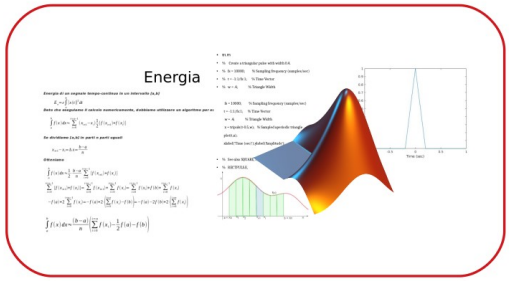
UNIVERSITÀ DEGLI STUDI DI TRIESTE

Programmazione Informatica

Ingegneria Civile e Ambientale
Ingegneria Navale

Matlab 8 – Automi a stati finiti – Costrutto switch – Manipolazione di grafici

Antonio Fiumara
antonio.fiumara@dia.units.it



A. A. 2025-26

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Struttura del Corso

Gli argomenti trattati nel corso saranno:
Lezione Teoria 1 – Introduzione, presentazione del corso, codifica dell'informazione
Lezione Teoria 2 – Rappresentazione dell'informazione in un computer
Lezione Teoria 3 – Algebra booleana, introduzione e applicazioni nella programmazione
Lezione Teoria 4 – Architettura dei calcolatori. HW, SW, Sistemi Operativi
Lezione Teoria 5 – Algoritmi e strutture dati - Implementazione di algoritmi notevoli (ordinamento , ricerca, ecc)
Lezione Programmazione - Matlab 1 – Introduzione, installazione, comandi di base, matrici
Lezione Programmazione - Matlab 2 – Input/Output (I/O), strutture di controllo
Lezione Programmazione - Matlab 3 – Funzioni, ciclo for
Lezione Programmazione - Matlab 4 – Grafici. Sistemi Lineari, Algebra lineare, Analisi matematica
Lezione Programmazione - Matlab 5 – Strutture dati avanzate. Assegnazione esercitazione di gruppo n.1.
Lezione Programmazione - Matlab 6 – Funzioni nel dominio del tempo e della frequenza. Analisi ed elaborazione dei segnali. Assegnazione esercitazione di gruppo n.2
Lezione Programmazione - Matlab 7 – Applicazioni ingegneristiche avanzate
Lezione Programmazione - Matlab 8 – Correzione esercitazioni di gruppo. Ripasso e approfondimenti

Esonero esame Programmazione Matlab (3/12/25)

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Introduzione FSM (finite state machine)

Introduzione con un esempio (bistabile - serbatoio)

Formalizzazione FSM (finite state machine)

Diagramma degli stati

Esempio di applicazione con la GUI

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Introduzione FSM (finite state machine)

Simulazione di un interruttore bi-stabile

Realizzare un programma che a fronte della pressione del tasto ‘u’, simuli l’accensione di una lampada (output mediante la stringa ***lampada***)
Inizialmente la lampada sarà spenta (lampada = “luce spenta”)
Se premo il tasto ‘u’, se la lampada è spenta dovrà accendersi, se è accesa dovrà spegnersi

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Introduzione FSM (finite state machine)

Premessa

Per gestire la pressione di un tasto senza dover premere invio, come nel caso di input(), sfruttiamo due funzioni di manipolazione dei grafici

Figure;
w = waitforbuttonpress;
axes;

e

Get()

Creiamo una funzione getcher.m che ricorda l'omonima funzione getchar() del C

```
function c=getchar()
figure(1);
w = waitforbuttonpress;
    if w
    |  c = get(gcf, 'CurrentCharacter');
    end
end
```

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Introduzione FSM (finite state machine)

Premessa

Prova della funzione che abbiamo creato

```
clear all;

while 1
  p = getchar();
          if p=='0' break
            end

          disp(p) %displays the character that was pressed
         disp(double(p))  %displays the ascii value of the chara
end
```

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Introduzione FSM (finite state machine)

Simulazione di un interruttore bi-stabile

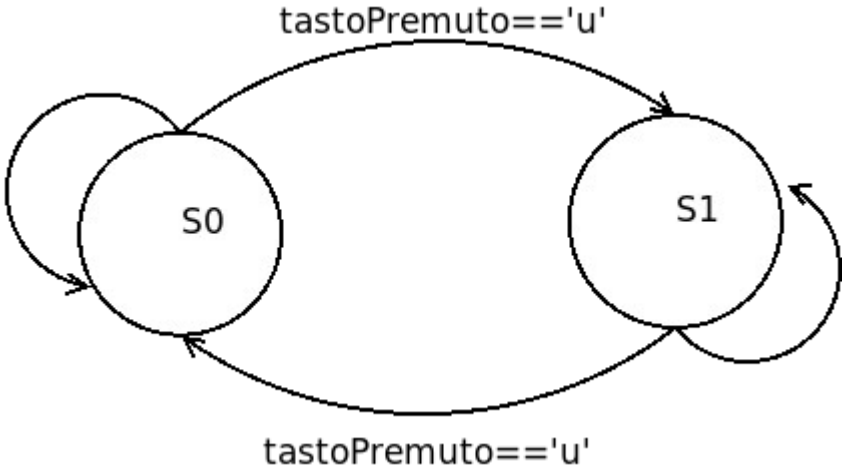
Realizzare un programma che a fronte della pressione del tasto ‘u’, simuli l’accensione di una lampada (output mediante la stringa ***lampada***)
Inizialmente la lampada sarà spenta (lampada = “luce spenta”)
Se premo il tasto ‘u’, se la lampada è spenta dovrà accendersi, se è accesa dovrà spegnersi

```
clear all;
stato=0;
lampada="luce spenta";
while 1
   p=getchar();
       if stato==0
           if p=='u';
               stato=1;
               lampada = "luce accesa"; % Change the state of 'luce' to
           end
       elseif stato==1
            if p=='u';
                 stato=0;
                 lampada = "luce spenta"; % Change the state of 'luce'
            end
        end
             if p==48 break
             end
    disp(lampada) %displays the character that was presse
end
```

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Introduzione FSM (finite state machine)

Il nostro “processo” ha due “**stati**”, e due valori possibili per l’uscita **lampada**, “luce accesa” e “luce spenta”. l’input è “immissione continua di un carattere” memorizzato nella variabile **tastoPremuto**.
Il grafo orientato che raffigura in maniera schematica questo processo è detto **diagramma degli stati**.
I cerchi rappresentano gli stati (un cerchio per ciascun stato), gli archi orientati rappresentano le transizioni da uno stato ad un altro a causa di un **evento**.



UNIVERSITÀ DEGLI STUDI DI TRIESTE

Introduzione FSM (finite state machine)

Al posto della catena di if elseif … è possibile utilzzare il costrutto switch case

```
clear all;
stato=0;
lampada="luce spenta";
while 1
     p = getchar();
           switch stato
               case 0
                   if p=='u'
                       stato=1;
                       lampada = "luce accesa"; % Change the state of 'luc
                   end
               case 1
                   if p=='u'
                       stato=0;
                       lampada = "luce spenta"; % Change the state of 'luc
                   end
           end
           if p==48 break
           end
           disp(lampada+" stato="+stato) %displays the character that was
end
```

UNIVERSITÀ DEGLI STUDI DI TRIESTE

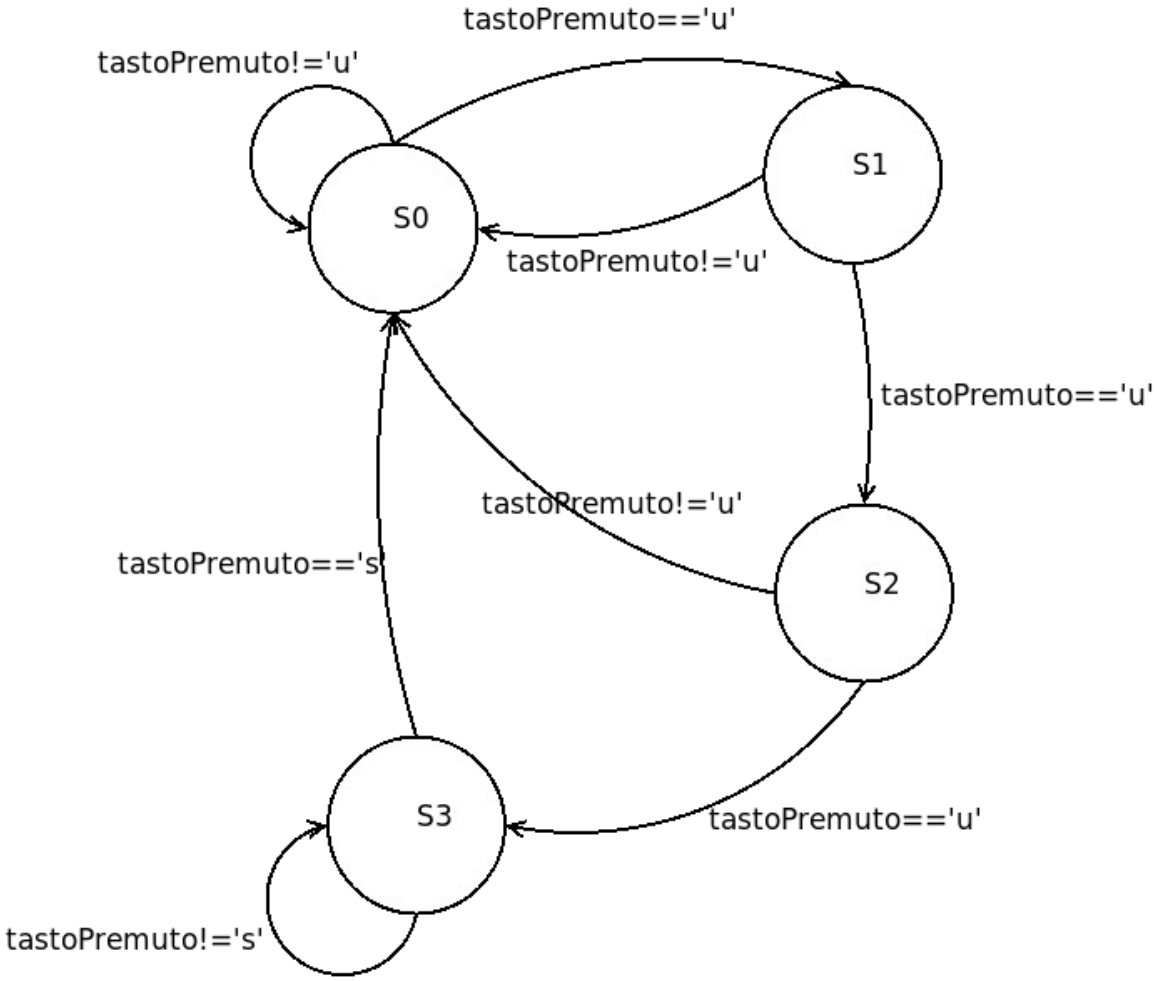
FSM (finite state machine) altro esempio

Simulazione di un interruttore bi-stabile

Introduciamo un po' di complessità nel comportamento della FSM.
Realizzare un programma che simuli l'accensione di una lampada (output mediante la stringa lampada) a fronte della pressione del tasto 'u' per tre volte consecutive.
Se la sequenza è interrotta bisogna ripartire dall'inizio. Se la lampada è accesa, la pressione del tasto 's' ne causerà lo spegnimento.
Inizialmente la lampada sarà spenta (lampada = "luce spenta")

UNIVERSITÀ DEGLI STUDI DI TRIESTE

FSM (finite state machine) altro esempio



UNIVERSITÀ DEGLI STUDI DI TRIESTE

Approfondimenti e manipolazione di grafici

figure()

- figure crea una nuova finestra della figura utilizzando i valori predefiniti delle proprietà. La figura risultante è la figura corrente.

- figure(Name,Value) modifica le proprietà della figura utilizzando uno o più argomenti della coppia nome-valore.
  - Ad esempio, figure('Color','white') imposta il colore dello sfondo sul bianco.

- esempio

- f = figure(___) restituisce l'oggetto Figure.
  Utilizzare f per eseguire una query o modificare le proprietà della figura dopo averla creata.

- esempio

- figure(f) rende la figura specificata da f la figura corrente e la visualizza sopra tutte le altre figure.

- esempio

- figure(3) trova una figura in cui la proprietà Number è uguale a 3 e la rende la figura corrente. Se non è presente una figura con quel valore di proprietà, MATLAB® crea una nuova figura e imposta la relativa proprietà Number su n.

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Approfondimenti e manipolazione di grafici

N.B. Per salvare i grafici su disco è possibile utilizzare la funzione

saveas(fig, 'nomefile.png')

Dove fig è l'identificativo del grafico ottenuto come valore di ritorno della funzione figure(),

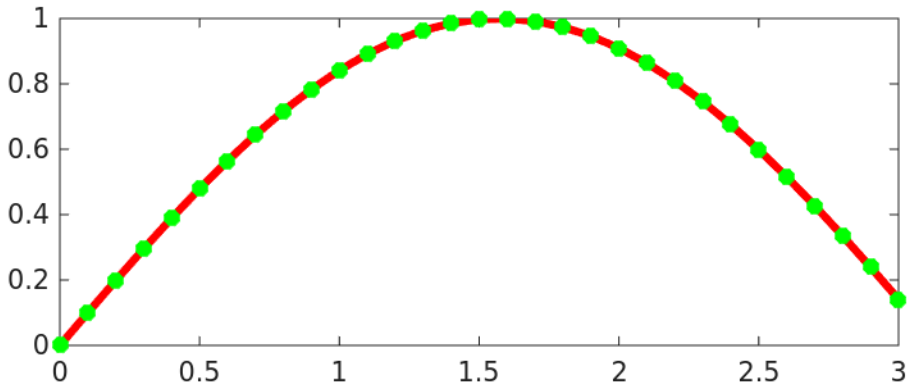
Esempio
fig1=figure(1)
plot(x,y)
saveas(fig1, 'grafico1.png')

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Approfondimenti e manipolazione di grafici

p = plot() *restituisce un oggetto* ***Line*** *o un* ***array di oggetti Line****. Utilizzare* ***p*** *per modificare le proprietà del grafico dopo averlo creato. Per un elenco delle proprietà, vedere* ***Line Properties****.*

```
x = 0:0.1:3;
y = sin(x);
fig=figure(1);
ph=plot(x,y);
ph.LineWidth = 3;
colore=ph.Color;
ph.Color = 'r';
ph.Marker = '*';
ph.MarkerEdgeColor = 'g';
ph.MarkerFaceColor = 'b';
saveas(fig,'esempio.png')
```



Colore =

0.0660 0.4430 0.7450

```
ph.Color = colore+[0 .2 0];
```

https://it.mathworks.com/help/matlab/ref/matlab.graphics.chart.primitive.line-properties.html

Approfondimenti e manipolazione di grafici

Possiamo cambiare il colore della linea

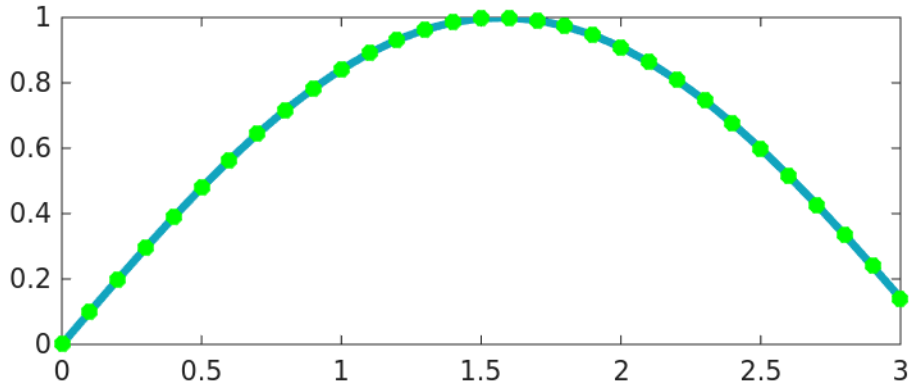
ph.Color = 'b';

Oppure modificare il singolo valore RGB
colore=ph.Color;
Esempio, aumentiamo il canale G

Colore =

0.0660 0.4430 0.7450

ph.Color = colore+[0 .2 0];



https://it.mathworks.com/help/matlab/ref/matlab.graphics.chart.primitive.line-properties.html

UNIVERSITÀ DEGLI STUDI DI TRIESTE

Energia

Energia di un segnale tempo-continuo in un intervallo [a,b]

$$E_x = \varepsilon \int_a^b |x(t)|^2 dt$$

Dato che eseguiamo il calcolo numericamente, dobbiamo utilizzare un algoritmo per e:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{i=n-1} (x_{i+1}-x_i)\frac{1}{2}[f(x_{i+1})+f(x_i)]$$

Se dividiamo [a,b] in parti n parti uguali

$$x_{i+1}-x_i = \Delta x = \frac{b-a}{n}$$

Otteniamo

$$\int_a^b f(x)dx \approx \frac{1}{2}\cdot\frac{b-a}{n}\sum_{i=0}^{i=n-1}[f(x_{i+1})+f(x_i)]$$

$$\sum_{i=0}^{i=n-1}[f(x_{i+1})+f(x_i)] = \sum_{i=0}^{i=n-1} f(x_{i+1}) + \sum_{i=0}^{i=n-1} f(x_i) = \sum_{i=0}^{i=n-1} f(x_i) + f(b) + \sum_{i=0}^{i=n-1} f(x_i)$$

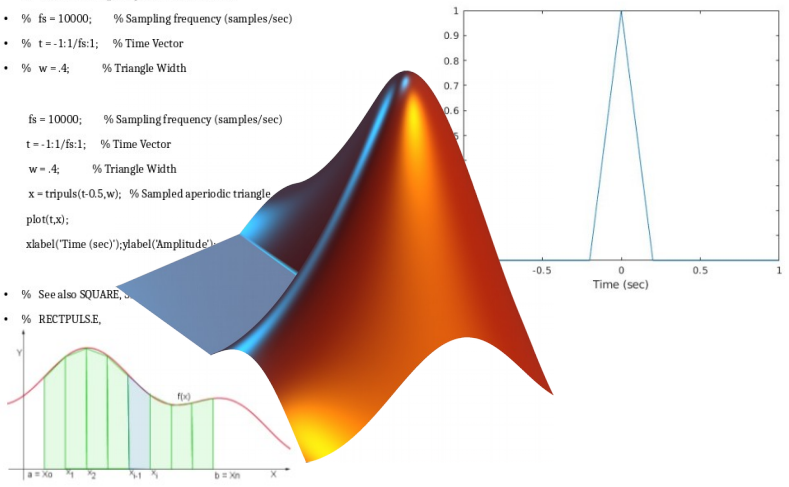
$$-f(a)+2\sum_{i=0}^{i=n-1} f(x_i) = -f(a)+2\left(\sum_{i=0}^{i=n} f(x_i) - f(b)\right) = -f(a)-2f(b)+2\left(\sum_{i=0}^{i=n} f(x_i)\right)$$

$$\int_a^b f(x)dx \approx \frac{(b-a)}{n}\left(\sum_{i=0}^{i=n} f(x_i) - \frac{1}{2}f(a) - f(b)\right)$$

- tri.m
- % Create a triangular pulse with width 0.4.
- % fs = 10000; % Sampling frequency (samples/sec)
- % t = -1:1/fs:1; % Time Vector
- % w = .4; % Triangle Width

```
fs = 10000;      % Sampling frequency (samples/sec)
t = -1:1/fs:1;   % Time Vector
w = .4;          % Triangle Width
x = tripuls(t-0.5,w);  % Sampled aperiodic triangle
plot(t,x);
xlabel('Time (sec)');ylabel('Amplitude')
```

- % See also SQUARE, S
- % RECTPULSE,



https://it.mathworks.com/videos/matlab-overview-61923.html