m,n)*, allora la funzione genererà una matrice con m righe e n colonne di zeri. Poiché la funzione *size* calcola il numero di righe e di colonne di una matrice, possiamo utilizzarla per generare una matrice di zeri che abbia le stesse dimensioni di un'altra matrice.

Le seguenti istruzioni illustrano questi vari casi:

```
A = zeros(3);  
B = zeros(3,2);  
C = [ 1 2 3 ; 4 2 5];  
D = zeros(size(C));
```

Le matrici generate sono le seguenti:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 2 & 5 \end{bmatrix} \quad D = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Matrice di uno La funzione *ones* genera una matrice che contiene tutti uno, così come la funzione *zeros* ne genera una contenente tutti zeri. Se l'argomento della funzione è uno scalare, come in *ones(6)*, allora la funzione genera una matrice quadrata utilizzando l'argomento sia come numero delle righe che delle colonne.

Se la funzione ha due argomenti scalari, come in *ones(m,n)*, allora la funzione genererà una matrice con m righe e n colonne di zeri. Poiché la funzione *size* calcola il numero di righe e di colonne di una matrice, possiamo utilizzarla per generare una matrice di zeri che abbia le stesse dimensioni di un'altra matrice. Le seguenti istruzioni illustrano questi vari casi:

```
A = ones(3);  
B = ones(3,2);  
C = [ 1 2 3 ; 4 2 5];  
D = ones(size(C));
```

Le matrici generate sono le seguenti:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 2 & 5 \end{bmatrix} \quad D = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Matrice identità Una matrice identità è una matrice con tutti uno negli elementi della diagonale principale e zero altrove. Per esempio, la seguente matrice con quattro righe e quattro colonne è diagonale:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Notate che la **diagonale principale** contiene elementi in cui il numero di riga è lo stesso del numero di colonna. Perciò, gli indici della diagonale principale sono (1,1), (2,2), (3,3) e così via.

In MATLAB, le matrici identità possono essere generate con la singola funzione *eye*. Gli argomenti di *eye* sono simili a quelli delle funzioni *zeros* e *ones*. Se la funzione ha un unico argomento, come in *eyes(6)*, essa genererà una matrice identità utilizzando l'argomento sia come numero delle righe sia delle colonne. Per generare una matrice identità con le stesse dimensioni di un'altra già esistente, potete utilizzare di nuovo il comando *size*. Sebbene la maggior parte delle applicazioni utilizzino una matrice identità quadrata, la definizione può essere estesa a matrici non quadrate. Le seguenti istruzioni illustrano questi vari casi:

```
A = eye(3);
B = eye(3,2);
C = [ 1 2 3 ; 4 2 5];
D = eye(size(C));
```

Le matrici generate sono le seguenti:

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 2 & 5 \end{bmatrix} \quad D = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Vi raccomandiamo di non chiamare la matrice identità *i*, perché *i* non rappresenterà più 1 - nelle istruzioni seguenti.

Matrici diagonali La funzione *diag* può essere utilizzata per creare una matrice diagonale o estrarre una delle diagonali di una matrice. Per estrarre la diagonale principale dalla matrice *A* definita sopra, utilizzate:

```
diag(A)
```

Verrà generato il seguente vettore colonna:

```
ans =
1
1
1
```

Diagonali diverse possono essere estratte passando un secondo parametro k alla funzione `diag`. Il secondo parametro indica la posizione della diagonale desiderata rispetto a quella principale (che ha $k = 0$). Utilizzando la matrice C dell'esempio sopra, digitate:

```
diag(C,1)
```

Verrà generato il seguente vettore colonna:

```
ans =  
2  
5
```

Se il primo argomento di `diag` è un vettore V , allora la funzione genererà una matrice quadrata. Se $k = 0$, allora gli elementi di V verranno inseriti nella diagonale principale. Se $k > 0$ essi sono posizionati sopra la diagonale principale e se $k < 0$ allora sono posizionati sotto.

Esempio

Per esempio, immettete:

```
V = [ 2, 4, 6, 8 ];  
diag(V, 0);
```

Verrà mostrato il seguente risultato:

```
ans =  
2 0 0 0  
0 4 0 0  
0 0 6 0  
0 0 0 8
```