



UNIVERSITÀ  
DEGLI STUDI  
DI PADOVA

Laboratorio 2  
Rappresentazione di segnali a tempo  
continuo



In Matlab un segnale può essere rappresentato:

1. Tramite una funzione che ne effettua il calcolo
2. Tramite un vettore di valori assunti in corrispondenza di un opportuno vettore di valori della variabile indipendente

## Esempio 1:

```
>> x1 = @(t) sin(2*pi*t);  
>> x1(1/3)  
ans =  
    0.8660
```

@(t) significa che stiamo definendo una funzione di t.  
È la versione *on-line* di  
function y=x1(t)  
y=sin(2\*pi\*t);

Eseguite il codice  
mostrato

## Esempio 2:

```
>> t = 0:1e-4:1;  
>> x2 = sin(2*pi*t);  
>> x2(3334)  
ans =  
    0.8661
```

Qui invece stiamo definendo due vettori: la « variabile indipendente » t ed i valori assunti x2  
Notare che potremmo sostituire  $x2 = \sin(2\pi t)$ ; con  $x2 = x1(t)$ ;  
Per calcolare approssimativamente  $\sin\left(\frac{2\pi}{3}\right)$  bisogna prendere l'indice 3334 che corrisponde a  $t=0.3333$



Per fare i grafici dobbiamo **sempre** usare dei vettori

Tuttavia i vettori sono dei dati in forma discreta

Per ottenere una rappresentazione *accettabile* dei segnali a tempo continuo bisogna che il vettore rappresenti un campionamento *abbastanza fitto* dell'asse temporale (In corsivo vedete dei termini che hanno un significato empirico)

Osserviamo che Matlab userà un'interpolazione lineare tra i valori consecutivi calcolati (cioè, li « unisce con un segmento »)



# Esercizio 1: Rappresentazione di una sinusoide

Creare lo script es1.m

Effettuare le seguenti operazioni:

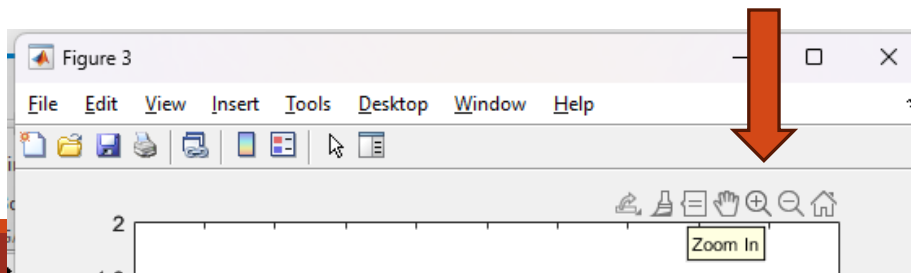
1. Creare un vettore dei tempi che va da 0 a 10 a passo 1
2. Creare un vettore dei tempi che va da 0 a 10 a passo 1/10
3. Creare un vettore dei tempi che va da 0 a 10 a passo 1/100
4. Calcolare e tracciare  $x = \sin 2\pi t$  usando i tre vettori



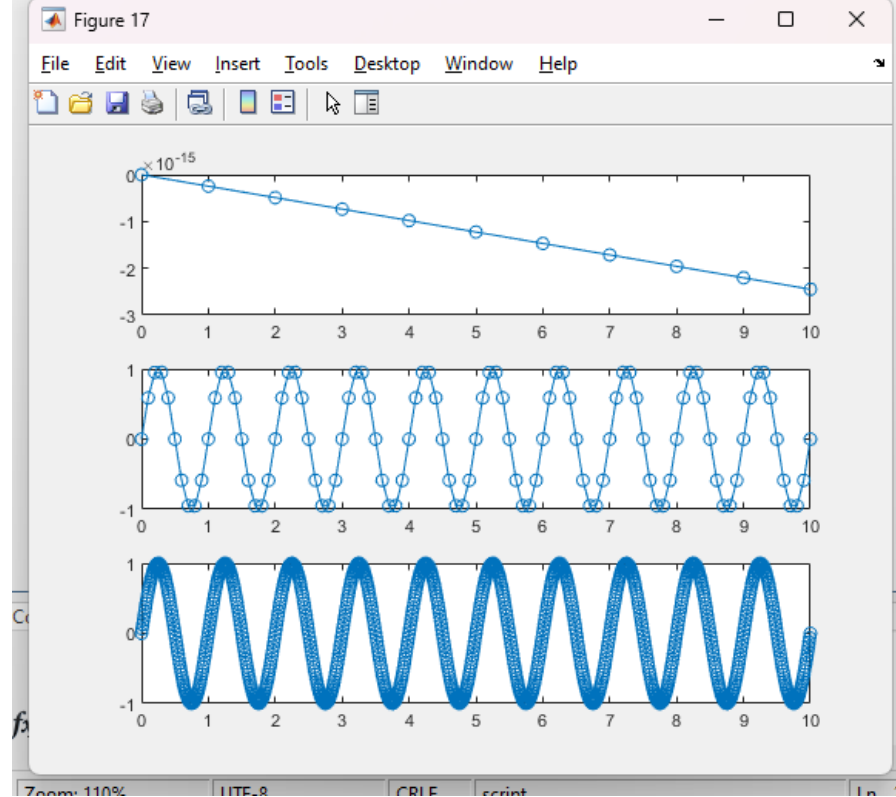
L'opzione `'-o'` indica a Matlab di tracciare un cerchio in ogni punto e di unire i punti con dei tratti continui.

Osserviamo che prendere i campioni a passo unitario fa perdere completamente l'andamento del segnale.

Provate a ingrandire (zoom in) i subplot 2 e 3: si vedranno i tratti di retta usati da Matlab per creare il grafico. Per fare lo zoom, portate il puntatore sul bordo in alto a destro della figura: apparirà un menù:



```
1 t1 = 0:10;  
2 t2 = 0:1e-1:10;  
3 t3 = 0:1e-2:10;  
4  
5 x1 = sin(2*pi*t1);  
6 x2 = sin(2*pi*t2);  
7 x3 = sin(2*pi*t3);  
8  
9 figure(1);  
10 subplot(311); plot(t1,x1, '-o');  
11 subplot(312); plot(t2,x2, '-o');  
12 subplot(313); plot(t3,x3, '-o');
```





## Esercizio 2: Rappresentazione di *tanh*

Si consideri il segnale a tempo continuo  $x(t) = \tanh(t)$

- 1) Creare un vettore  $t$  che rappresenta l'intervallo di valori per il quali calcolare il valore del segnale. Ci sono due opzioni: capire la differenza tra i due casi

```
t=-10:1e-2:10;
```

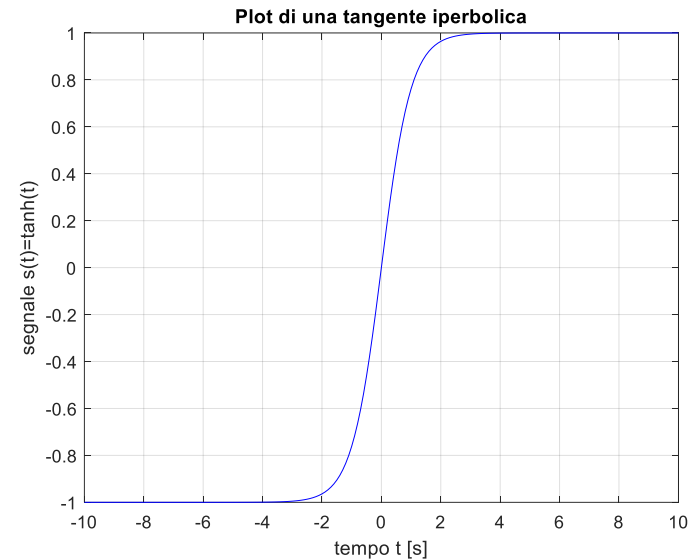
```
t = linspace(-10,10,1001);
```

Per capire il funzionamento di `linspace`, scrivere `help linspace` oppure `doc linspace` e leggere la documentazione

Ricordare che quando si calcola una funzione usando il vettore  $t$  come argomento, l'uscita è un vettore della stessa dimensione in cui ogni elemento è il valore della funzione valutata nel corrispondente valore di  $t$

- 2) Si tracci il segnale  $x(t)$  in funzione di  $t$  con linea continua blu ('b') per  $-10 \leq t \leq 10$

Usare `title`, `xlabel`, `ylabel` e `grid`





## Esercizio 3: Traslazione di un segnale

Si consideri il segnale a tempo continuo  $x(t) = \tanh(t)$

Si crei uno script denominato `es3.m` che tracci sullo stesso grafico i segnali traslati

$$s_1(t) = x(t - b)$$

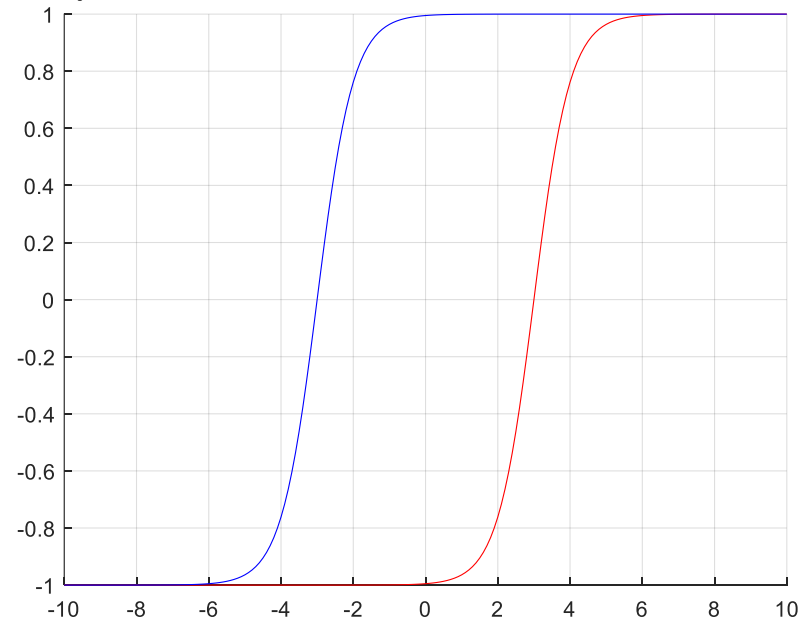
$$s_2(t) = x(t + b)$$

con linea continua rossa ('r') e blu ('b') rispettivamente (si inserisca una legenda nel grafico).  
Usare un ritardo  $b=3$

Per tracciare più curve sullo stesso grafico ci sono 2 possibilità:

- a. dare un unico comando plot per tutte le curve:  
`plot(t, s1, 'r', t, s2, 'b')`
- b. attivare l'opzione «hold» per permettere la sovrascrittura di un grafico  
`plot(t, s1, 'r');`  
`hold on;`  
`plot(t, s2, 'b');`

Ricordare di disattivare hold dopo aver tracciato l'ultima curva: `hold off`





# Soluzione

```
1 clear variables
2 close all
3 clc
4 %% Tangente iperbolica
5 t = -10:0.1:10;
6 s = tanh(t);
7 figure(1)
8 plot(t,s,'-b')
9 grid
10 xlabel('tempo t [s]')
11 ylabel('segnale s(t)=tanh(t)')
12 title('Plot di una tangente iperbolica')
13 %%
14 b = 3;
15 s1 = tanh(t-b);
16 s2 = tanh(t+b);
17 figure(2)
18 plot(t,s1,'-r',t,s2,'-g')
19 grid
20 xlabel('tempo t [s]')
21 legend('traslazione dx','traslazione sx')
22 %% Modo alternativo per tracciare 2 curve
23 figure;
24 hold on;
25 p1=plot(t,s1);
26 set(p1,'Color','red');
27 p2=plot(t,s2);
28 set(p2,'Color','blue');
29 hold off
30 grid on
31 %%
```

Cancella tutte le variabili nel workspace;  
Chiude tutte le figure  
Pulisce la command window

Crea una griglia

Eseguite il codice  
mostrato

hold on  
mantiene tutte le  
curve sul grafico  
corrente,  
impedendo che il  
successivo plot le  
cancelli  
hold off  
disattiva questo  
comportamento

Crea la leggenda: gli argomenti  
sono associati alle curve  
nell'ordine in cui sono tracciate

p1 è un puntatore all'oggetto creato dal  
plot (la curva). Può essere usato per  
cambiare gli attributi della linea, come  
colore, spessore, tipo di linea, tipo di  
marker





# Esercizio 4: Segnali Sinusoidali a Tempo Continuo

In uno script denominato `es4.m` eseguire le operazioni qui descritte.

Si considerino i segnali sinusoidali:

$$x(t) = \cos(\omega_0 t + \phi_0) \quad \text{periodico di periodo } T_0 = \frac{2\pi}{\omega_0}$$

$$y(t) = \sin(\omega_1 t + \phi_1) \quad \text{periodico di periodo } T_1 = \frac{2\pi}{\omega_1}$$

Consideriamo l'intervallo  $(0,10)$ , discretizzato come `t=linspace(0,10,1001)` ;

Per  $\omega_0 = 2\pi, \phi_0 = \frac{\pi}{2}, \omega_1 = \pi, \phi_1 = \frac{\pi}{3}$

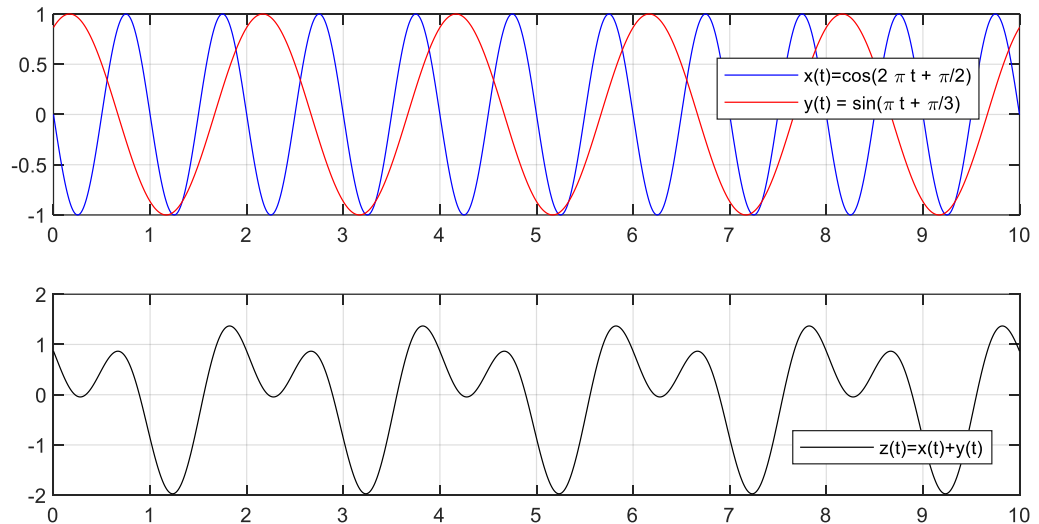
1) Tracciare in una stessa figura  
 $x(t)$  ed  $y(t)$

2) Verificare il periodo dei  
due segnali

3) Tracciare il segnale

$$z(t) = x(t) + y(t):$$

È periodico? Se sì, qual è  
il periodo?





# Esercizio 5: Segnale Esponenziale Complesso

Si consideri il segnale esponenziale complesso:

$$x(t) = Ae^{(a+j2\pi f)t}$$

con  $A = 100$ ,  $a = -1$ ,  $f = 1$

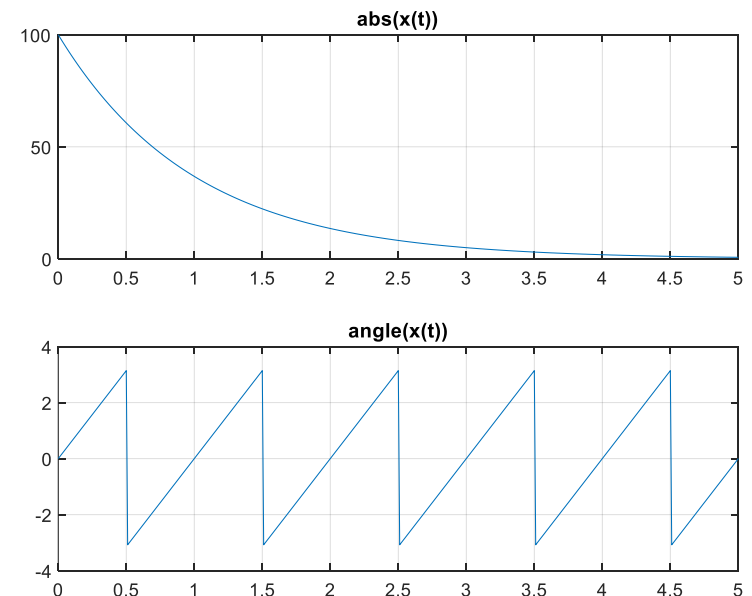
L'intervallo temporale è  $(0, 5)$  a passo  $\frac{1}{100}$ :  $t=0:1e-2:5$ ;

Nota: in Matlab si usa **1i** per l'unità immaginaria.

Tracciare in una figura (due subplot) modulo e fase di  $x(t)$

```
t=0:1e-2:5;  
A= 100; a = -1; f = 1;  
x = A*exp((a+1i*2*pi*f)*t);
```

```
figure(1); subplot(2,1,1);  
plot(t,abs(x))  
title('abs(x(t))'); grid  
subplot(2,1,2);  
plot(t,angle(x))  
title('angle(x(t))'); grid
```





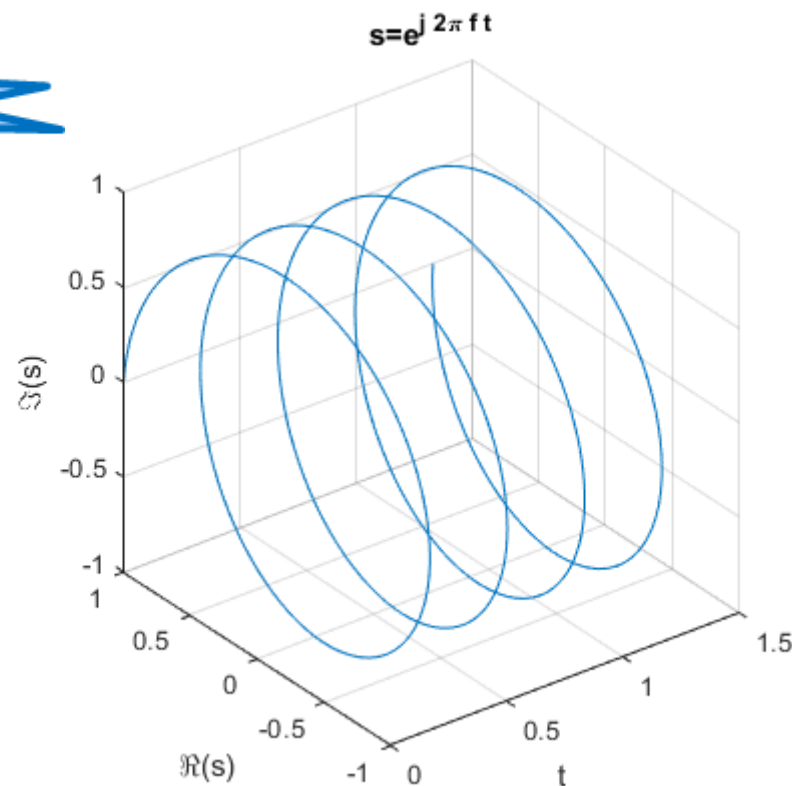
# Esercizio 6: Segnale Esponenziale Complesso

Si consideri il segnale  $s(t) = e^{j2\pi ft}$ . Eseguire il codice seguente (in uno script o nella command window) e capirne il funzionamento.

Cambiare i parametri e verificare il risultato



```
f = 3;  
T = 1/f;  
N = 500;  
NP = 4;  
t= linspace(0, NP*T, N); % NP periodi  
  
s = exp(2i*pi*f*t); % esponenziale complesso  
  
x = real(s);  
y = imag(s);  
figure;  
plot3(t,x,y)  
grid on; axis square;  
xlabel('t')  
ylabel('\Re(s)')  
zlabel('\Im(s)')  
title('s=e^{j 2\pi f t}')
```





# Esercizio 7: Calcolo numerico di convoluzioni a tempo continuo

Dati due segnali di variabile reale  $v$  e  $w$ , sotto ipotesi di convergenza la convoluzione tra essi è:

$$x = v * w$$

$$x(t) = \int_{-\infty}^{+\infty} v(\tau)w(t - \tau)d\tau$$

Per un  $t$  fissato, l'integrale di convoluzione può essere approssimato numericamente come:

$$x(t) \approx \sum_i v(\tau_i)w(t - \tau_i)\Delta\tau$$

dove i valori  $\tau_i$  sono spaziati di  $\Delta\tau$ :

$$\tau_{i+1} = \tau_i + \Delta\tau$$



# Esercizio 7: Calcolo numerico di convoluzioni a tempo continuo

## Esercizio:

Aprire il live script *convoluzione\_tc.mlx* fornito sul moodle.

Eseguirne le varie parti e comprendendo il codice

- La parte di visualizzazione include dei comandi avanzati che non è necessario approfondire

Rispondere alle domande scrivendo le risposte nel Live Script:

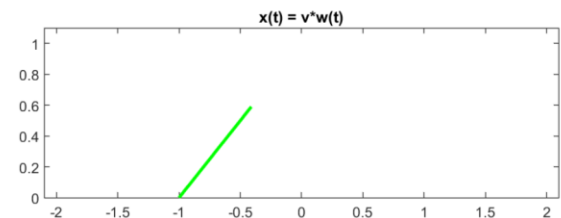
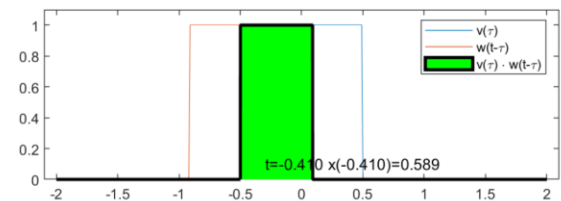
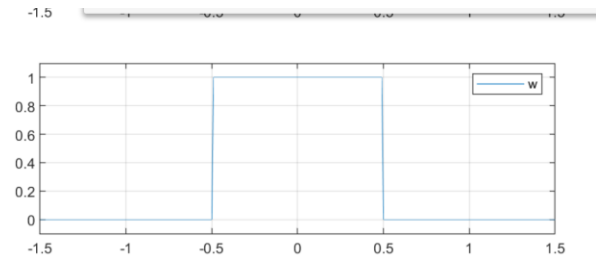
```
% Inizializzazione del risultato
x = zeros(size(tValues));
index = 0;

figure(2);
for t = tValues,
    index=index+1;
    % Calcoliamo l'integranda
    integranda = v(tau).*w(t-tau);
    % Calcoliamo l'approssimazione dell'integrale
    x(index) = sum(integranda) * DeltaTau;

    % Visualizzazione -->
    subplot(211);
    T = -2:-2:2; % Solo per visualizzazione
    plot(T, v(T), T, w(t-T)); hold on;
    h1=area(tau,integranda); axis([-2.1 2.1 0 1.1]);
    line([-2 -0.5],[0 0], 'LineWidth', 2, 'Color', 'k');
    line([0.5 2],[0 0], 'LineWidth', 2, 'Color', 'k');
    set(h1,'FaceColor', 'green', 'EdgeColor', 'black', 'LineWidth', 2);
    legend('v(\tau)', 'w(t-\tau)', 'v(\tau) \cdot w(t-\tau)');
    text(-0.3,0.1,sprintf('t=%4.3f x(%4.3f)=%4.3f',t,t, x(index)))
    hold off
    subplot(212); plot(tValues(1:index),x(1:index),'g','LineWidth',2);
    axis([-2.1 2.1 0 1.1]);
    title('x(t) = v*w(t)')
    pause(0.02);
    % Visualizzazione <--
end
```

### Domande:

- Tra le variabili  $v$ ,  $w$ , e  $x$ , quali sono funzioni e quali vettori di valori? Usare il comando `whos`
- Qual è il ruolo della variabile `index` nel codice Matlab fornito?
- Quale parte del codice Matlab calcola l'approssimazione dell'integrale?
- Esercizio da fare con carta e penna: calcolare la convoluzione tra  $v(t) = \text{rect}(t)$  e  $w(t) = \text{rect}(t)\cos(\pi t)$
- Decomentare e completare le righe 11 e 12 per verificare il risultato ottenuto analiticamente





# Esercizio 7: Calcolo numerico di convoluzioni a tempo continuo

La convoluzione tra l'impulso rettangolare e il semiperiodo di un coseno deve fornire un risultato come in figura:

