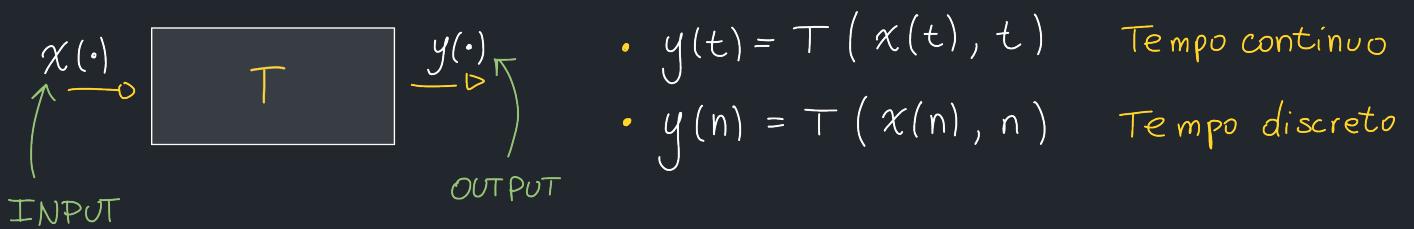


# Sistemi nel dominio del tempo

I sistemi nel dominio del tempo sono sistemi che possono essere descritti in base al tempo in cui i loro segnali o input cambiano. Questi sistemi possono essere analizzati e descritti utilizzando le equazioni del tempo che mostrano come i loro segnali di input si traducono in segnali di uscita.

## Sistemi

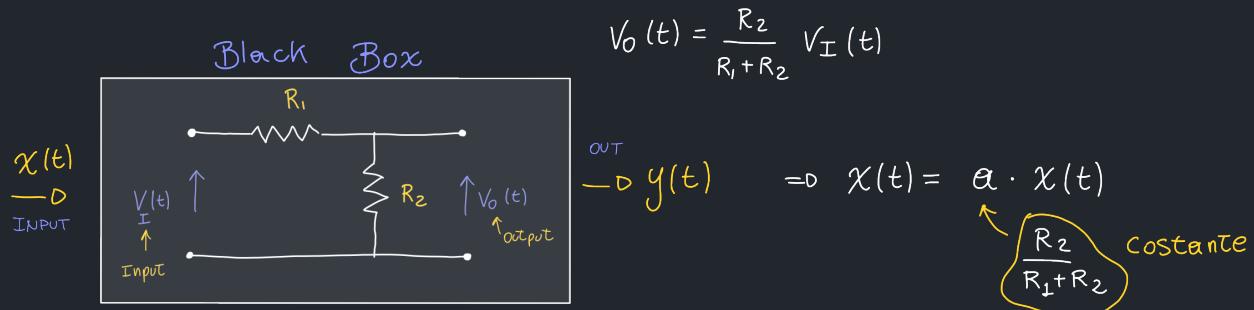


Sostanzialmente i sistemi sono delle **black box** che dato un input restituiscono un output.

## Esempi di sistemi

### Esempio di sistema continuo: Partitore Resistivo

ES: un Sistema continuo. PARTITORE RESISTIVO



⇒ Qualsiasi cosa che si comporta andando a modellare per una COSTANTE il segnale di ingresso, può essere rappresentato come una BLACK BOX e quindi come un SISTEMA.

## Esempio di sistema discreto: Ritardo elementare

ES: Un sistema Discreto : Ritardo Elementare

$$y(n) = x(n-1)$$

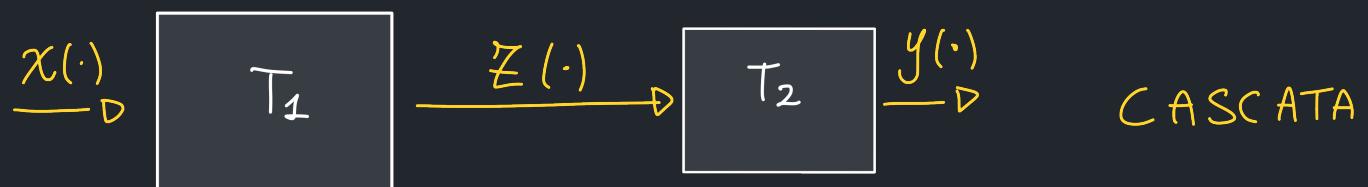
$\xrightarrow{\text{In}}$   $Z^{-1}$   $\xrightarrow{\text{Out}}$

## Disposizioni di sistemi

---

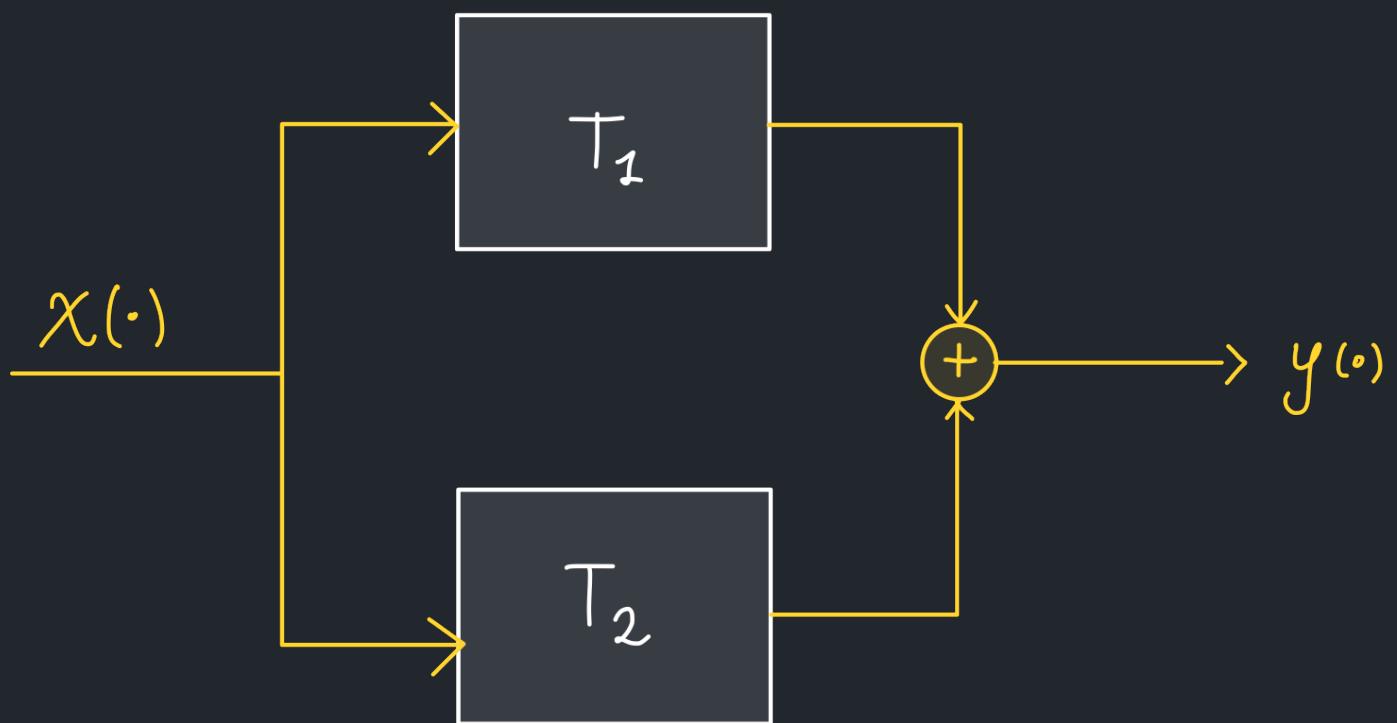
### Sistemi connessi in cascata

Connessione in cascata tra Sistemi



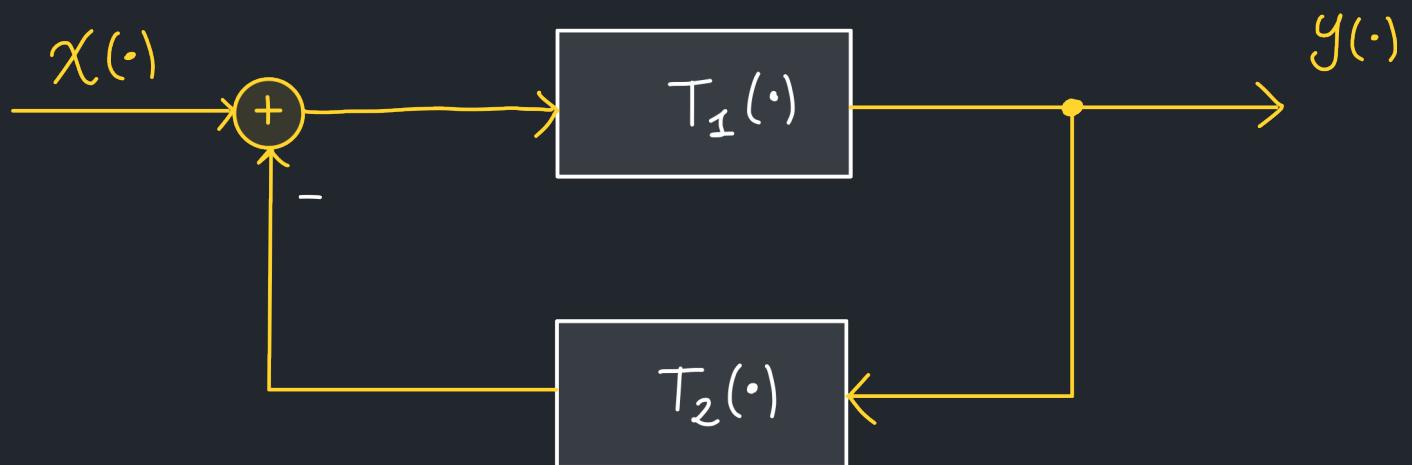
### Sistemi connessi in parallelo

## Connessione in Parallello tra Sistemi



## Sistemi connessi in retroazione

### Connessione in RETROAZIONE tra Sistemi



## Esempio di sistemi connessi in parallelo: filtro MA - Moving Average

Il filtro MA non fa altro che prendere gli ultimi  $n$  input e calcolarne la media; il valore risultante verrà poi trasmesso in output.

L'utilità di questo filtro risiede nel fatto che **smussa** in modo direttamente proporzionale al valore di  $N$ ; questo vuol dire che **di più elementi è calcolata la media ad ogni iterazione, più smussato sarà l'output risultante.**

E Sem pio MA - FILTER



- 2 Punti       $y[n] = \frac{x[n] + x[n-1]}{2}$
- 3 Punti       $y[n] = \frac{x[n] + x[n-1] + x[n-2]}{3}$
- $N$  punti       $y[n] = \frac{1}{N} \sum_{m=0}^{N-1} x[n-m]$

### Codice matlab per il filtro MA

```

function filtered_signal = getFilteredSignal(originalSignal, magnitude)
    filtered_signal = zeros(1, length(originalSignal));
    for n = 1:length(originalSignal)
        j = n;
        times = 0;
        for m = 0:magnitude-1
            times = times + originalSignal(j);
            j = j + 1;
        end
        filtered_signal(n) = times / magnitude;
    end
end
  
```

```

sum = 0;

while (times < magnitude && j > 1)
    sum = sum + originalSignal(j);
    j = j-1;
    times = times + 1;
end

filtered_signal(n) = sum/magnitude;
end
end

```

Questa funzione riceve in ingresso due parametri:

- Segnale da filtrare
- il numero di elementi di cui fare la media

Il codice si serve di due loop:

- Loop foor

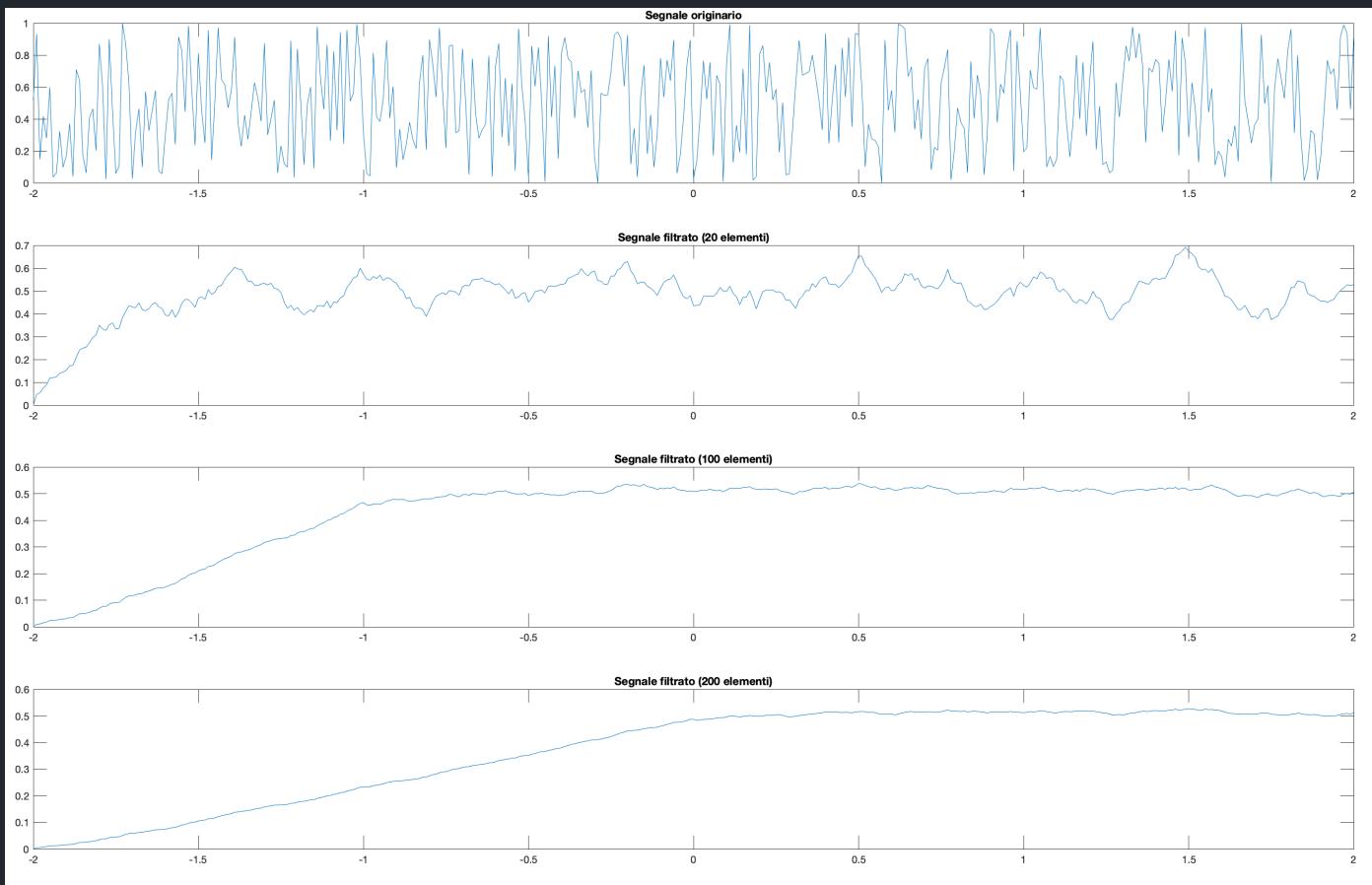
Questo loop serve ad iterare su tutti gli elementi dell'array, in modo da visitare tutti gli elementi del segnale e calcolarne la media mobile.

Alla fine di ogni ciclo viene calcolata la media degli ultimi m elementi in input; la media viene posta alla posizione corrente di un nuovo array (segnale filtrato).

- Loop While

Questo loop serve a calcolare la media degli m elementi (magnitude)

Alla fine delle varie operazioni la funzione restituisce un nuovo segnale filtrato:



Nella parte iniziale c'è una "rampa" proprio perchè la media deve "accumularsi" fino a cui magnitude = n elementi arrivati in input.

Se ad esempio decidiamo di fare la media di 50 elementi, e ne sono arrivati 20 è ovvio che la media non sarà completa.

## Proprietà dei sistemi

### Proprietà 1 dei sistemi: Sistemi Dispersivi / Sistemi con memoria

Un sistema è dispersivo o **con memoria** nel momento in cui l'**uscita in un istante di tempo dipende anche da valori precedenti del tempo**:

ES: Il partitore è NON DISPERSIVO

ES: L'elemento di ritardo  $Z^k$  è DISPERSIVO.

ES: Il filtro MA è DISPERSIVO

Il filtro MA visto prima ne è un esempio lampante: il segnale in output dipende proprio dalla media degli  $m$  valori precedenti all'istante di tempo corrente.

## Proprietà 2 dei sistemi: Sistemi Causali

La definizione di sistema causale è, se vogliamo, molto simile a quella dei sistemi non dispersivi; infatti un sistema causale è un sistema il cui segnale in uscita dipende solo dai valori del segnale in input **dello stesso istante di tempo o al massimo a valori del segnale per istanti di tempo precedenti a quello corrente.**

--> In un sistema causale l'output non dipende da valori futuri dell'input, rispetto all'istante di tempo corrente.

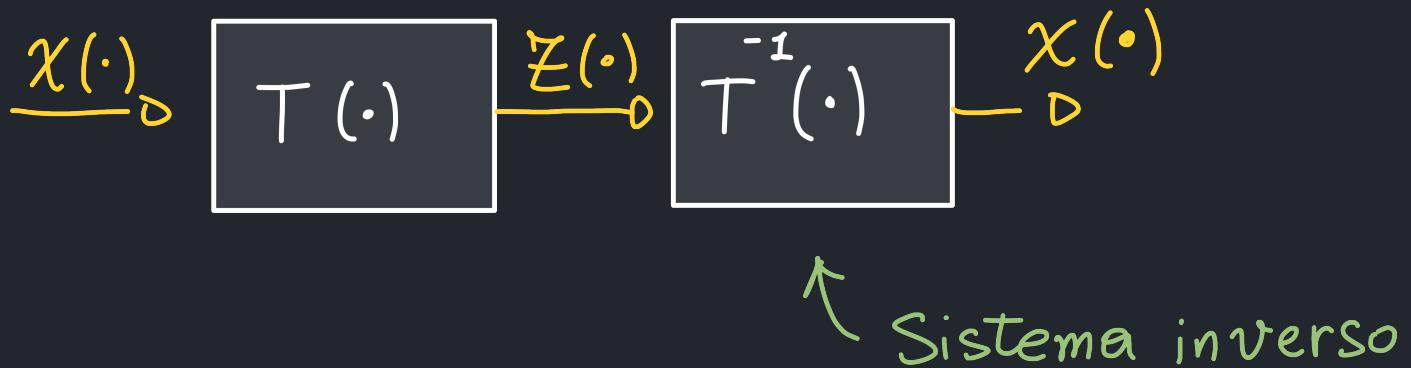
Dipende da  $x(t) / x(n)$  ed eventualmente versioni ritardate con  $T > 0 : x(t-T)$

ES: Sistema Non Causale

$$y(n) = b_0 x(n) + b_1 x(n+1) \quad \text{con} \quad b_1 \neq 0$$

## Proprietà 3 dei sistemi: Sistemi invertibili

Un sistema è invertibile se esiste un sistema, detto sistema inverso, che in uscita restituisca l'ingresso del primo sistema.



### Esempio di sistema non invertibile: elevazione al quadrato

ES: Sistema NON Invertibile

$$y(t) = x^2(t) \rightarrow \text{Matematicamente si: } \sqrt{x^2(t)} = \begin{cases} + \\ - \end{cases} x(t)$$

Ma poi sceglieremo il + o - ?

### Esempio di sistema invertibile: Accumulatore ed invertitore

L'accumulatore è un sistema che somma il valore corrente con quelli precedenti; quindi l'elemento n varrà la somma di tutti gli input fino all'input n;

Deduciamo (dopo qualche ragionamento) che il valore di input all'istante n, è uguale alla somma di tutti i valori fino all'istante n, **meno la somma di tutti i valori fino all'istante n-1**, ovvero:

ES: Accumulatore :  $y(n) = \sum_{k=-\infty}^n x(k)$  Somma da  $-\infty$  al valore corrente

→ L'invertitore e':  $z(n) = y(n) - y(n-1) = \sum_{k=-\infty}^n x(k) - \sum_{k=-\infty}^{n-1} x(k) = x(n)$

Possiamo convincerci di questa cosa andando a riprodurre il tutto su matlab:

```
% funzione che funge da sistema accumulatore
function accumulated_signal = getAccumulatedSignal(starting_signal)
    accumulated_signal = zeros(1, length(starting_signal));
```

```

for n = 1:length(starting_signal)
    acc_value = 0;
    for j = 1:n
        acc_value = acc_value + starting_signal(j);
    end
    accumulated_signal(n) = acc_value;
end

% funzione che funge da sistema inverso: invertitore
function original_signal = getOriginalSignal(accumulated_signal)
    original_signal = zeros(1, length(accumulated_signal));
    original_signal(1) = accumulated_signal(1);

    for n = 2:length(accumulated_signal)
        original_signal(n) = accumulated_signal(n) - accumulated_signal(n-1); % where the magic
happens
    end
end

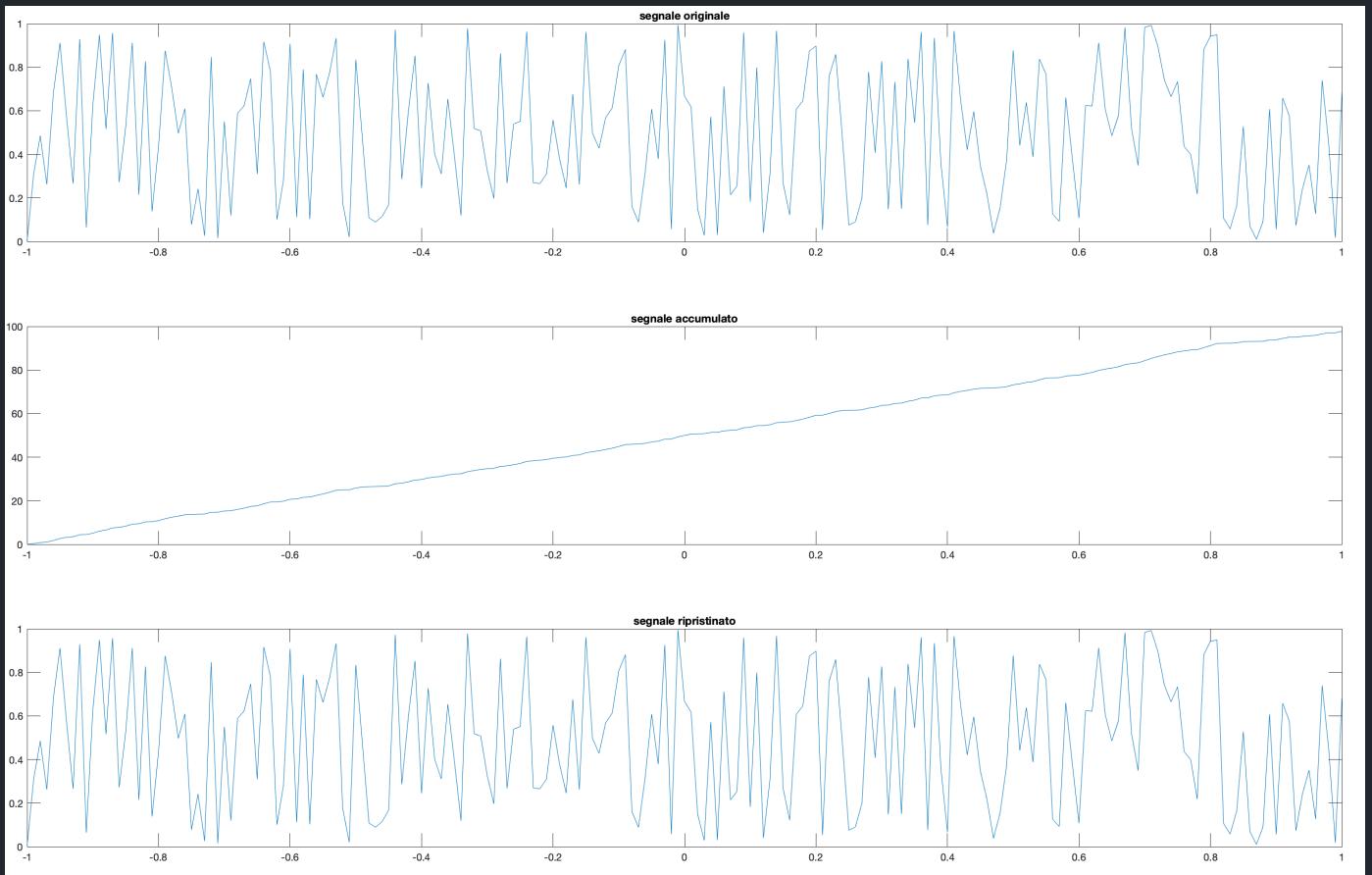
```

Le due funzioni sono molto semplici:

1. La prima funzione (di accumulazione) somma tutti gli elementi fino all'elemento corrente e posiziona la somma nella posizione equivalente alla corrente di un nuovo array.
2. La seconda funzione (di inversione) sottrae all'elemento corrente la somma degli elementi dell'elemento precedente:

```
original_signal(n) = accumulated_signal(n) -
accumulated_signal(n-1); .
```

Otteniamo quindi il seguente output:



## Proprietà 4 dei sistemi: Invarianza temporale

Un sistema è **invariante temporalmente** se ad una traslazione dell'ingresso corrisponde la stessa traslazione nel tempo dell'uscita.

Questo vuol dire che per controllare di trovarci con un sistema temporalmente invariante ci basta fornire al sistema un segnale ritardato di  $T_0$  ed osservare il segnale in output: se il segnale in output è ritardato di  $T_0$  allora il sistema è temporalmente invariante:

$$x(t) \xrightarrow{T(\cdot)} y(t) \Rightarrow x(t-T) \xrightarrow{\quad} y(t-T)$$

In poche parole questo ci dice che il sistema non varia a variare del tempo.

- Gli esempi precedenti erano tutti temporalmente invarianti.

ES:  $y(n) = n \cdot x(n)$

$\underbrace{T(x(n), n)}$   
La trasformazione  
dipende dal tempo (oppure  $n$ )

$$\Rightarrow x(n-N) \xrightarrow{\quad} n \cdot x(n-N) \neq y(n-N)$$

## Proprietà 5 dei sistemi: Stabilità

Un sistema è stabile se la sua uscita (risposta) non cresce all'infinito a seguito di un'entrata (stimolo) limitato. In altre parole, se la risposta a un segnale limitato non diventa eccessivamente grande o piccola nel tempo, il sistema è stabile.

$$|x(\cdot)| \leq M < \infty \Rightarrow |y(\cdot)| \leq M \leq \infty$$

Queste proprietà devono valere per un **qualsiasi segnale**, in questo caso limitato; quindi per dimostrare che un sistema non è stabile il contrario ci basta trovare un controesempio per cui questa proprietà non vale:

Per convincerci che un sistema sia o meno stabile, ci basta effettuare il limite per il tempo che tende all'infinito dell'output considerando un input limitato; in questo caso sceglieremo come input il gradino unitario che sappiamo avere valori limitati anche per valori *infiniti* di tempo:

ES:  $y(n) = n x(n)$

$$x(n) = u(n) \leq 1$$

$$\Rightarrow y(n) = n \cdot u(n) \quad \text{Non e' limitato, perche' per } n \rightarrow \infty : \lim_{n \rightarrow \infty} n \cdot u(n) = +\infty$$

$\downarrow_{+\infty}$

## Proprietà 6 dei sistemi: Linearità

Un sistema è lineare quando la sua risposta ad una combinazione di segnali in ingresso è uguale alla combinazione delle sue risposte ai singoli segnali di ingresso.

Dal punto di vista "fiscale" un sistema è detto lineare nel momento in cui esso è sia **omogeneo** che **additivo**:

### Proprietà 6.1 dei sistemi: Omogeneità

Un sistema è omogeneo nel momento in cui applichiamo un cambiamento di scala in input e questo si ripresenta anche in output:

- **OMOGENEITÀ**:  $\chi(\cdot) \xrightarrow{T} y(\cdot)$  quando applichiamo un cambiamento di scala:  
 $a \cdot \chi(\cdot) \xrightarrow{} a \cdot y(\cdot)$  Il cambiamento di scala  
persiste

### Proprietà 6.2 dei sistemi: Additività

Un sistema è additivo quando in input viene fornito un segnale composto dalla somma di due (o più segnali) ed in output il sistema restituirà la somma degli output:

- **ADDITIVITÀ**:  $\chi_1(\cdot) \xrightarrow{} y_1(\cdot)$   
 $\chi_2(\cdot) \xrightarrow{} y_2(\cdot)$   $\xrightarrow{} \underbrace{\chi_1(\cdot) + \chi_2(\cdot)}_{\text{Entrata}} \xrightarrow{} \underbrace{y_1(\cdot) + y_2(\cdot)}_{\text{Uscita}}$

---

Capiamo bene che un sistema è lineare nel momento in cui c'è una **combinazione di queste due proprietà**:

$\Rightarrow$  Se il sistema è lineare:  $a_1 \chi_1(\cdot) + a_2 \chi_2(\cdot) \xrightarrow{} a_1 y_1(\cdot) + a_2 y_2(\cdot)$

Ci aspettiamo che la maggior parte dei sistemi siano lineari; non a caso nel corso ci focalizziamo proprio su questo tipo di sistemi.

### Esempio di sistema lineare

Abbiamo visto come un sistema lineare debba rispettare la proprietà:

Dato un input composto dalla somma di due (o più) segnali, l'output deve essere la somma degli output; se nel sistema c'è una moltiplicazione del segnale per una costante, questa deve ripercuotersi su ogni "sottosegnale".

Possiamo convincerci di ciò grazie a del buon codice matlab:

```
function y = linear_system(x, a)
% LINEAR_SYSTEM modifica un segnale di ingresso in modo lineare
% x: segnale di ingresso
% a: fattore di scala
% y: segnale di uscita

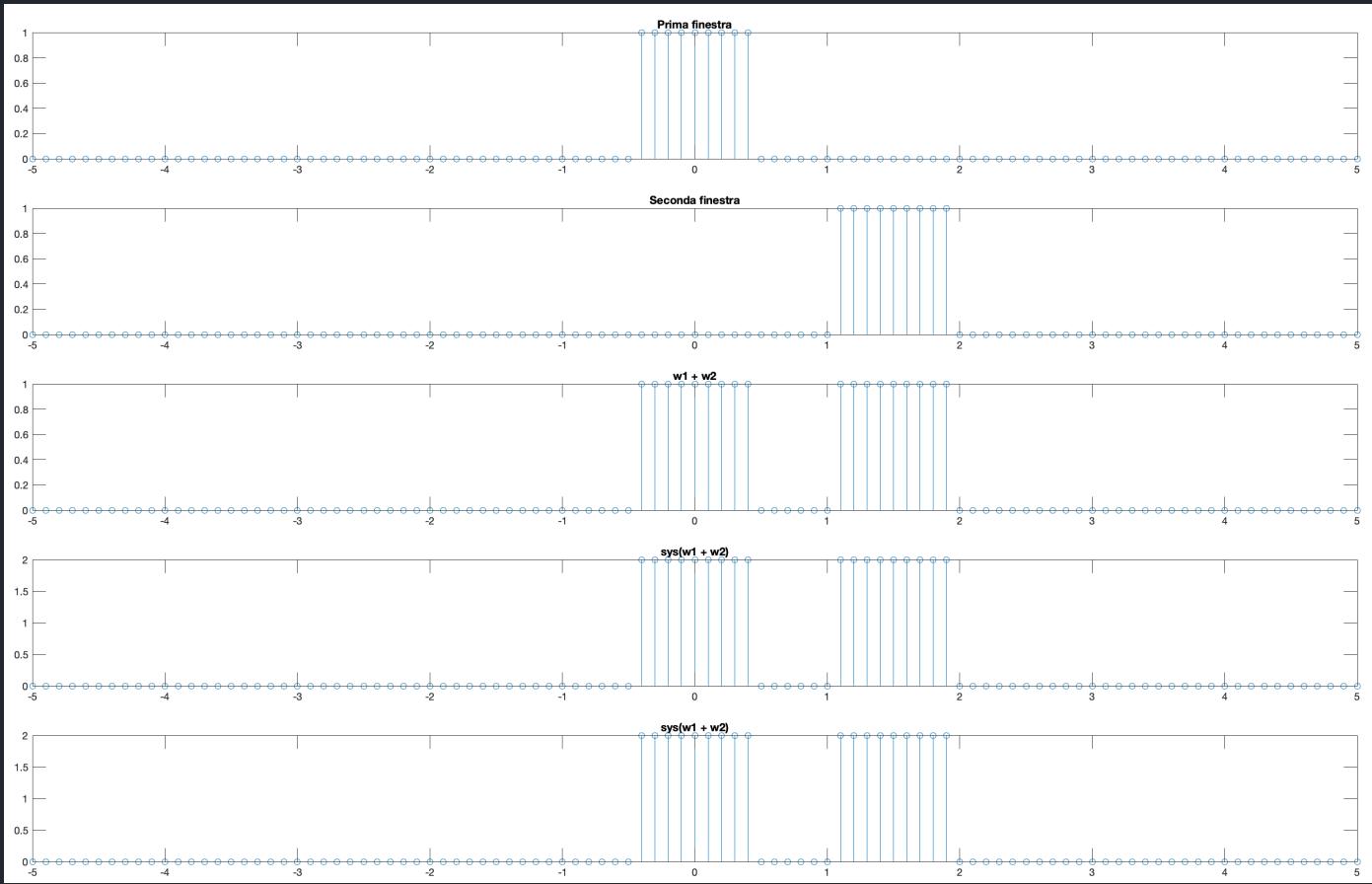
y = a .* x;
end
```

Questa funzione semplicemente moltiplica ogni elemento di un segnale in input per una costante; possiamo fornire alla funzione sia un segnale elementare che un segnale ottenuto dalla somma di due (o più) segnali elementari; infatti è proprio quello che faremo per dimostrare la linearità di questo sistema:

1. Forniamo al sistema un input composto **dalla somma di due segnali elementari** (due finestre diverse) in modo da ottenere  
 $y(t) = \text{sys}(w1 + w2)$
2. Osserviamo l'output del sistema.

1. Forniamo al sistema un input composto da un segnale elementare (una finestra);
2. Successivamente effettueremo la somma di due output del sistema, in modo da ottenere:  
 $y(t) = \text{sys}(w1) + \text{sys}(w2)$
3. Osserviamo l'output del sistema.

Noteremo che i due output sono identici!



## Esempio di sistema non lineare

Un esempio di sistema non lineare può essere l'elevazione a potenza:

ES: Sistema non lineare

$$y(n) = n x^2(n) \quad \text{Omogeneo? } x(n) = u(n) \rightarrow y(n) = \underbrace{(n)}_{\text{non c'era prima}} u^2(n)$$

Adoliti vuoi?

$$\begin{aligned} x_1(n) &= u_1(n) \\ x_2(n) &= u_2(n) \end{aligned} \Rightarrow y(n) = n \cdot (u_1(n) + u_2(n))^2 = \begin{cases} n u_1^2(n) + n u_2^2(n) + 2 n u_1(n) u_2(n) \\ \neq u_1(n) + u_2(n) \end{cases}$$

```

function y = nonlinear_system(x, power)
% NONLINEAR_SYSTEM modifica un segnale di ingresso in modo non lineare
% x: segnale di ingresso
% y: segnale di uscita

y = x.^power;
end

```

Questa funzione non fa altro che elevare ogni elemento di un segnale alla potenza specificata dall'argomento power; per dimostrare la non linearità del sistema andremo ad effettuare due plot:

1. L'output del sistema moltiplicato per 3.

**input -> sys -> y(n) \* 3**

2. Fornisco un input moltiplicato per 3 al sistema e poi lo lotto.

**input \* 3 -> sys -> y(n)**

