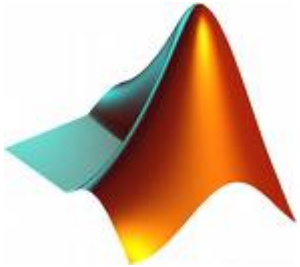


Matlab-Simulink per l'Ingegneria

Slides Lezione 2025
Parte prima

Prof. Alessandro Pisano
apisano@unica.it

Istruzioni utili per gli script

`return` Interrompe l'esecuzione dello script.
Spesso inserito in costrutti condizionali IF

`pause` Interrompe l'esecuzione dello script finché non viene premuto un tasto qualunque. Può servire per ispezionare il valore di alcune variabili durante l'esecuzione dello script, avendo cura di visualizzare il valore di tali variabili nel prompt.

CTLR + C Digitato durante l'esecuzione di uno script, ne interrompe l'esecuzione.
Utile quando ad esempio si compie un errore di programmazione che instaura una condizione di stallo o di loop infinito per il programma.

Esercizio preliminare

Scrittura di una function e predisposizione di uno script che analizza l'output della function

(15 dicembre 2020)

Scrivere un function file che accetti in ingresso un vettore di 10 elementi reali $z = [z_1 \quad z_2 \quad \dots \quad z_{10}]$ e restituisca la matrice quadrata di dimensione 10:

$$A = \begin{bmatrix} 1 & z_1 & \dots & z_1^8 & z_1^9 \\ 1 & z_2 & \dots & z_2^8 & z_2^9 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & z_9 & \dots & z_9^8 & z_{10}^9 \\ 1 & z_{10} & \dots & z_{10}^8 & z_{10}^9 \end{bmatrix}$$

Mediante uno script, si verifichi numericamente che in corrispondenza del vettore di ingresso z=[1. 2, …, 10] la somma degli autovalori di A è uguale alla sua traccia, ossia alla somma degli elementi diagonali. Detta x la soluzione del sistema lineare $Ax = b$, dove b è un vettore colonna con tutti gli elementi unitari, calcolare all'interno dello script la quantità $e = b^T Ax - \|b\|^2$, essendo $\|b\| = \sqrt{b^T b}$ la norma euclidea di b.

$$A = \begin{bmatrix} 1 & z_1 & \dots & z_1^8 & z_1^9 \\ 1 & z_2 & \dots & z_2^8 & z_2^9 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & z_9 & \dots & z_9^8 & z_{10}^9 \\ 1 & z_{10} & \dots & z_{10}^8 & z_{10}^9 \end{bmatrix}$$

La seconda colonna della matrice coincide con il trasposto del vettore passato in ingresso alla funzione

La terza colonna della matrice contiene gli elementi del vettore passato in ingresso alla funzione elevati al quadrato

La decima colonna della matrice contiene gli elementi del vettore passato in ingresso alla funzione elevati alla nona potenza

FUNCTION

```
function Mout = creamatrice(vin)

Mout(:,1)=ones(10,1);

for i=2:10
Mout(:,i)=vin'.^(i-1);
end

end
```

Costruzione della prima colonna

Tutte le successive colonne sono costruibili mediante elevamento a potenza elemento per elemento del vettore vin

SCRIPT

```
clear all
clc

v=1:10;
A=creamatrice(v);

sommaaut=sum(eig(A))

%calcoliamo la traccia in 3 modi diversi
traccia1=trace(A)

traccia2=sum(diag(A))

traccia3=0;
for i=1:10
traccia3=traccia3+A(i,i);
end
traccia3

b=ones(10,1);
x=A\b;

e=(b')*A*x-(b'*b)
```

Somma degli autovalori

Risoluzione sistema lineare Ax=b

Calcolo e

SINTASSI CONDIZIONALE «IF»

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

```
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 (costrutto di base)

```
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 (costrutto con due alternative)

L'istruzione `disp` consente anche di visualizzare nella Command Window 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 (costrutto di base)

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

Esempio 2 (costrutto con due alternative)

```
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 n/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;
M=zeros(N);

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

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

Creo una matrice quadrata di zeri (preallocazione degli array)

Riassegno i valori sulla diagonale principale

Riassegno i valori sulla sopra e sotto-diagonale
N.B. Un elemento (i,j) appartiene alla sopra- o sotto-diagonale se $|i-j|=1$.

Codice alternativo

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

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

costrutto con 3 alternative

Altro codice alternativo (solo ciclo FOR)

```
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
```

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

Rimozione di elementi da un array

E' possibile rimuovere un elemento da un vettore attribuendogli il valore «[]» (che significa «vettore vuoto») tale elemento viene rimosso dal vettore, del quale quindi viene ridotta di 1 la dimensione.

Definiamo un vettore `V` di test con 5 elementi e Rimuoviamo dal vettore **l'ultima componente**

Ciò può essere fatto mediante la seguente sintassi

```
V=[3 1 7 4 5]
V(end)=[]
```

```
V =

     3     1     7     4     5

V =

     3     1     7     4
```

Rimozione simultanea della **seconda e terza componente**

```
w=[1 3 5 7 9 11]
w([2 3])=[]
```

```
w =

     1     3     5     7     9    11

w =

     1     7     9    11
```

Modifica di array

Se si assegna un valore ad una componente di un vettore con indice più elevato della dimensione del vettore **la dimensione del vettore viene corrispondentemente aumentata e tutti i nuovi valori aggiunti, eccetto l'ultimo, sono posti pari a zero.**

Definiamo un vettore `V` di test con 4 elementi e poi attribuiamo un valore alla decima componente del vettore.

```
V=[1 2 3 4]
V(10)=7
```

```
V =

     1     2     3     4


V =

     1     2     3     4     0     0     0     0     0     7
```

Tempo [s]	Pressione [bar]
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 da uno stream di dati al di sotto di un valore di soglia

Carichiamo nel workspace, con il comando

```
load stream
```

il file stream.mat

Esso provoca il salvataggio nel workspace di due vettori, time e data che contengono rispettivamente istanti di acquisizione e misure acquisite (misure di pressione acquisite ogni decimo di secondo per 3 secondi, quindi 31 acquisizioni)

Si notino due misure nulle (poste in 14esima e 26esima posizione), che denotano un errore di acquisizione.

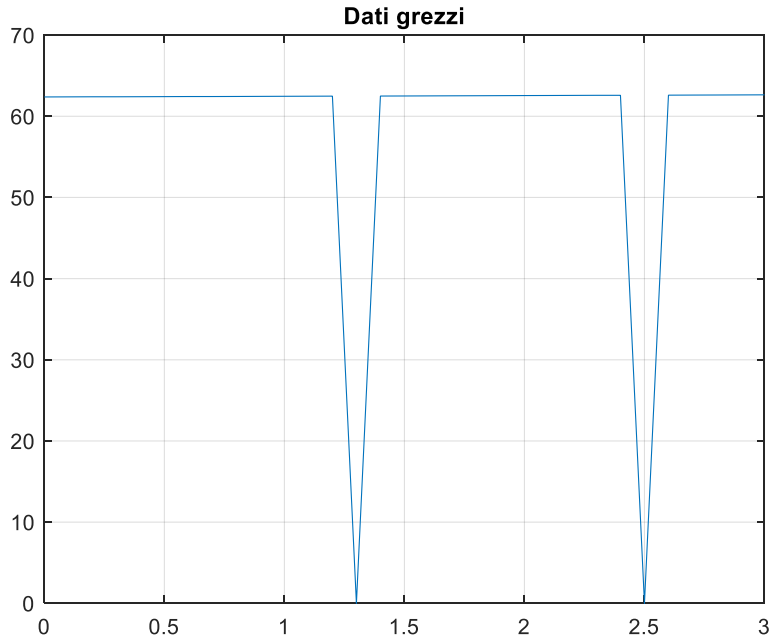
Desideriamo rimuovere dallo stream le misure inferiori a 10 bar che ove andassi a processare i dati (ad esempio per estrarne il valore minimo) causerebbero errori.

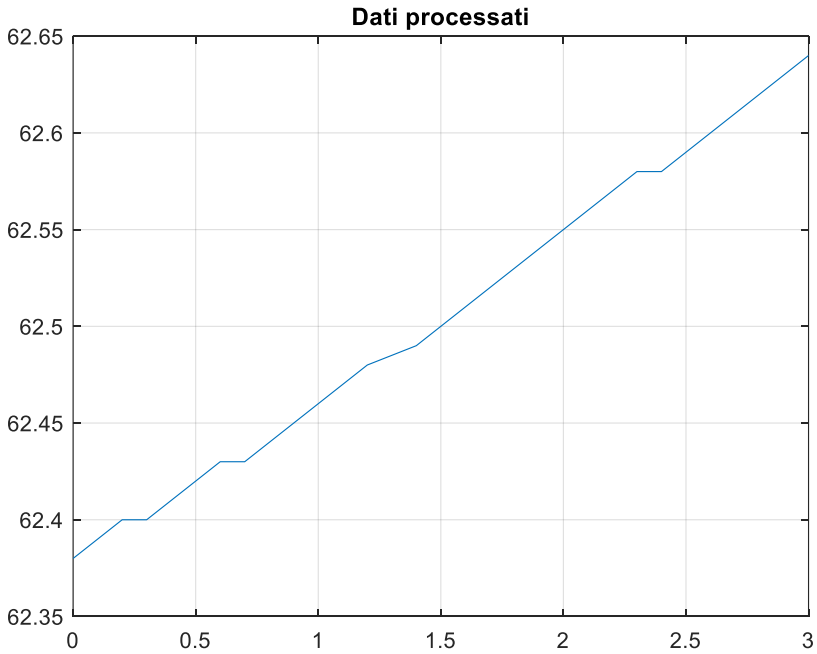
Script

```
clear all
clc

load stream

data_filt=data;
time_filt=time;

soglia=10;

for i=1:length(data)
for i=length(data):-1:1

  if data(i)<soglia
    data_filt(i)=[];
    time_filt(i)=[];
  end

end
```

modo errato di costruire il ciclo FOR

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





Operatori relazionali

<	*minore*
<=	*minore o uguale*
>	*maggiore*
>=	*maggiore o uguale*
==	*uguale*
~=	*diverso*

~ → Alt+126

Possono essere applicati fra un array ed uno scalare, oppure fra due array delle medesime dimensioni

Nel primo caso, gli elementi dell'array vengono tutti confrontati con lo scalare.
Nel secondo caso vengono confrontati gli elementi con posizione omologa.
In entrambi i casi, si ottiene un array di valori booleani (0 ed 1).

Frequentemente utilizzati nella condizione di attivazione di costrutti sintattici `if` o `while`

Vediamo esempi di applicazione delle varie casistiche e operatori.

Confronto fra scalari

```
x=1; y=3;
B1=x<=y
```

```
B1 =

  logical

   1
```

Confronto fra vettore e scalare

```
v=[1 2 3 4 5]
Boolv=v<3
```

```
v =

     1     2     3     4     5


Boolv =

  1×5 logical array

   1   1   0   0   0
```

Confronto fra vettori

```
v1=[1 2 3 6 5]
v2=[1 2 3 8 5]
Boolv2=v1~=v2
```

Istruzione di assegnazione

Operatore Relazionale di diversità

```
v1 =

     1     2     3     6     5


v2 =

     1     2     3     8     5


Boolv2 =

  1×5 logical array

   0   0   0   1   0
```

Confronto fra matrice e scalare

```
M=[1 2 3;4 5 6;7 4 4]
BoolM=M==4
```

Istruzione di assegnazione

Operatore Relazionale di uguaglianza

```
M =

     1     2     3
     4     5     6
     7     4     4


BoolM =

  3×3 logical array

   0   0   0
   1   0   0
   0   1   1
```

Confronto fra matrici

```
A1=[10 5; 1 1]
A2=[2 3;4 5]


BoolA12=A1>A2
```

```
A1 =

    10     5
     1     1


A2 =

     2     3
     4     5
```

```
BoolA12 =

  2×2 logical array

   1   1
   0   0
```

Funzioni logiche

***any* (*x*)**	*1 se almeno un elemento di x è non nullo, zero altrimenti.*
***all* (*x*)**	*1 se tutti gli elementi di x sono non nulli, zero altrimenti*
***find* (*x*)**	*restituisce gli indici degli elementi non nulli del vettore in ingresso*

```
x=[4 1 0 2 6 0]

a1=any(x)
a2=all(x)
a3=find(x)
```

```
a1 =

  logical

   1


a2 =

  logical

   0


a3 =

     1     2     4     5
```

Con input matriciale, `any` ed `all` lavorano separatamente sulle colonne, mentre `find` restituisce le posizioni di tutti gli elementi non nulli con una notazione «particolare»

```
x=[11 12 13;4 0 5; 9 8 0]
a1=any(x)
a2=all(x)
a3=find(x)
```

```
x =

    11    12    13
     4     0     5
     9     8     0
```

```
a1 =

  1×3 logical array

   1   1   1
```

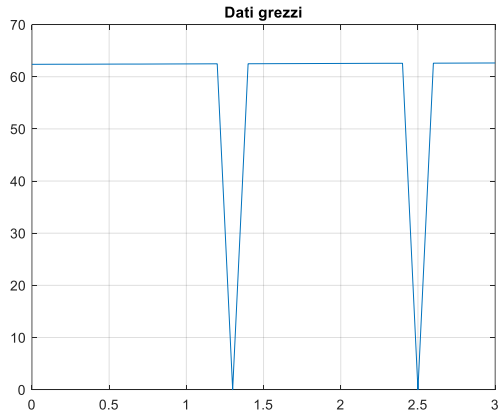
```
a2 =

  1×3 logical array

   1   0   0
```

```
a3 =

     1
     2
     3
     4
     6
     7
     8
```

Tempo [s]	Pressione [bar]
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
1.3000	0
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
2.5000	0
2.6000	62.6000
2.7000	62.6100
2.8000	62.6200
2.9000	62.6300
3	62.6400

Rimozione da uno stream di dati al di sotto di un valore di soglia (stesso esempio trattato in precedenza)

```
clear all, clc
load stream

figure(1)
plot(time,data),grid
title('Dati grezzi')
```




```
soglia=10;
z1=data<soglia;
```

z1 è un vettore di elementi booleani, che contiene 1 in corrispondenza dei dati sottosoglia da scartare (il 14-esimo ed il 26-esimo), e zero altrove

```
z1 =

  31×1 logical array

   0
   0
   0
   0
   0
   0
   0
   0
   0
   0
   0
   0
   1
   0
   ⋮
   1
   0
   0
   0
   0
   0
```

Con la funzione find estraiamo gli indici associati alle posizioni di tali misure «sottosoglia».

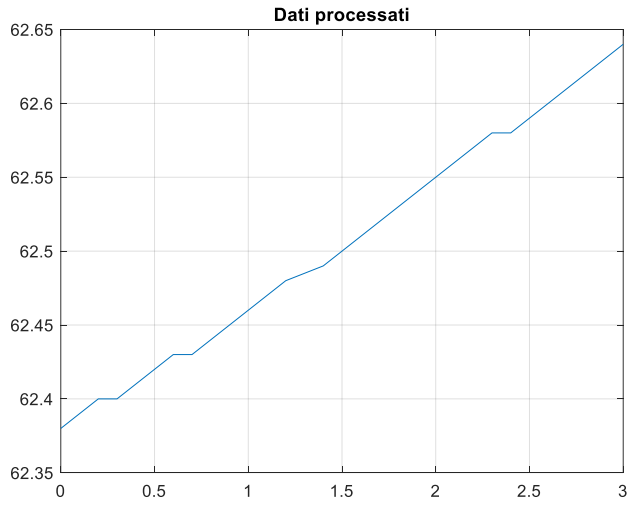
```
indici=find(z1)
```

```
indici =

    14
    26
```

```
data(indici)=[];
time(indici)=[];
```

Queste due istruzioni, secondo quanto visto in precedenza, rimuovono dai vettori `data` e `time` la 14esima e la 26esima componente. Ora i vettori hanno quindi dimensione 29, non più 31.

```
figure(2)
plot(time,data),grid
title('Dati processati')
```



Il grafico dei dati processati mostra come le misure «spurie» siano state rimosse dai dati.

Operatori logici

~	*NOT*
&	*AND*
|	*OR*

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 stringa 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
```

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
```

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: versione con FOR e IF

Function: versione compatta con diag e triu

```
function M = creaM(n)

A = diag(n*ones(1,n))+diag(2*ones(1,n-1),1)+diag(2*ones(1,n-1),-1)
B=triu(3*ones(n),0)
M=[A B;B' A];

end
```

```
>> 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

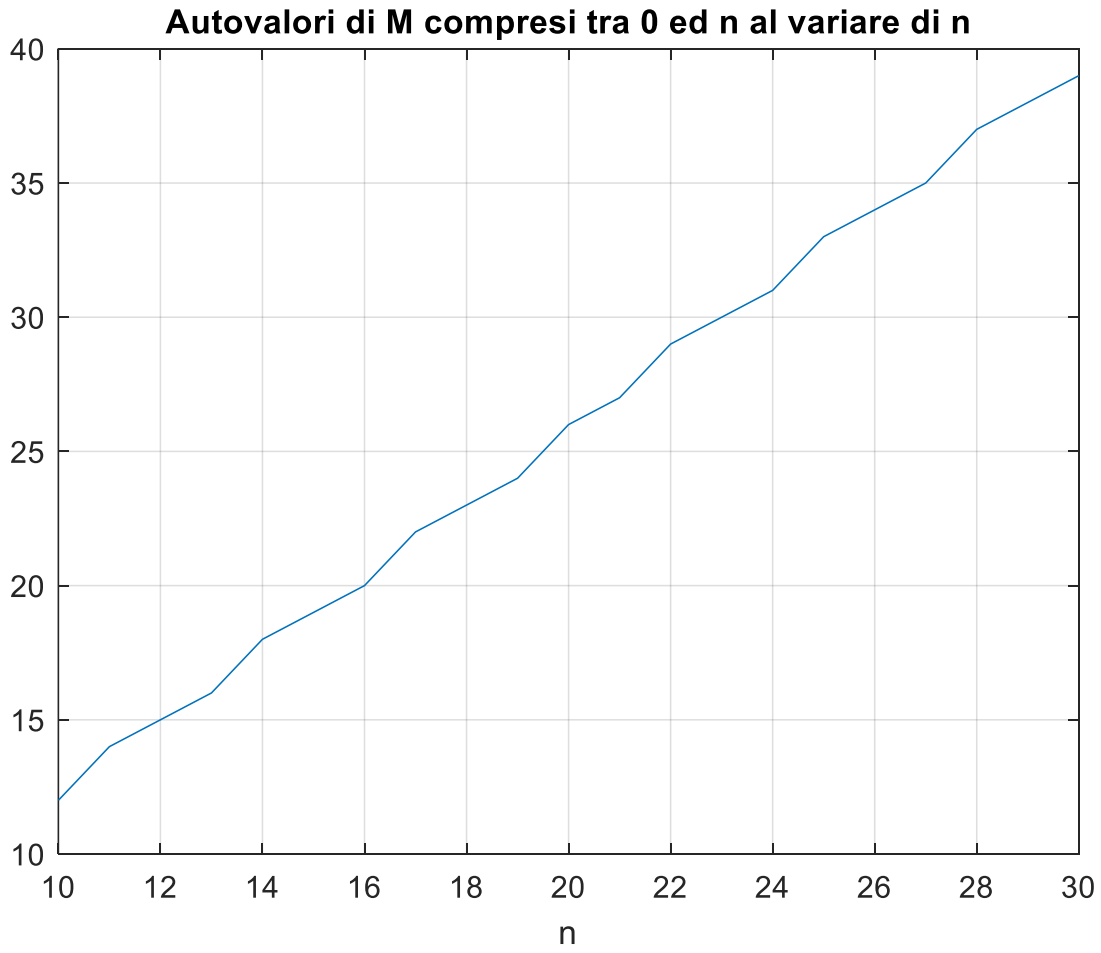
```
clc,
i=10:1:30;
ind=1;

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
```

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

```
plot(10:30,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 finché “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

Esercizio

Calcolo iterativo di un parametro caratterizzante il funzionamento di un compressore centrifugo

Sono assegnati i parametri costanti

r_2, r_3 c_{2r}, c_{3u} γ, R, C_p $\dot{m}$

T_2, T_{30} p_2 b_2

Deve essere determinato il parametro c_{3r}, che rappresenta la componente radiale della velocita del flusso d'aria in una sezione di un compressore multistadio

Per tenere in considerazione le variazioni della densità (che dipende da pressione e temperatura) il parametro c_{3r} va dapprima «inizializzato» impiegando la seguente formula

$$c_{3r} = c_{2r}\frac{r_2}{r_3}$$

e successivamente «ricalcolato» più volte, onde migliorare la precisione, attraverso l'esecuzione iterativa di un set di istruzioni, da compiersi finché la differenza fra i valori del parametro ottenuti in due iterazioni successive non sia sufficientemente piccola, segno del fatto che la procedura di calcolo è andata a convergenza.

Dati di ingresso (noti) r_2, r_3 c_{2r}, c_{3u} γ, R, C_p $\dot{m}$

T_2, T_{30} p_2 b_2

$$c_{3r} = c_{2r}\frac{r_2}{r_3}$$ Inizializzazione del parametro

$$c_3 = \sqrt{c_{3u}^2 + c_{3r}^2}$$

$$T_3 = T_{30} - \frac{c_3^2}{2C_p}$$

$$p_3 = p_2\left(\frac{T_3}{T_2}\right)^{\frac{\gamma}{\gamma-1}}$$

$$\rho_3 = \frac{p_3}{RT_3}$$

$$c_{3r} = \frac{\dot{m}}{2\,\pi r_3 \rho_3 b_2}$$

Calcolo iterativo

Dati di ingresso (noti) r_2, r_3 c_{2r}, c_{3u} γ, R, C_p $\dot{m}$

T_2, T_{30} p_2 b_2

$$c_{3r,1} = c_{2r}\frac{r_2}{r_3}$$ Inizializzazione del parametro

$$c_3 = \sqrt{c_{3u}^2 + c_{3r,i}^2}$$

$$T_3 = T_{30} - \frac{c_3^2}{2C_p}$$

$$p_3 = p_2\left(\frac{T_3}{T_2}\right)^{\frac{\gamma}{\gamma-1}}$$

$$\rho_3 = \frac{p_3}{RT_3}$$

$$c_{3r,i+1} = \frac{\dot{m}}{2\,\pi r_3 \rho_3 b_2}$$

Calcolo iterativo [i=1,2,3,.....]

Arrestiamo le iterazioni quando:

$$\left|c_{3r,i+1} - c_{3r,i}\right| \leq 10^{-4}$$

Realizziamo uno script che calcoli il parametro cercato con una accuratezza di 4 cifre decimali

Assegnazione parametri costanti

r_2, r_3 c_{2r}, c_{3u} γ, R, C_p $\dot{m}$

T_2, T_{30} p_2 b_2

```
clear all
clc

r2=0.2449;
r3=0.2694;
c2r=146.2950;
c3u=306.9331;

gamma=1.4;
R=287;% costante universale dei gas [J/(kg*K)]

Cp=1005; % calore specifico a p=cost [J/(kg*K)]
mpunto=7.5; % portata massica [kg/s]

T2=351.6722;
T30= 419.0365;
p2= 1.7115e+05;
b2=0.0196;
```

Desideriamo memorizzare in un vettore tutti i successivi valori di c_{3r} calcolati alle varie iterazioni, in modo da poter analizzare la rapidità del processo di convergenza del parametro

Inizializzazione e calcolo iterativo del parametro c_{3r}

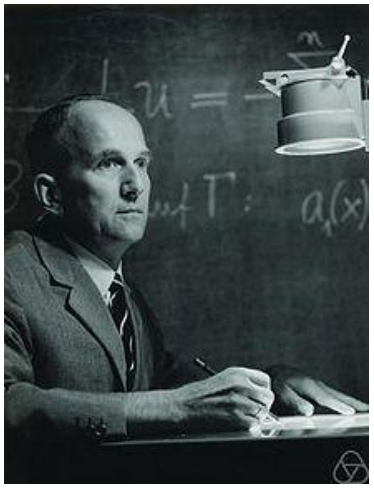
```
c3r(1)=0;
c3r(2)=c2r*r2/r3;

i=2;

while abs(c3r(i)-c3r(i-1))>0.0001
  c3=sqrt(c3r(i)^2+c3u^2);
  T3=T30-c3^2/(2*Cp);
  P3=p2*(T3/T2)^(gamma/(gamma-1));
  rho3=P3/(R*T3);
  c3r(i+1)=mpunto/(2*pi*r3*rho3*b2);
  i=i+1;
end

c3r_FINALE=c3r(end)
C3r
```

```
c3r_FINALE =

  121.6356
```

```
c3r =

         0  132.9905  122.8429  121.7581  121.6480  121.6368  121.6357  121.6356  121.6356
```

Script:
ParametroCompressore.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

MathPrize

POSTS

Collatz conjecture Prize 120 million JPY

July 7, 2021

A prize of 120 million JPY will be paid to those who have revealed the truth of the Collatz conjecture. The conjecture is also known as the $3x+1$ problem or the $3n+1$ problem.

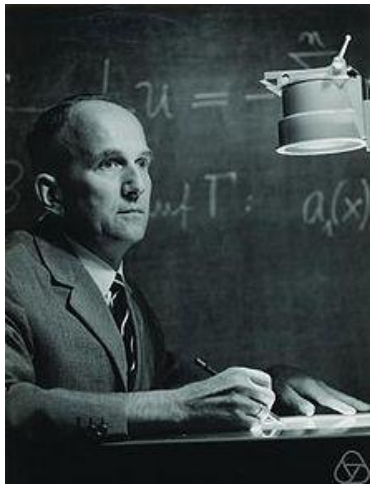
120 million JPY in USD

Collatz conjecture

Repeatedly applying the function $f(x)$ defined below to any positive integers will eventually result in 1.

$$f(x) = \begin{cases} x/2 & (x \equiv 0 \pmod 2) \\ 3x+1 & (x \equiv 1 \pmod 2) \end{cases}$$

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(CSEQ);  %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: collatz.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
```

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
```