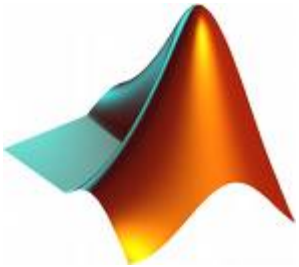


Simulazione dei Sistemi dinamici con Matlab-Simulink

Istruzioni di controllo

Ing. Alessandro Pisano
apisano@unica.it

Istruzioni di controllo

Il linguaggio Matlab possiede i classici **costrutti iterativi**

for *ripetizione di un insieme di istruzioni per un numero predeterminato di interazioni (deve terminare con* ***end****)*

while *ripetizione di un insieme di istruzioni finchè una determinata condizione rimane vera (deve terminare con* ***end****)*

break

termina l'esecuzione di un ciclo ***for*** *o* ***while***

e **condizionali**

if *istruzione condizionale (deve terminare con* ***end****) può utilizzare* ***else*** *e* ***elseif***

else *identifica un blocco di istruzioni alternative*

elseif *esegue un blocco di istruzioni se è soddisfatta una condizione alternativa*

end *termina le istruzioni* ***if***, ***for*** *e* ***while***

switch *indirizza il controllo di un programma confrontando l'espressione di input con le espressioni associate alle clausole* ***case***

case *utilizzato con* ***switch*** *per controllare l'esecuzione di un programma*

CICLI ITERATIVI FOR

Sintassi generale

```
for j = vettore di elementi

            serie di istruzioni dipendenti dalla variabile di ciclo j
end
```

La serie di istruzioni viene eseguita tante volte quanti sono gli elementi del vettore, attribuendo in successione alla variabile di ciclo j il valore deigli elementi del vettore.

In una vasta casistica di esempi, la variabile assume i valori consecutivi 1,2,….,n.

Esempio 1

Generare il vettore $x = \left[1 \quad \frac{1}{2} \quad \frac{1}{3} \quad \frac{1}{4} \quad \ldots\ldots \quad \frac{1}{N}\right]$

```
N=10;
for j = 1:N
            x(j)= 1/j;
end
```

Sintassi equivalente

```
N=10;
j = 1:N
x(j)= 1./j;
```

Esempio 2

Generare il vettore $x = \left[\frac{1}{2} \quad \frac{1}{4} \quad \frac{1}{6} \quad \ldots \quad \frac{1}{12}\right]$

```
i=1;
for j = [2 4 6 8 10 12]
            x(j)= 1/j;
            i=i+1;
 end
```

Sintassi equivalente

```
j = 2:2:12
x= 1./j;
```

Esempio 2a

Generare il vettore $x = \left[\frac{1}{12} \quad \frac{1}{10} \quad \frac{1}{8} \quad \ldots \quad \frac{1}{2}\right]$

```
i=1;
for j =12:-2:2
   x(j)= 1/j;
   i=i+1;
 end
```

Sintassi equivalente

```
j = 12:-2:2
x= 1./j;
```

Esempio 3

Creare all'interno di uno script la matrice di Hilbert di ordine N, cioè la matrice il cui elemento generico avente indice di riga i e indice di colonna j è il seguente:

$$H_{ij} = \frac{1}{i+j-1}$$

```
n=7
for i = 1:n
      for j = 1:n
            M(i,j) = 1/(i+j-1);
      end
end

M
```

Questa ultima istruzione provoca unicamente la visualizzazione del risultato finale nella command window

Esempio 3a

Creare un function file che restituisca la matrice di Hilbert di ordine N, cioè la matrice il cui elemento generico avente indice di riga i e indice di colonna j è il seguente:

$$H_{ij} = \frac{1}{i+j-1}$$

```
function [M] = creaHilbert(n)
% creaHilbert Questo function file restituisce la matrice
% di Hilbert di ordine n, in cui n è un numero naturale positivo.

for i = 1:n
     for j = 1:n
           M(i,j) = 1/(i+j-1);
     end
end

end
```

Function: creaHilbert.m

Esempio 4

Scrivere uno script file che calcoli il fattoriale di N utilizzando un ciclo for

```
N=7;
fatt=1;
for i=2:N
    fatt=fatt*i;
end

fatt
```

Flusso delle istruzioni

	Istruzione eseguita	Valore della variabile
	`fatt=1;`	
`i=2`	`fatt=1*2;`	`fatt=2`
`i=3`	`fatt=2*3;`	`fatt=6`
`i=4`	`fatt=6*4;`	`fatt=24`

Ecc ecc

Esempio 5

Si acquisiscano dal file `segnali6x716_storici.xls` i dati relativi a misure sperimentali da un set di 5 sensori di pressione e temperatura.

Processare i dati estraendo, per ciascun sensore, il valore massimo della misura e l' istante di tempo in cui si presenta tale valore.

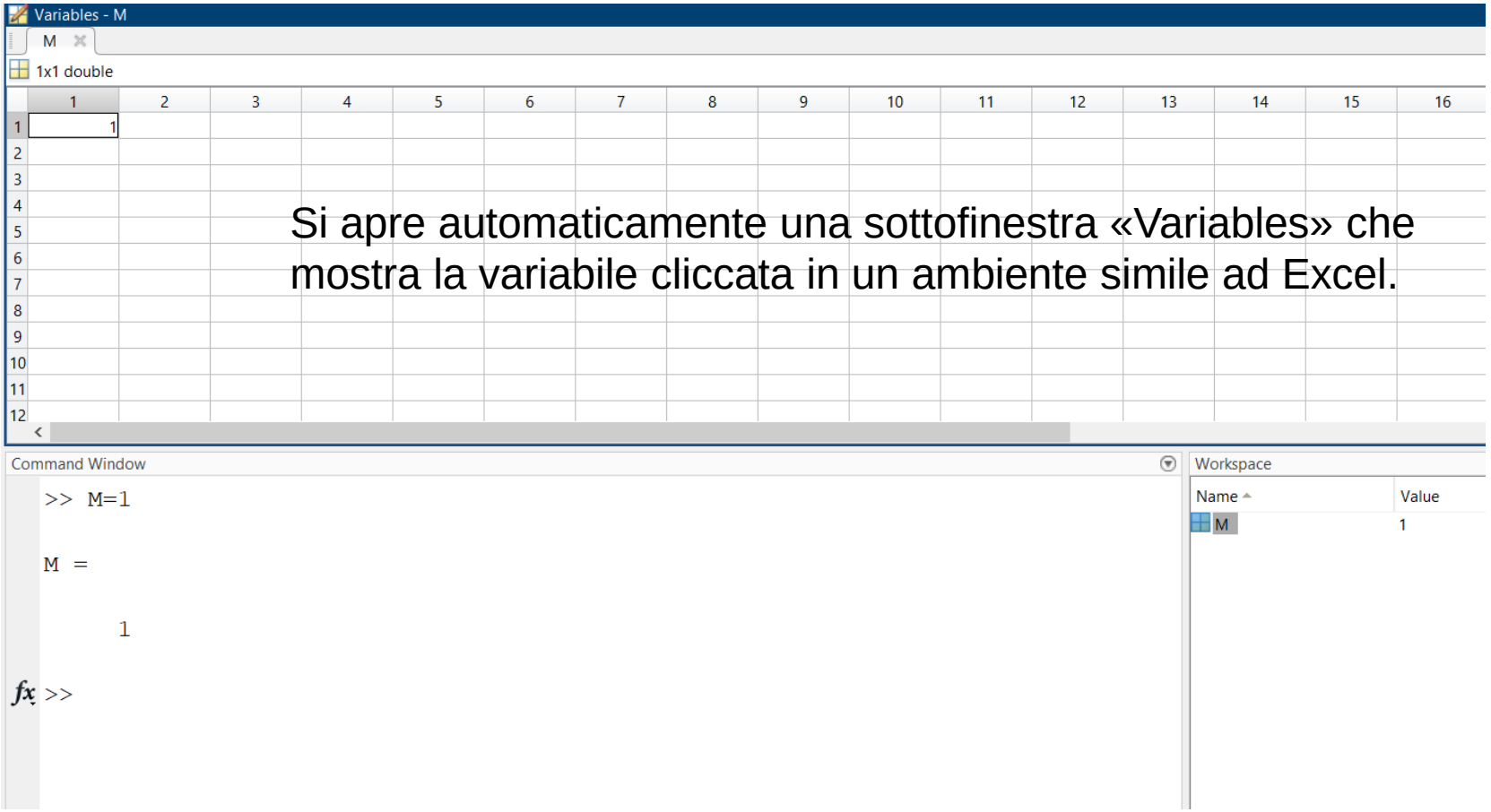
	A	B	C	D	E	F	G
1	time	temperatura1	pressione1	pressione2	temperatura2	pressione3	
2	0.0	472.08	62.38	65.67	472.08	65.67	
3	0.1	472.08	62.39	65.68	472.08	65.68	
4	0.2	472.08	62.4	65.69	472.08	65.69	
5	0.3	472.08	62.4	65.71	472.08	65.71	
6	0.4	472.08	62.41	65.72	472.08	65.72	
7	0.5	472.08	62.42	65.73	472.08	65.73	
8	0.6	472.08	62.43	65.74	472.08	65.74	
9	0.7	472.08	62.43	65.76	472.08	65.76	
10	0.8	472.08	62.44	65.77	472.08	65.77	
11	0.9	472.07	62.45	65.78	472.07	65.78	
12	1.0	472.07	62.46	65.79	472.07	65.79	

714	71.2	404.56	385.24	323.86	404.56	323.86
715	71.3	402.15	384.37	324.68	402.15	324.68
716	71.4	399.74	383.49	325.5	399.74	325.5
717	71.5	397.33	382.62	326.32	397.33	326.32
718						

Contenuto del file excel (717 righe in tutto)

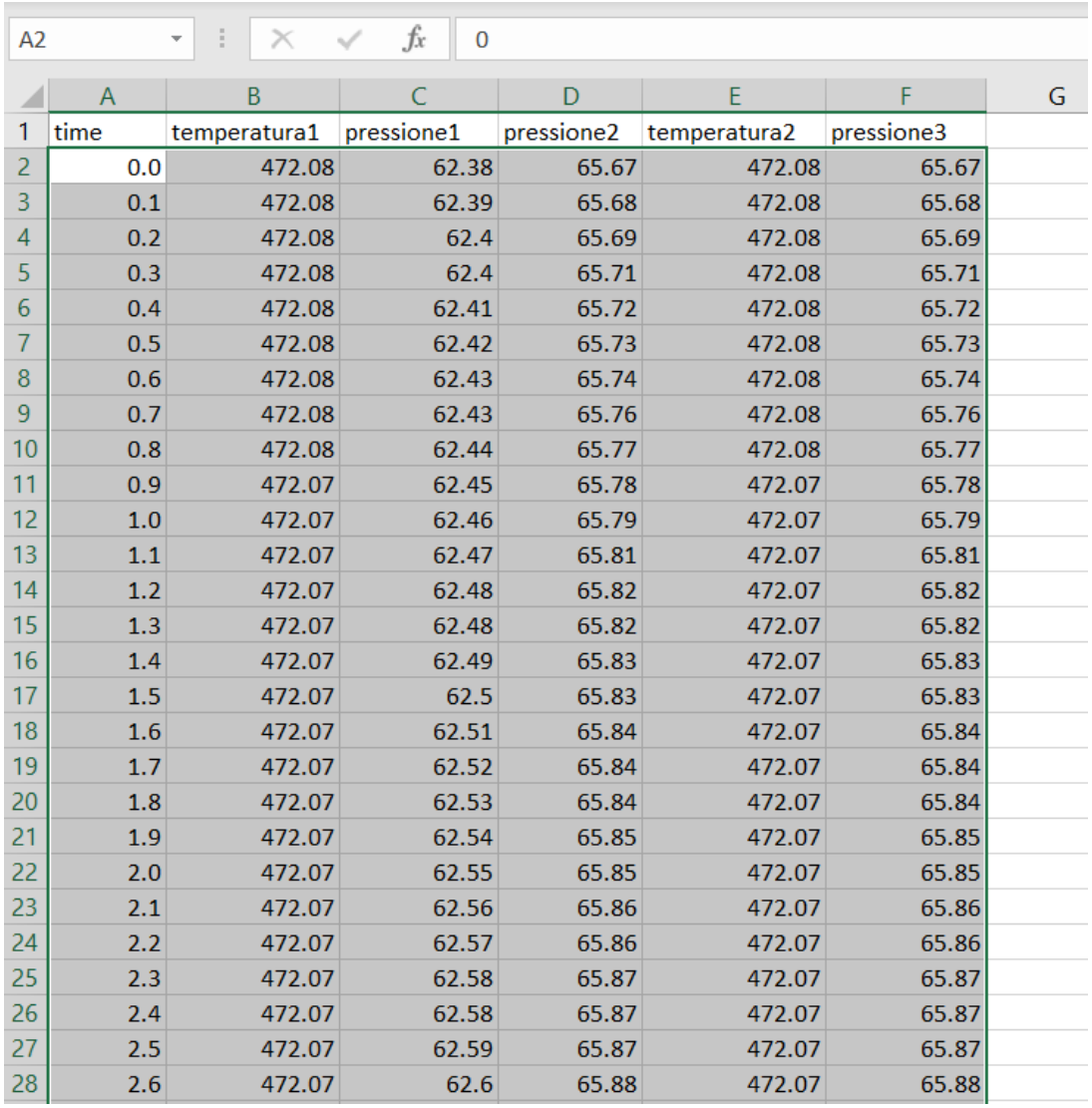
Anziché importare i dati seguendo la procedura precedentemente vista seguiamo una strada alternativa.

Creiamo una variabile fittizia scalare M=1 e successivamente facciamo doppio click sul nome della variabile all'interno della finestra Workspace.



Si apre automaticamente una sottofinestra «Variables» che mostra la variabile cliccata in un ambiente simile ad Excel.

Ora selezioniamo e copiamo in excel i dati che devono essere importati

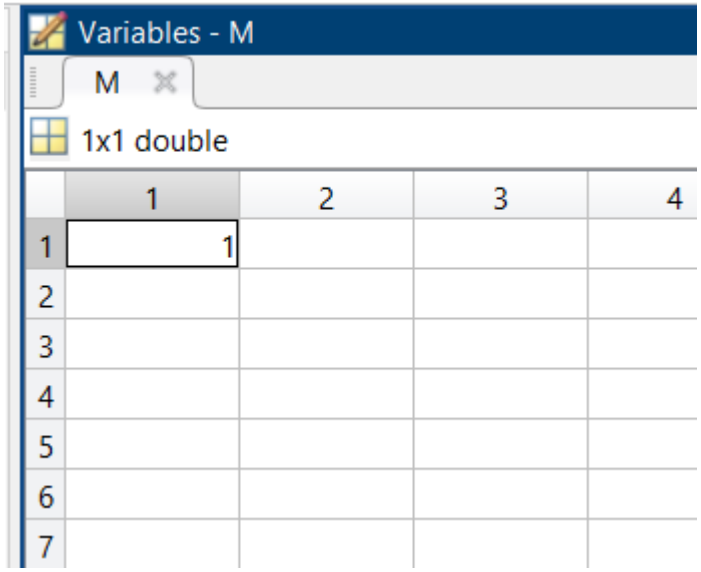


A2 | 0

	A	B	C	D	E	F	G
1	time	temperatura1	pressione1	pressione2	temperatura2	pressione3	
2	0.0	472.08	62.38	65.67	472.08	65.67	
3	0.1	472.08	62.39	65.68	472.08	65.68	
4	0.2	472.08	62.4	65.69	472.08	65.69	
5	0.3	472.08	62.4	65.71	472.08	65.71	
6	0.4	472.08	62.41	65.72	472.08	65.72	
7	0.5	472.08	62.42	65.73	472.08	65.73	
8	0.6	472.08	62.43	65.74	472.08	65.74	
9	0.7	472.08	62.43	65.76	472.08	65.76	
10	0.8	472.08	62.44	65.77	472.08	65.77	
11	0.9	472.07	62.45	65.78	472.07	65.78	
12	1.0	472.07	62.46	65.79	472.07	65.79	
13	1.1	472.07	62.47	65.81	472.07	65.81	
14	1.2	472.07	62.48	65.82	472.07	65.82	
15	1.3	472.07	62.48	65.82	472.07	65.82	
16	1.4	472.07	62.49	65.83	472.07	65.83	
17	1.5	472.07	62.5	65.83	472.07	65.83	
18	1.6	472.07	62.51	65.84	472.07	65.84	
19	1.7	472.07	62.52	65.84	472.07	65.84	
20	1.8	472.07	62.53	65.84	472.07	65.84	
21	1.9	472.07	62.54	65.85	472.07	65.85	
22	2.0	472.07	62.55	65.85	472.07	65.85	
23	2.1	472.07	62.56	65.86	472.07	65.86	
24	2.2	472.07	62.57	65.86	472.07	65.86	
25	2.3	472.07	62.58	65.87	472.07	65.87	
26	2.4	472.07	62.58	65.87	472.07	65.87	
27	2.5	472.07	62.59	65.87	472.07	65.87	
28	2.6	472.07	62.6	65.88	472.07	65.88	

Ctrl+C

Ora selezioniamo l’elemento unitario all’interno della finestra «Variables»



ed incolliamo i valori copiati (Ctrl+V)

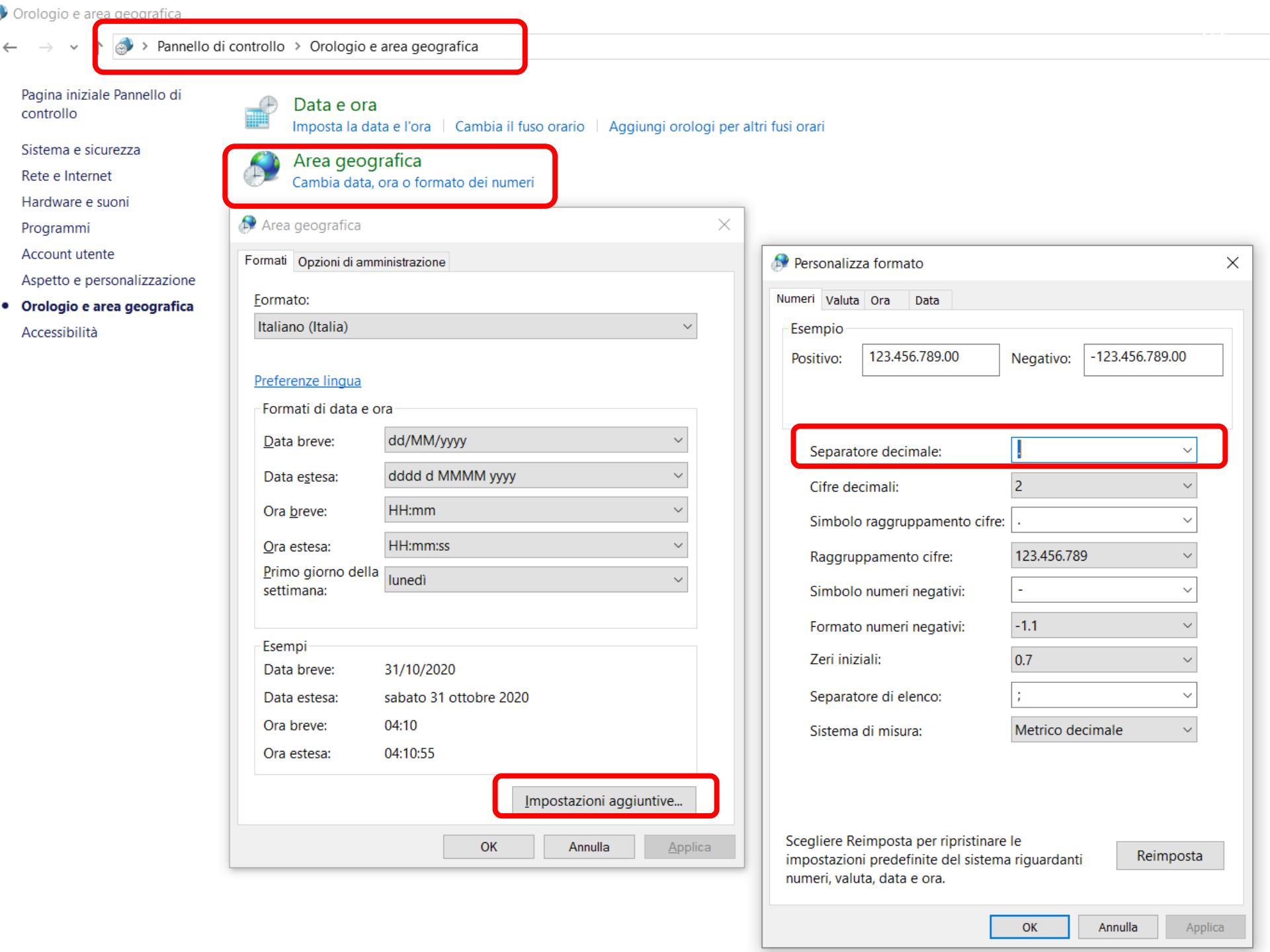
La variabile viene aggiornata ed è ora una matrice 716x6

M

716x6 double

	1	2	3	4	5	6	7
1	0	472.0800	62.3800	65.6700	472.0800	65.6700	
2	0.1000	472.0800	62.3900	65.6800	472.0800	65.6800	
3	0.2000	472.0800	62.4000	65.6900	472.0800	65.6900	
4	0.3000	472.0800	62.4000	65.7100	472.0800	65.7100	
5	0.4000	472.0800	62.4100	65.7200	472.0800	65.7200	
6	0.5000	472.0800	62.4200	65.7300	472.0800	65.7300	
7	0.6000	472.0800	62.4300	65.7400	472.0800	65.7400	
8	0.7000	472.0800	62.4300	65.7600	472.0800	65.7600	
9	0.8000	472.0800	62.4400	65.7700	472.0800	65.7700	
10	0.9000	472.0700	62.4500	65.7800	472.0700	65.7800	
11	1	472.0700	62.4600	65.7900	472.0700	65.7900	
12	1.1000	472.0700	62.4700	65.8100	472.0700	65.8100	

NB questa procedura funziona solo se nel file Excel il carattere separatore dei decimali è il punto. E' una impostazione che si definisce nel Pannello di controllo -> Orologio e Area Geografica (v. slide successiva)


Orologio e area geografica
Pannello di controllo › Orologio e area geografica
Pagina iniziale Pannello di controllo
Sistema e sicurezza
Rete e Internet
Hardware e suoni
Programmi
Account utente
Aspetto e personalizzazione
Orologio e area geografica
Accessibilità
Data e ora
Imposta la data e l'ora | Cambia il fuso orario | Aggiungi orologi per altri fusi orari
Area geografica
Cambia data, ora o formato dei numeri
Area geografica
Formati
Opzioni di amministrazione
Formato:
Italiano (Italia)
Preferenze lingua
Formati di data e ora
Data breve: dd/MM/yyyy
Data estesa: dddd d MMMM yyyy
Ora breve: HH:mm
Ora estesa: HH:mm:ss
Primo giorno della settimana: lunedì
Esempi
Data breve: 31/10/2020
Data estesa: sabato 31 ottobre 2020
Ora breve: 04:10
Ora estesa: 04:10:55
Impostazioni aggiuntive...
OK
Annulla
Applica
Personalizza formato
Numeri
Valuta
Ora
Data
Esempio
Positivo: 123.456.789.00
Negativo: -123.456.789.00
Separatore decimale: ,
Cifre decimali: 2
Simbolo raggruppamento cifre: .
Raggruppamento cifre: 123.456.789
Simbolo numeri negativi: -
Formato numeri negativi: -1.1
Zeri iniziali: 0.7
Separatore di elenco: ;
Sistema di misura: Metrico decimale
Scegliere Reimposta per ripristinare le impostazioni predefinite del sistema riguardanti numeri, valuta, data e ora.
Reimposta
OK
Annulla
Applica

Ora decidiamo come formattare i dati di uscita. Decidiamo che i dati relativi ai 5 sensori saranno prodotti mediante una matrice 2x5 che chiameremo OUT

OUT=

Sensore T1	Sensore P1	Sensore P2	Sensore T2	Sensore P3
Valore massimo	Valore massimo			Valore massimo
Istante del massimo	Istante del massimo			Istante del massimo

Per risolvere questo problema useremo la funzione `max` precedentemente illustrata, in particolare sfrutteremo la possibilità di ottenere dalla funzione `max` non solo la componente massima di un vettore ma anche la posizione all'interno del vettore della componente massima, da cui potremo desumere il relativo istante di massimo.

Ricordiamo che nella prima colonna della matrice M sono custoditi gli istanti di acquisizione delle misure, mentre nelle restanti colonne 5 troviamo i dati acquisiti dai vari sensori.

```
j=1;

for i=2:6
[mass(j),posmax(j)]=max(M(:,i));
 j=j+1;
end

tmass=M(posmax,1)'

OUT=[mass ; tmass]
```

Commentiamo questo codice
Nella slide successiva

```
OUT =

  551.7300  497.4200  389.2500  551.7300  389.2500
   69.7000   45.9000   46.5000   69.7000   46.5000
```

Script: massimosegnali.m

```
j=1;
for i=2:6
 [mass(j),posmax(j)]=max(M(:,i));
 j=j+1;
end

tmass=M(posmax,1)'

OUT=[mass ; tmass]
```

Script: massimosegnali.m

Il ciclo FOR scandisce le colonne della matrice M dalla seconda alla sesta.

Il ciclo FOR intende creare due vettori riga, chiamati `mass` e `posmax`, che conterranno nell'ordine i valori massimi acquisiti dai vari sensori e le relative posizioni dei valori massimi all'interno delle corrispondenti colonne della matrice M

La variabile j serve per fare in modo che, ad ogni passo del ciclo FOR, vengano via via definiti gli elementi dei vettori `mass` e `posmax`

Che risultato otterremmo impiegando il ciclo FOR modificato riportato qui a destra?

```
for i=2:6
 [mass(i),posmax(i)]=max(M(:,i));
end
```

```
j=1;
for i=2:6
 [mass(j),posmax(j)]=max(M(:,i));
 j=j+1;
end

tmass=M(posmax,1)'

OUT=[mass ; tmass]
```

Script: massimosegnali.m

L'istruzione `tmass=M(posmax,1)'` estrae dalla prima colonna della matrice M gli istanti di tempo associati ai vari punti di massimo, che sono allocati nelle componenti aventi indice `posmax`. La trasposizione finale serve per fare in modo che `tmass` diventi un vettore riga.

Concludiamo questo esempio con una «anticipazione» in merito all'argomento della creazione dei grafici.

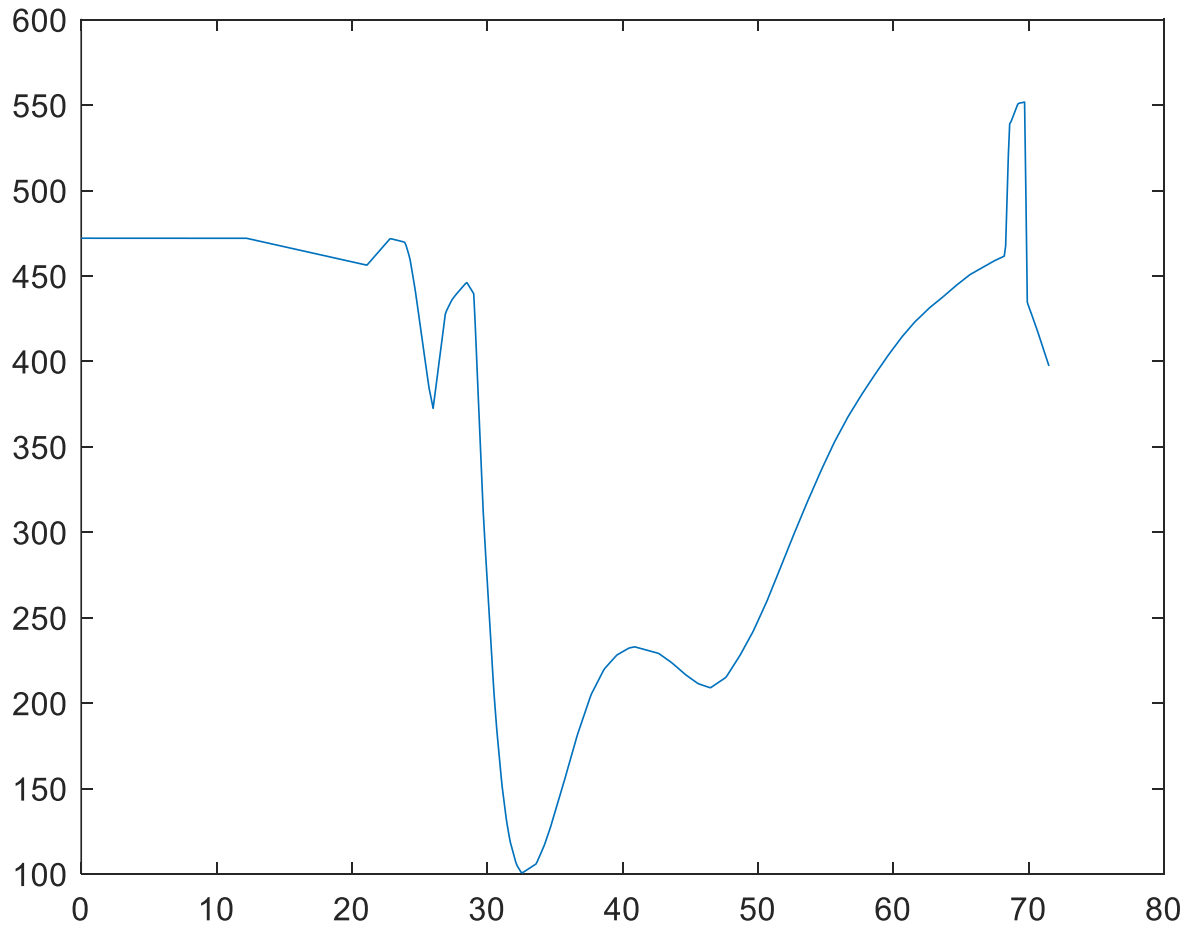
Desideriamo in particolare creare un grafico che mostri i dati acquisiti da uno dei sensori onde verificare che l'analisi dei dati che abbiamo effettuato sia corretta.

Per creare un grafico 2D di un segnale si utilizza in genere la funzione `plot` alla quale dovremo passare in ingresso, nell'ordine, il vettore che contiene gli istanti di acquisizione dei campioni del segnale (con riferimento all'esempio corrente si tratta dei dati custoditi nella prima colonna della matrice M) ed il vettore che contiene i campioni del segnale (i dati che si trovano, per i vari sensori, nelle altre colonne di M).

L'istruzione di base per graficare, ad esempio, le misure acquisite dalla sonda T1 è pertanto la seguente

```
plot(M(:,1),M(:,2))
```

```
plot(M(:,1),M(:,2))
```

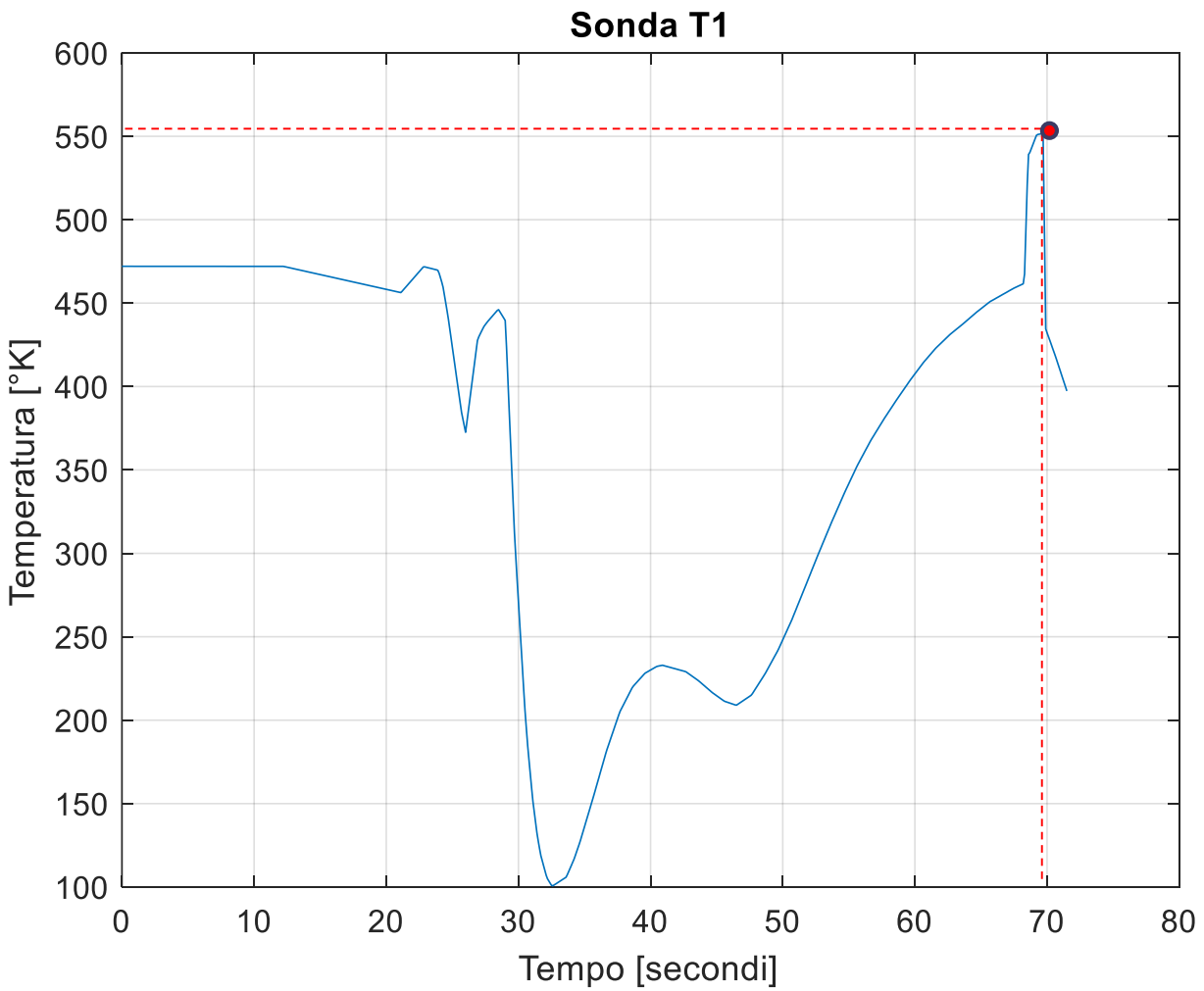

600
550
500
450
400
350
300
250
200
150
100
0
10
20
30
40
50
60
70
80

```
plot(M(:,1),M(:,2))
grid
xlabel('Tempo [secondi]')
ylabel ('Temperatura [°K]')
title('Sonda T1')
```

```
OUT =

  551.7300  497.4200  389.2500  551.7300  389.2500
   69.7000   45.9000   46.5000   69.7000   46.5000
```

Grafico corredato
di etichette e titolo



Stringhe

Un tipo particolare di array è costituito dalle **stringhe di caratteri**.
Si definisce tra **apici**

```
>> s='questa e'' una stringa'

s =

questa e' una stringa
```

E' possibile concatenare fra loro diverse stringhe accorpandole tra parentesi quadre, come mostrato nel seguente esempio.

La funzione `num2str` consente di "importare" all'interno di una stringa il valore numerico di una variabile

```
a=3;
s=['Il valore di a è pari a ' num2str(a)]
```

```
s =

    'Il valore di a è pari a 3'
```

`eval(s)` *Esegue una stringa s come comando Matlab*

Il seguente ciclo FOR genera 12 matrici random 2 x 2 M1, M2, … , M12

```
for n=1:12
   eval(['M' num2str(n) '=rand(2)'])
end
```

```
M1 =

    0.8147    0.1270
    0.9058    0.9134


M2 =

    0.6324    0.2785
    0.0975    0.5469
```

```
M11 =

    0.4387    0.7655
    0.3816    0.7952


M12 =

    0.1869    0.4456
    0.4898    0.6463
```

Ciclo FOR esploso istruzione per istruzione

```
M1=rand(2)
M2=rand(2)
M3=rand(2)
      …
M12=rand(2)
```

Script: Eval01.m

SINTASSI CONDIZIONALE IF

```
if   condizione che restituisce 0-1

        Istruzioni
end
```

Costrutto di base

```
if   condizione che restituisce 0-1

          Istruzioni
else
          Istruzioni alternative
end
```

Costrutto con due alternative

```
if   condizione1 che restituisce 0-1

          Istruzioni

elseif  condizione2 che restituisce 0-1

         Istruzioni alternative1
else
         Istruzioni alternative2

end
```

Costrutto con tre alternative

SINTASSI IF

Esempio 1

```
x=2.5;
if x>2
     disp(x)
end
```

L'istruzione `disp` (abbreviazione di display) visualizza nella Command Window il valore di una variabile senza riportare anche il nome della variabile.

Esempio 2

L'istruzione `disp` consente di visualizzare nella Command Window anche delle stringhe di testo, che in ambiente Matlab si denotano fra apici.

```
clc
disp('Riconoscimento di numeri pari o dispari.')

n=input('Inserisci un numero intero n e premi invio \n n=');

if mod(n,2)==0
     disp('il numero n è pari')
 else
     disp('il numero n è dispari')
 end
```

Script: paridispari.m

SINTASSI IF

Esempio 1

```
x=2.5;
if x>2
     disp(x)
end
```

L'istruzione `disp` (abbreviazione di display) visualizza nella Command Window il valore di una variabile senza riportare anche il nome della variabile.

Esempio 2

L'istruzione `disp` consente di visualizzare nella Command Window anche delle stringhe di testo, che in ambiente Matlab si denotano fra apici.

```
clc
disp('Riconoscimento di numeri pari o dispari.')

n=input('Inserisci un numero intero n e premi invio \n n=');

if mod(n,2)==0
     disp('il numero n è pari')
 else
     disp('il numero n è dispari')
 end
```

L'istruzione `input` consente di acquisire il valore di una variabile (in questo caso la variabile `n`) attraverso il suo inserimento diretto da parte dell'utente, con una stringa descrittiva di commento

SINTASSI IF

Esempio 1

```
x=2.5;
if x>2
     disp(x)
end
```

L'istruzione `disp` (abbreviazione di display) visualizza nella Command Window il valore di una variabile senza riportare anche il nome della variabile.

Esempio 2

```
clc
disp('Riconoscimento di numeri pari o dispari.')

n=input('Inserisci un numero intero n e premi invio \n n=');

if mod(n,2)==0
     disp('il numero n è pari')
 else
     disp('il numero n è dispari')
 end
```

L'istruzione `mod(n,m)` restituisce il resto della divisione fra i numeri n ed m. Inserendo come secondo argomento m=2 si può quindi impiegare per verificare se un numero intero n sia pari o dispari

Esempi – FOR/IF

Costruire la matrice tridiagonale

$$M = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & \cdots & \ddots & \cdots & 0 \\ \vdots & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & \cdots & 0 & -1 & 2 \end{bmatrix}$$

```
N=7;
for I=1:N
    for J=1:N
        if I == J
                M(I,J) = 2;
                elseif abs(I-J) == 1
                M(I,J) = -1;
                else
                M(I,J) = 0;
        end
     end
end
```

Script: tridiagonale01.m

Codice alternativo

```
N=7;
M=zeros(N);
for i = 1:N
      for j = 1:N
            if i == j
                M(i,j) = 2;
            end
            if (abs(i-j)==1)
                M(i,j)=-1;
            end
      end
end
```

$$M = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & \cdots & \ddots & \cdots & 0 \\ \vdots & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & \cdots & 0 & -1 & 2 \end{bmatrix}$$

Script: tridiagonale02.m

Codice alternativo

```
N=7;
M=zeros(N);
for i = 1:N
    M(i,i)=2;
end

for i = 1:N-1
   M(i,i+1)=-1;
   M(i+1,i)=-1;
end
```

$$M = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & 0 & \cdots & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & \cdots & \ddots & \cdots & 0 \\ \vdots & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & \cdots & 0 & -1 & 2 \end{bmatrix}$$

Script: tridiagonale03.m

Esempio

Function file che riceve in ingresso due matrici quadrate di pari dimensione e restituisce il determinante della matrice prodotto

```
function [out] = detmatprod2(A1,A2)
%DETMATPROD determinante della matrice prodotto
% con verifica dimensionale

[nrA1, ncA1]=size(A1);
[nrA2, ncA2]=size(A2);

if(nrA1~=ncA1)
    disp('la prima matrice non è quadrata')
    return
end

if(nrA2~=ncA2)
    disp('la seconda matrice non è quadrata')
    return
end

if(nrA1~=nrA2)
    disp('le due matrici hanno dimensioni differenti')
    return
end

out=det(A1*A2);
end
```

Completiamo questa function, realizzata in precedenza, con delle istruzioni aggiuntive che verificano le dimensioni delle matrici inserite ed interrompono l'esecuzione della funzione (comando `return`) in caso di incompatibilità, con la produzione di un messaggio di errore

function: detmatprod2.m

0	62.3800
0.1000	62.3900
0.2000	62.4000
0.3000	62.4000
0.4000	62.4100
0.5000	62.4200
0.6000	62.4300
0.7000	62.4300
0.8000	62.4400
0.9000	62.4500
1	62.4600
1.1000	62.4700
1.2000	62.4800
1.4000	62.4900
1.5000	62.5000
1.6000	62.5100
1.7000	62.5200
1.8000	62.5300
1.9000	62.5400
2	62.5500
2.1000	62.5600
2.2000	62.5700
2.3000	62.5800
2.4000	62.5800
2.6000	62.6000
2.7000	62.6100
2.8000	62.6200
2.9000	62.6300
3	62.6400

Rimozione di dati erronei da uno stream

Carichiamo nel workspace, con il comando LOAD, il file stream.mat

Esso provoca il caricamento nel WS di due vettori, time e data.

Desideriamo rimuovere dallo stream i dati nulli, per ipotesi fuori range.

Script

```
clear all
load stream
j=1;
for i=1:length(data)
     if data(i)==0
       p(j)=i;
       j=j+1;
     end
end
p
data_f=data;
time_f=time;
data_f(p)=[];
time_f(p)=[];
```

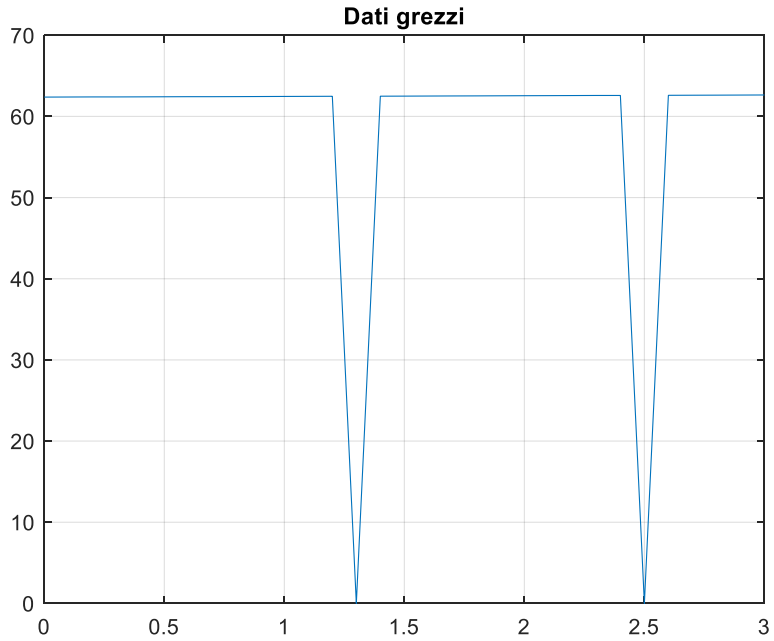
Creiamo il vettore `p` che contiene gli indici associati alle misure da rimuovere dallo stream

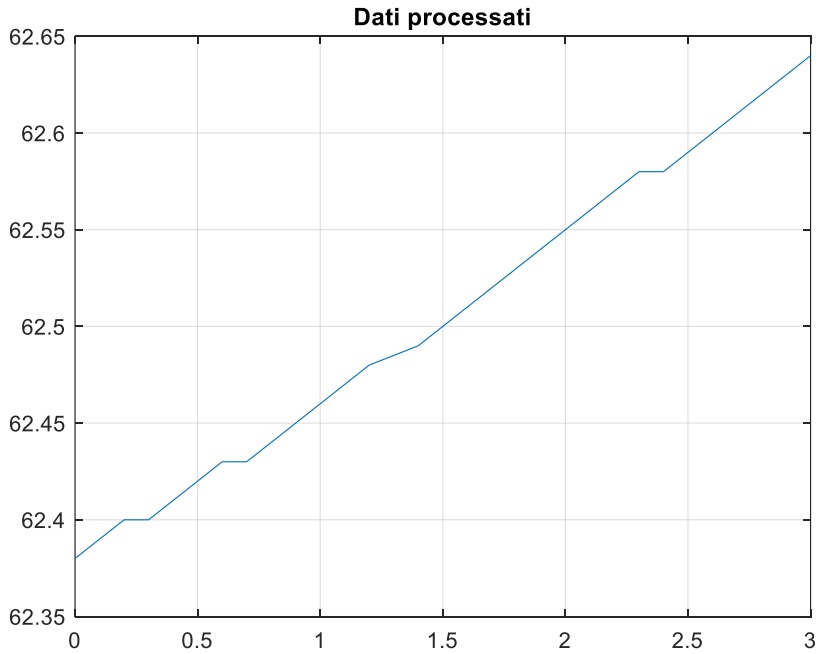
```
p =

    14    26
```

Script: stream01.m

```
figure(1)
plot(time,data),grid
title('Dati grezzi')
figure(2)
plot(time_f,data_f),grid
title('Dati processati')
```





Operatori logici

~ NOT
& AND
| OR
xor(a,b)

Gli operatori logici si applicano tra scalari, tra vettori (o matrici) di dimensioni analoghe, oppure tra un vettore (o matrice) ed uno scalare.

Utili nel contesto dei cicli IF o WHILE per costruire condizioni complesse

I seguenti script acquisiscono un dato mediante inserimento diretto da tastiera e verificano se appartiene o meno all'intervallo [Xmin , Xmax], producendo una corrispondente striga di testo

Versione dello script che utilizza l'operatore AND

```
X=input('Inserire il dato: \n X=')
Xmin=1;  Xmax=80;
if ((X>=Xmin) & (X <= Xmax))
     disp('Il dato è nel range ammissibile')
else
     disp('Il dato è fuori range')
end
```

Script: logicand.m

Versione dello script che utilizza l'operatore OR

```
X=input('Inserire il dato: \n X=')
Xmin=1; Xmax=80;
if ((X<Xmin) | (X > Xmax))
     disp('Il dato è fuori range')
else
     disp('Il dato è nel range ammissibile')
end
```

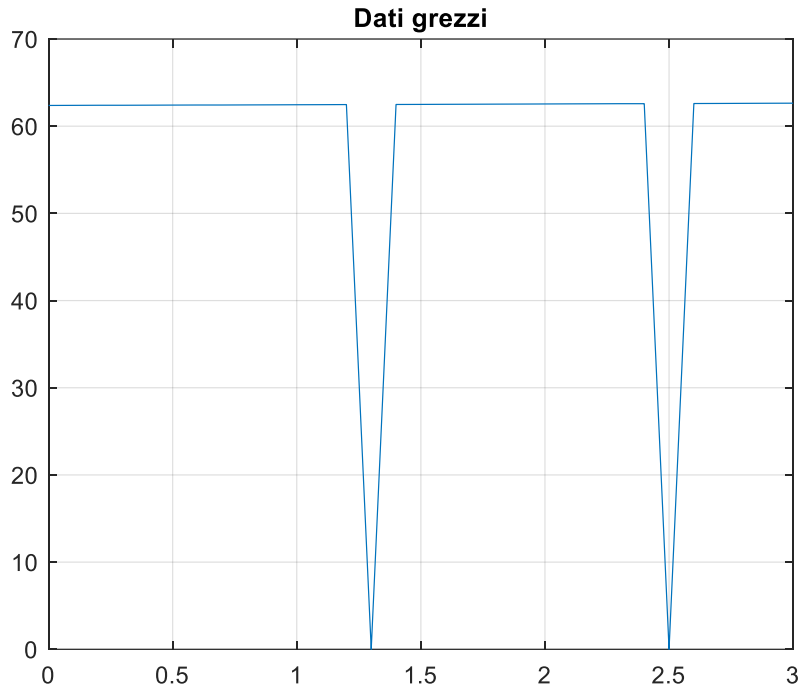
Script: logicor.m

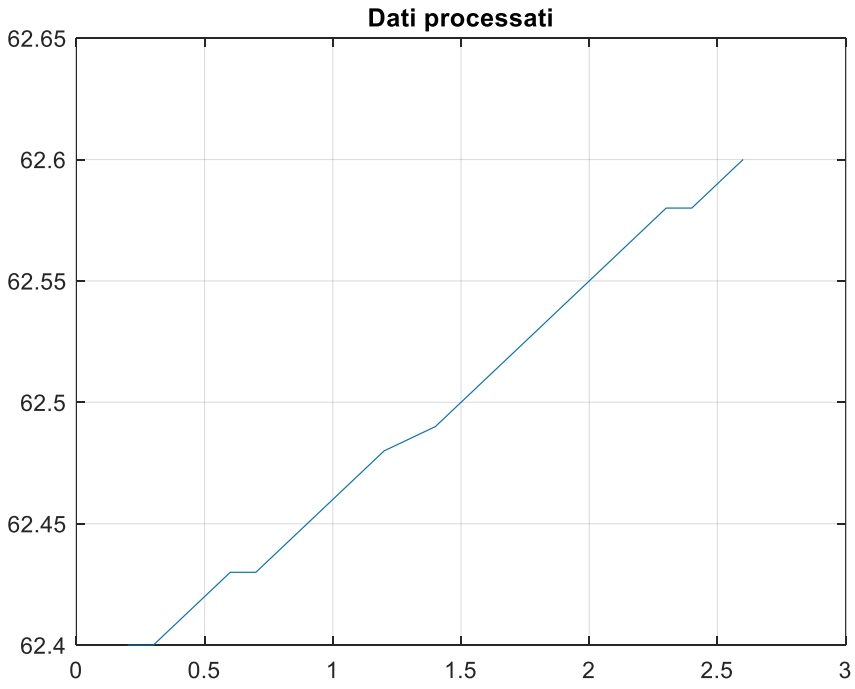
Sviluppiamo una variante dello script predisposto poche slides fa per rimuovere le misure nulle da uno stream di dati.
Il codice seguente rimuove tutte le misure il cui valore è al di fuori della soglia ammissibile `[sogliamin , sogliamax]`

```
clear all
load stream
sogliamin=62.4
sogliamax=62.6
j=1;
for i=1:length(data)
     if (data(i)<sogliamin) | (data(i)>sogliamax)
       p(j)=i;
       j=j+1;
     end
end
p
data_f=data;
time_f=time;
data_f(p)=[];
time_f(p)=[];
```

Script: stream02.m

```
figure(1)
plot(time,data),grid
title('Dati grezzi')
figure(2)
plot(time_f,data_f),grid
title('Dati processati')
```





Esercizio

Scrivere una funzione che, assegnata una dimensione *n* in input, costruisca la matrice

$$M = \begin{bmatrix} A & B \\ B^T & A \end{bmatrix} \qquad A, B \in \mathbb{R}^{n \times n} \qquad M \in \mathbb{R}^{2n \times 2n}$$

i cui blocchi, tutti di dimensione *n*, sono i seguenti: *A* è una tridiagonale che contiene il numero *n* sulla diagonale principale e il numero 2 sulla sottodiagonale e sulla sopradiagonale; B è una matrice triangolare superiore i cui elementi non nulli sono tutti uguali a 3.

Scrivere quindi uno script che, per ogni dimensione n=10,11,12,….,30 costruisca la matrice M utilizzando la funzione creata e calcoli per ciascun valore di n il numero di autovalori di M compresi fra 0 ed n. Il numero di autovalori compresi fra 0 ed n sarà alloggiato in una matrice OUT, insieme al corrispondente valore di n, avente la struttura seguente

Struttura della matrice OUT

valore di n	10	11	12	----	29	30
numero di autovalori compresi fra 0 ed n	12	14	15	----	38	39

```
function M = creaM(n)

A=zeros(n);
for i=1:n
    A(i,i)=n;
end
for i=1:n
    for j=1:n
        if abs(i-j)==1
            A(i,j)=2;
        end
    end
end
B=zeros(n);
for i=1:n
    for j=1:n
        if (j>=i)
            B(i,j)=3;
        end
    end
end

M=[A B;B' A];
end
```

FUNCTION

Function: creaM.m

```
>> M=creaM(4)

M =

     4     2     0     0     3     3     3     3
     2     4     2     0     0     3     3     3
     0     2     4     2     0     0     3     3
     0     0     2     4     0     0     0     3
     3     0     0     0     4     2     0     0
     3     3     0     0     2     4     2     0
     3     3     3     0     0     2     4     2
     3     3     3     3     0     0     2     4
```

SCRIPT

```
clc,
i=10:1:30;
eig0n=zeros(1,length(i));
ind=1;
for n=i
    M=creaM(n);
    eigM=eig(M)
    num=0;
    for j=1:length(eigM)
        if (eigM(j)>=0 && eigM(j)<=n)
                num=num+1;
        end
    end
    eig0n(ind)=num;
    ind=ind+1
end

OUT=[i; eig0n]
clear eig0n n eigM M
```

Pre-allocazione del vettore

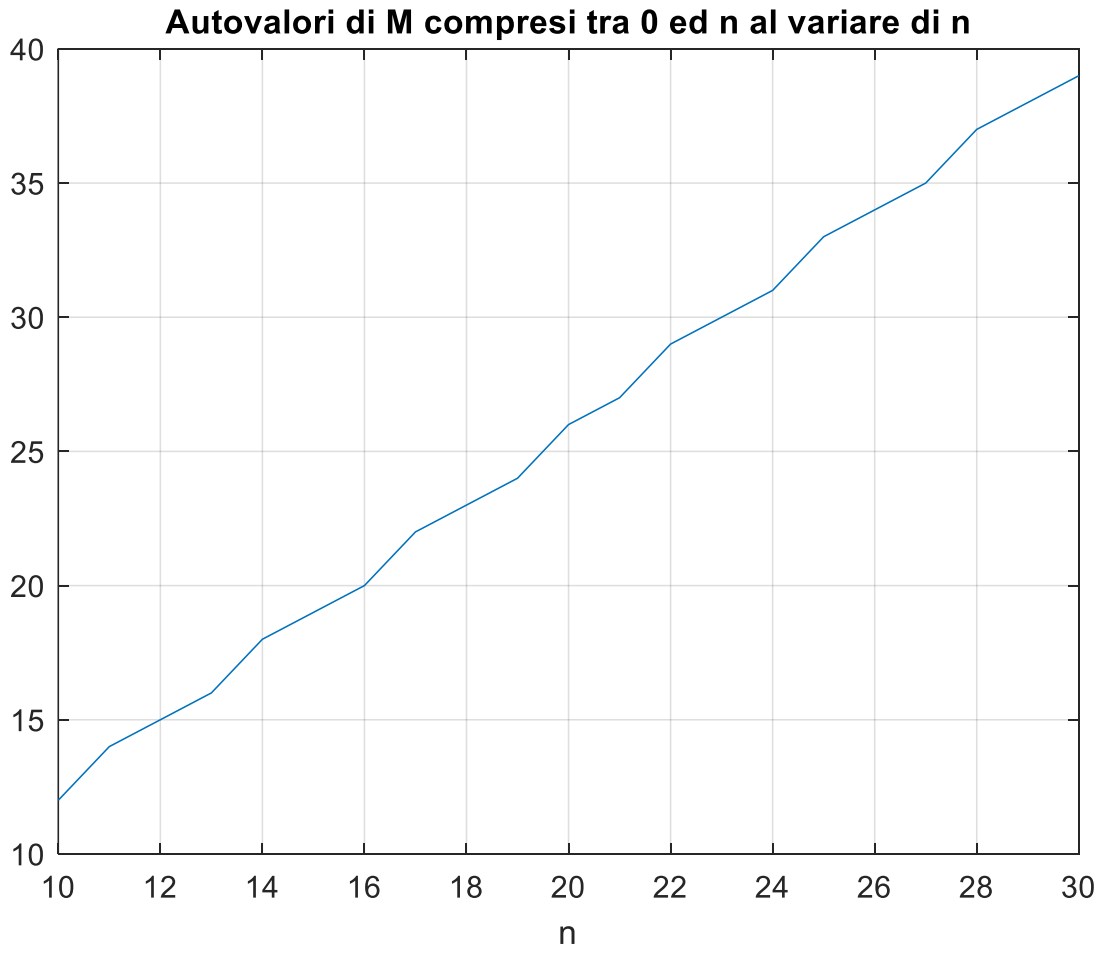
La variabile num conta il numero di autovalori compresi fra 0 ed n

Rimozione variabili ridondanti

Nella prossima slide si riportano le istruzioni aggiuntive per la creazione di un grafico che illustra i risultati

Script: ScriptcreaM.m

```
plot(OUT(1,:),OUT(2,:)),grid
xlabel('n')
title('Autovalori di M compresi tra 0 ed n al variare di n')
```



CICLI WHILE

Sintassi di base

```
while condizione
    sequenza di istruzioni
end
```

La sequenza di istruzioni viene eseguita finche “condizione” risulta True.

Una o più delle istruzioni devono influenzare la condizione, in caso contrario le istruzioni vengono eseguite infinite volte oppure mai.

Rischio di loop infinito

CLTR+C per interrompere l’esecuzione di uno script che sia in loop infinito

Esempio Stampare i primi 10 numeri interi

```
zp = 0;

while zp <10
        zp=zp+1;
        disp(zp)
end
```

Script: While01.m

Esempio Generare dei numeri random e sommarli fra di loro finchè il numero random generato alla generica iterazione non sia maggiore di una soglia

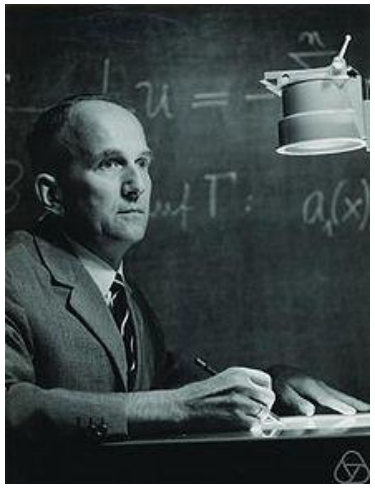
```
soglia = 0.9;
s = 0; i=0;
while 1
    tmp = rand;
    if tmp > soglia
        break
    end
    s = s + tmp;
    i=i+1;
end
i
```

Il ciclo while ha una condizione di ciclo sempre vera, e si esce dal ciclo grazie alla funzione break. La variabile i è impiegata per contare il numero di iterazioni che vengono svolte prima della uscita dal ciclo

Script: While02.m

Esercizio

La congettura di Collatz (L. Collatz, 1937) stabilisce che la seguente sequenza numerica



$$n_{k+1} = \begin{cases} \dfrac{n_k}{2} & \text{se } n_k \text{ è pari} \\ 3n_k + 1 & \text{se } n_k \text{ è dispari} \end{cases}$$

converge ad 1 in un numero finito di passi qualunque sia il valore intero positivo di partenza n_o

Verificare la congettura di Collatz per un numero intero di partenza n_o inserito a piacere dall'utente.
Memorizzare in un numero intero NUMEL il numero di iterazioni necessario.
Memorizzare in un vettore CSEQ la sequenza corrispondente.

```
clear all, clc
n=input('Inserire il numero n0 di partenza: \n n0=');
i=1;
CSEQ(i)=n;
while n>1
  if mod(n,2)==0
       n=n/2;
  else n=3*n+1;
  end
i=i+1;
CSEQ(i)=n;
end

CSEQ
NUMEL=i
%NUMEL=length(COLLATZ_SEQ);  %istruz alternativa
```

Output con n_0=22

```
CSEQ =

  Columns 1 through 12

    22    11    34    17    52    26    13    40    20    10     5    16

  Columns 13 through 16

     8     4     2     1


NUMEL =

    16
```

Script: collatz1.m

SWITCH/ CASE

```
switch variabile
    case valore_1
        istruzioni_1;
    case valore_2
        istruzioni_2;
    case valore_3
        istruzioni_3;
end
```

Sintassi di base

Se variabile assume valore1, valore2 oppure valore3 vengono eseguite rispettivamente le istruzioni1, istruzioni2 o istruzioni3.

```
switch variabile
    case valore_1
        istruzioni_1;
    case valore_2
        istruzioni_2;
    case valore_3
        istruzioni_3;
   otherwise
        istruzioni_4;
end
```

Sintassi piu generale

Se variabile assume valore1, valore2 oppure valore3 vengono eseguite rispettivamente le istruzioni1, istruzioni2 o istruzioni3.
Se variabile non assume nessuno di questi valori allora vengono eseguite le istruzioni4

Esempi Richiedere all'utente l'inserimento di una matrice quadrata, e successivamente di un numero che sarà «1» o «2» se si desidera che venga calcolato il determinante oppure gli autovalori della matrice inserita.

```
clear all,clc
M=input('Inserire una matrice quadrata: \n M=')

x=input(['Inserire "1" per calcolare il determinante \n' ...
     'oppure "2" per calcolare gli autovalori: \n']);

switch x
     case 1
             disp('Il determinante è:')
             D=det(M)
     case 2
             disp('Gli autovalori sono:')
             AUT=eig(M)
end
```

Script: SwitchCase01.m

Esempi Stesso esempio di prima, con un messaggio nel caso in cui il numero inserito non sia uno di quelli richiesti.

```
clear all,clc
M=input('Inserire una matrice quadrata: \n M=')

x=input(['Inserire "1" per calcolare il determinante \n' ...
     'e "2" per calcolare gli autovalori: \n']);
switch x
     case 1
              disp('Il determinante è:')
              D=det(M)
    case 2
              disp('Gli autovalori sono:')
              AUT=eig(M)
    otherwise
        disp('Numero inserito non corretto')
end
```

Script: SwitchCase02.m

Stesso esempio di prima con la possibilità di **scelte multiple**

```
clear all,clc
M=input('Inserire una matrice quadrata: \n M=')

x=input(['Inserire "1" oppure "D" per calcolare il determinante \n' ...
     'e "2" oppure "AUT" per calcolare gli autovalori: \n'],'s');

x=upper(x);  %riporta la stringa x in caratteri maiuscoli

switch x
     case {'D','1'}
              disp('Il determinante è:')
              D=det(M)
    case {'AUT','2'}
              disp('Gli autovalori sono:')
              AUT=eig(M)
    otherwise
        disp('Numero/stringa non corretta')
end
```

Script: SwitchCase03.m

Esercizio Verificare la congettura di Collatz per un set di numeri naturali di partenza inseriti a piacere dall'utente in un vettore `nv`.
Memorizzare in vari vettori chiamati CSEQ1, CSEQ2, etc. le sequenze corrispondenti

```
clc, clear all

nv=[10000 25000 25001];
j=1;

for n=nv
    i=1;
    eval(['CSEQ' num2str(j) '(i)=n;']);
        while n>1
          if mod(n,2)==1
             n=3*n+1;
          else n=n/2;
          end
        i=i+1;
        eval(['CSEQ' num2str(j) '(i)=n;']);
        end
  j=j+1;
end
```

Script: collatz2.m