7 Introduzione MATLAB

7.1 Basi

Per pulire il workspace (eliminare tutte le variabili esistenti):

```
clear
```

Per pulire la finestra dei comandi (command window):

```
clc
```

In MATLAB non è necessario dichiarare le variabili ed esse **non sono tipizzate**, ovvero possono contenere qualunque cosa:

```
a = 15;
b = 5;
a = 'a';
```

La stampa di ogni contenuto nella finestra dei comandi è implicita. La presenza del ";" nasconde il risultato dell'istruzione dalla finestra dei comandi

```
c = a + b;
c = a + b
```

Il risultato di una qualunque operazione, se non è assegnato ad altre variabili, viene associato alla variabile di default ans. Il comando whos mostra il contenuto del workspace (i.e., le variabili attualmente dichiarate). Compare anche la loro dimensione, ad esempio 1×1 o 3×4: il primo numero è il numero di righe, il secondo il numero di colonne. Per MATLAB tutto è una matrice (il singolo valore è una matrice 1×1).

```
whos
```

ATTENZIONE Esistono dei nomi riservati, come i, che è l'unità immaginaria, quindi evitate di dichiarare una variabile con quello stesso nome. Così facendo, sovrascrivereste la variabile predefinita.

```
ii = 2;
i
pi
```

7.2 Vettori

La dichiarazione di un vettore riga mediante operatore CAT orizzontale `[..., ...]` (le virgole sono opzionali).

```
riga = [10, 11, 12, 13, 14];
```

Accesso ad un elemento del vettore alla posizione corrispondente ad un indice (intero) `ii` avviene attraverso le parentesi tonde

```
riga(ii)
```

ATTENZIONE Gli indici in MATLAB iniziano da 1, quindi il primo elemento è `riga(1)`. I vettori non hanno dimensioni fissate. Per accodare mediante l'operazione CAT orizzontale, un elemento al vettore riga:

```
riga = [riga, 8]
```

In questo caso stiamo sovrascrivendo al vettore `riga` un vettore di dimensione maggiore. In MATLAB possiamo fare assegnamenti di vettori (diversamente dal C).

Se si provasse ad accedere alla posizione 10 di riga

```
riga(10)
```

MATLAB da errore `index out of matrix dimensions` Se si assegna un elemento alla posizione 10 del vettore riga (non allocato precedentemente), il vettore viene allungato e ai valori intermedi viene associato di default 0

```
riga(10) = 8;
```

Per avere un vettore colonna possiamo trasporre il vettore riga oppure nella dichiarazione usare il CAT verticale: `[... ; ...]`

```
col = riga';
col = [4; 5; 6];
col = [0; col]
whos
```

In questo caso riga sarà di dimensioni $1 \times n$ mentre col $n \times 1$.

La dichiarazione di matrici avviene usando congiuntamente CAT orizzontale e verticale

```
A = [1, 2; 3, 4]
A = [1 2; 3 4]
```

La matrice viene sviluppata in un vettore leggendo le lungo le colonne (ossia viene memorizzata per colonne)

```
aa = A(:)
```

È possibile accedere agli elementi della matrice specificando dei valori ad entrambi gli indici

```
A(1, 2)
```

Se si fornisce un solo indice si intende la posizione all'interno di `A(:)`

```
A(3)
```

Allo stesso modo se cerchiamo di accedere ad un elemento al di fuori della matrice MATLAB ci comunicherà un errore

```
A(4, 3)
```

in particolare `index exceeds matrix dimensions`.

Nel caso non volessimo dichiarare il vettore elemento per elemento possiamo anche inizializzarne uno scegliendo elemento iniziale, elemento finale e passo (step) tra gli elementi

```
inizio = 9;
step = 15;
fine = 223;

v = [inizio : step : fine]
%oppure
v = inizio : step : fine
```

Se non si specifica il passo di default abbiamo passo 1

Possiamo estrarre dei sottovettori specifici andando a selezionare solo alcuni indici:

```
indici = [1, 8, 3];
c = vettore(indici)
```

oppure andando a selezionare una slice (fetta) del vettore iniziale:

```
c = v(1 : 3)
```

La keyword `end`, se utilizzata all'interno degli indici di un vettore, assume il valore corrispondente alla lunghezza del vettore

```
d = v(end)
```

E' anche possibile riordinare il vettore `v`, specificando un opportuno vettore di indici

```
vettoreAlContrario = v([end : -1 : 1]);
```

Posso dichiarare matrici di zeri e uni (tasselli base per costruire le altre matrici):

```
A = zeros(5);
A = ones(5);
```

Posso sostituire valori all'interno di una matrice mediante definizione di sottoindici

```
a = A([1 : end] , 3)
%oppure
a = A(: , 3)
```

per esempio la precedente espressione associa il vettore estratto, ossia tutti gli elementi di `A` con indice della riga da `1` a `end` e indice di colonna `3` (terza colonna), al vettore `a` Se assegno ad una sottomatrice un valore scalare, esso viene ripetuto per tutta la sottomatrice:

```
A([1 : end], 3) = 2
A(3 : end , 3 : end ) = 1
```

È possibile sovrascrivere alla terza colonna di `A` un vettore colonna di `5` elementi (le dimensioni sono consistenti)

```
A(: , 3) = [1 : 5]'
A(: , 3) = [1 : 5]
```

Il secondo comando funziona allo stesso modo perchè `MATLAB` si occupa automaticamente della trasposizione. Allo stesso modo possiamo copiare parte della matrice in un'altra posizione, sempre controllando che le dimensioni siano consistenti:

```
A([ 1 :  2 ], 1) = A([4 , 5] , 3); %SI
A([ 1 :  2 ], 1) = A([5 , 5] , 3); %SI
A([ 1 :  2 ], 1) = A([5,4] , 3); %SI
A([ 1 :  2 ], 1) = A([3 , 3, 3 ] , 3); %NO
```

Per visualizzare la matrice possiamo usare le seguenti istruzioni:

```
figure()
imagesc(A)
```

La somma tra matrici avviene elemento per elemento. Si possono sommare due matrici solo nel caso in cui siano della stessa dimensione o che una delle due sia uno scalare (che verrà replicato in maniera opportuna da MATLAB)

```
a = [1 2 3]
b = [1 2 4]
c = a + b
d = c + 3
```

Il prodotto elemento per elemento viene eseguito dall'operatore `.*`:

```
a .* b
```

mentre l'elevamento a quadrato elemento per elemento è dato dall'operatore `.^`

```
a .^ 2
```

7.3 Costrutti

Ogni costrutti inizia con una parola chiave e finisce con la parola chiave `end`. Il costrutto condizionale `if` ha la seguente sintassi

```
if %condizione
        %istruzione 1
        %istruzione 2
        %...
elseif
        %istruzione 3
        %istruzione 4
        %...
else
        %istruzione 5
        %istruzione 6
        %...
end
```

dove l'`elseif` ha il significato di un if annidato.

Il `while` è analogo a quello visto in C:

```
while %condizione
        %istruzione 1
        %istruzione 2
        %...
end
```

In MATLAB non esiste il `do ... while`, ma possiamo aggirare il problema.

Il `for` itera su di un vettore:

```
for ii = %vettore
        %istruzione 1
        %istruzione 2
end
```

ATTENZIONE È buona norma non modificare il valore del contatore `ii` all'interno del ciclo.

La sintassi dello `switch` è:

```
switch %variabile
        case %valore1
                %istruzione 1
                %istruzione 2
        case %valore2
                %istruzione 3
                %istruzione 4
        otherwise
                %istruzione 5
                %istruzione 6
end
```

Possiamo fare confronti con valori interi e con stringhe. Le istruzioni all'interno dei vari case sono esclusive.

7.4 Operazioni logiche

Anche in MATLAB abbiamo la possibilità di valutare condizioni logiche. Allo stesso modo di C, lo zero sarà considerato come falso e qualunque altro valore come vero. Le operazioni logiche sono:

```
a = 0;
b = 1;

a && b %and
a & b %and
a || b %or
a | b %or
~a %negazione

a == b %uguale
a ~= b %diverso
a > b %maggiore
a < b %minore
```

7.5 Esercizi

Esercizio 7.1

Scrivere uno script che calcola il volume di una sfera dato il raggio, sapendo che il volume di una sfera di raggio r è $V = \frac{4}{3}\pi r^3$.

Esercizio 7.2

Scrivere uno script che chieda un anno all'utente. Stampare a video se l'anno è bisestile. Il programma deve continuare a chiedere all'utente anni, finché gli anni inseriti sono bisestili. Stampare a video il numero totale di anni bisestili inseriti.

Esercizio 7.3

Scrivere uno script che analizzi i voti dell'esame di InfoB, stampando a schermo:

- media dei voti
- la media dei voti sufficienti
- il numero di promossi

```
% voti primo appello 2018/2019
voti = [24 18 17.5 19.5 7.5 15 28 31.5 11.5 21 5 19 18.5 ...
    14.5 26.5 23 30 21.5 20.5 10.5 10 7.5 19 13 11 23.5 ...
    26 30.5 15.5 5 21 20.5 22 20.5 12 8.5 24.5 11 21.5 ...
    22.5 25 20.5 24.5 24.5 16 14.5 20 21 12.5 18 21 5.5 ...
```

```
6       27.5 14 29.5 9 14 11.5 17.5 24.5 16 25 30 17 21.5 ...
7       15 13 14 15.5 11.5 25.5 30 20.5 29.5 30 7.5 9.5 ...
8       9.5 10 30.5 11 28 18.5];
```

Esercizio 7.4

Scrivere uno script che legga una frase in ingresso e la stampi al contrario.

Esercizio 7.5

Scrivere uno script che legga una frase in ingresso e la converta in alfabeto farfallino.