

Controllo dei sistemi energetici

Introduzione a Matlab-Simulink

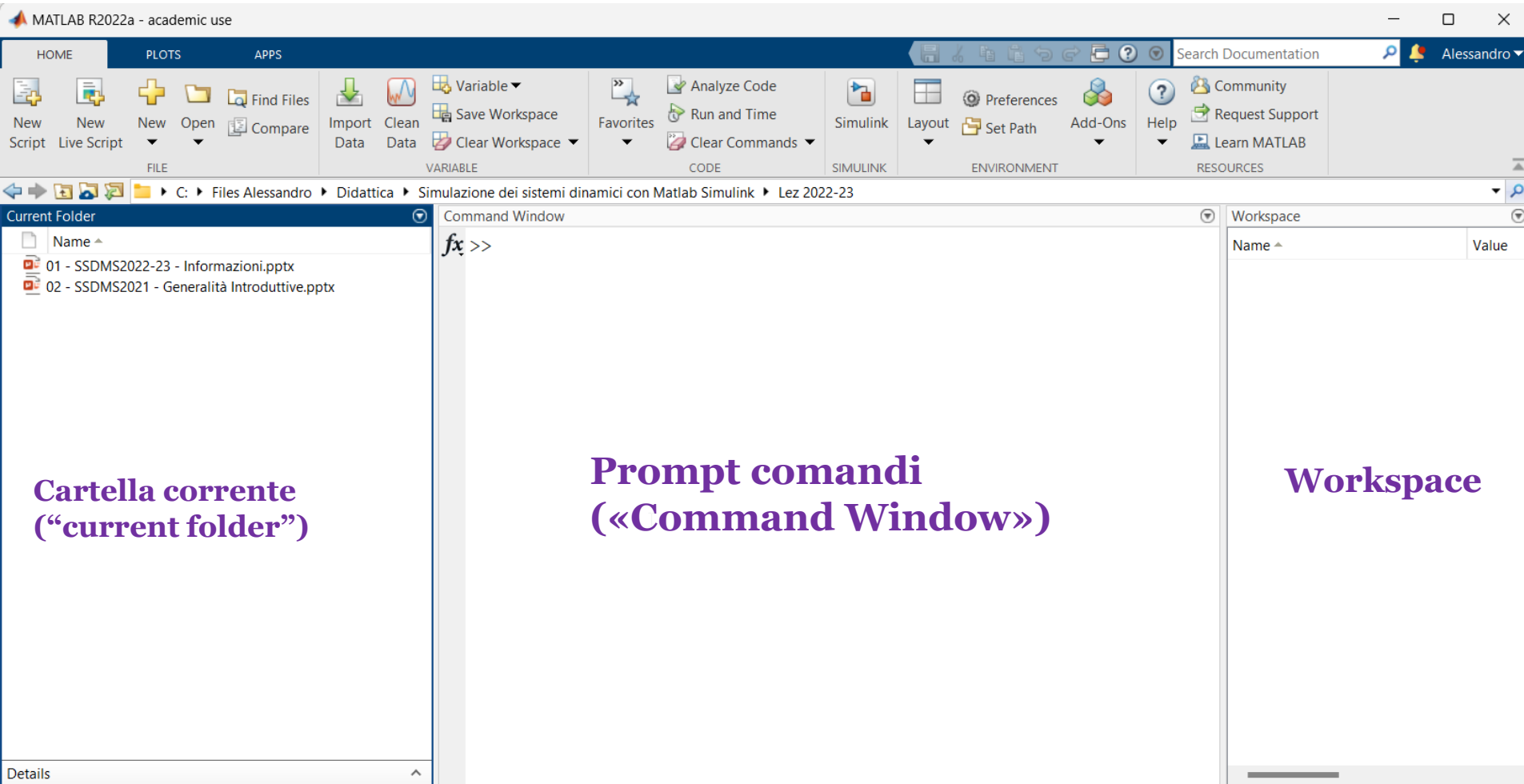
Prof. Alessandro Pisano
`apisano@unica.it`

Generalità

Matlab è un applicativo per il calcolo scientifico ampiamente utilizzato sia in ambito accademico che industriale.

Simulink, che ne costituisce una interfaccia grafica, è in particolare il software ad oggi maggiormente utilizzato per la simulazione dinamica in ambito industriale.

Finestra di avvio (v. R2022a)

HOME

**Cartella corrente
("current folder")**

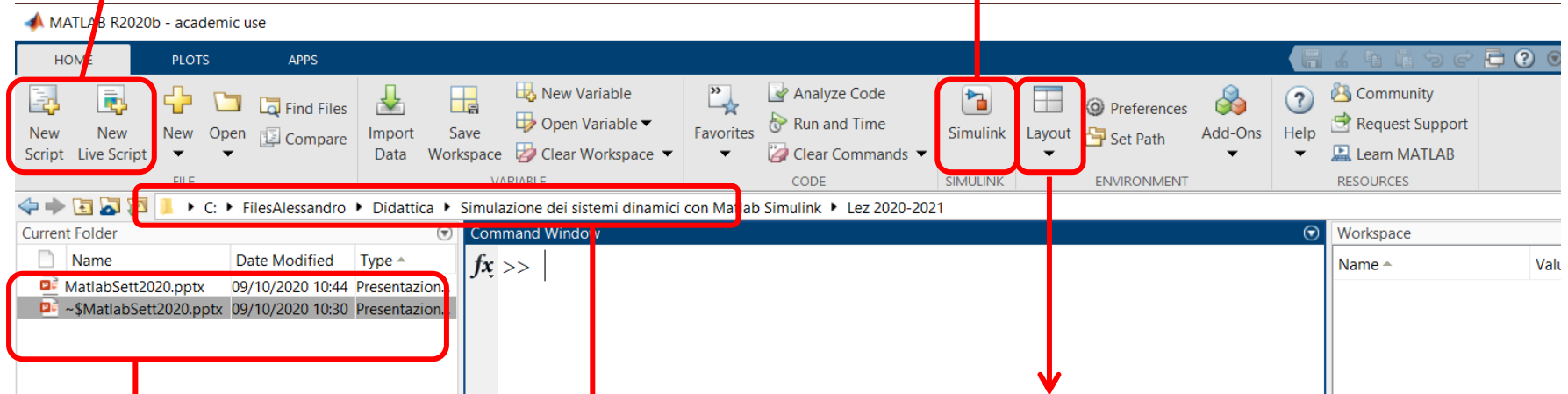
**Prompt comandi
("«Command Window»")**

Workspace

Finestra di avvio

Editor per «Script» e «Live Script»

Avvio SIMULINK



Contenuto della
cartella corrente

Cartella corrente
(«current folder»)

Layout della finestra
di avvio del programma

Preliminari

Command Window

New to MATLAB? Watch this [Video](#), see [Examples](#), or read [Getting Started](#).

```
>> (log(1.25)*3.45+sin(pi/2))/2
```

```
ans =
```

```
0.8849
```

```
fx >> |
```

Il prompt dei comandi (“command window”) può essere usato come una “calcolatrice”.

Operatori aritmetici base

+	-	*	/	^
---	---	---	---	---

```
>> c1=sqrt(2)^1.3
```

```
c1 =
```

```
1.5692
```

Il risultato delle operazioni può essere **memorizzato in variabili**.

Il contenuto del **workspace** viene aggiornato con la nuova variabile appena definita

Il **punto e virgola** alla fine di una istruzione disabilita la visualizzazione del risultato.

Workspace	
Name ▲	Value
ans	0.8849
c1	1.5692

Funzioni matematiche principali

<u>Trigonometric.</u>		<u>Exponential.</u>	
sin	Sine.	exp	Exponential.
sinh	Hyperbolic sine.	log	Natural logarithm.
asin	Inverse sine.	log10	Common logarithm.
asinh	Inverse hyperbolic sine.	sqrt	Square root.
cos	Cosine.		
cosh	Hyperbolic cosine.		
acos	Inverse cosine.	<u>Complex.</u>	
acosh	Inverse hyperbolic cosine.	abs	Absolute value.
tan	Tangent.	angle	Phase angle.
tanh	Hyperbolic tangent.	conj	Complex conjugate.
atan	Inverse tangent.	imag	Complex imaginary part.
atan2	Four quadrant inverse tangent.	real	Complex real part.
atanh	Inverse hyperbolic tangent.		
sec	Secant.	<u>Numeric.</u>	
sech	Hyperbolic secant.	fix	Round towards zero.
asec	Inverse secant.	floor	Round towards minus infinity.
asech	Inverse hyperbolic secant.	ceil	Round towards plus infinity.
csc	Cosecant.	round	Round towards nearest integer.
csch	Hyperbolic cosecant.	rem	Remainder after division.
acsc	Inverse cosecant.	sign	Signum function.
acsch	Inverse hyperbolic cosecant.		
cot	Cotangent.		
coth	Hyperbolic cotangent.		
acot	Inverse cotangent.		
acoth	Inverse hyperbolic cotangent.		

Informazioni ed esempi

<https://it.mathworks.com/help/matlab/elementary-math.html>

Matematica elementare

R2023a

Trigonometria, esponenziali e logaritmi, valori complessi, arrotondamenti, resti, matematica discreta

Le funzioni matematiche elementari includono le funzionalità per le operazioni aritmetiche (+, -, *, ...), costanti matematiche (Inf, pi, ...), operazioni polinomiali (poly, roots, ...) e funzioni matematiche speciali, come gamma e beta.

Categorie

Operazioni aritmetiche

Addizione, sottrazione, moltiplicazione, divisione, potenza, arrotondamento

Trigonometria

Seno, coseno e funzioni correlate, con risultati in radianti o gradi

Esponenti e logaritmi

Funzioni esponenziali, logaritmiche, di potenze e di radici

Numeri complessi

Componenti reali e immaginarie, angoli di fase

Matematica discreta

Fattori primi, fattoriali, permutazioni, frazioni razionali, minimo comune multiplo, massimo comune divisore

Polinomi

Adattamento della curva, radici, espansioni di frazioni parziali

Funzioni speciali

Funzioni di Bessel, Legendre, ellittiche, di errore, gamma e altre funzioni

Costanti e matrici di prova

Pi greco, numero non numero, infinito; matrici di Hadamard, compagna, di Pascal e altre matrici specializzate

Variabili e costanti speciali

- **ans**: variabile temporanea che contiene il risultato della operazione più recente;
- **eps**: il più piccolo numero reale che addizionato ad 1 crea un numero maggiore di 1 ($\cong 2.2E-16$);
- **i, j**: unità immaginaria;
- **Inf**: infinito;
- **NaN**: risultato numerico indefinito; (ad es. 0/0)
- **pi**: indica il numero π .

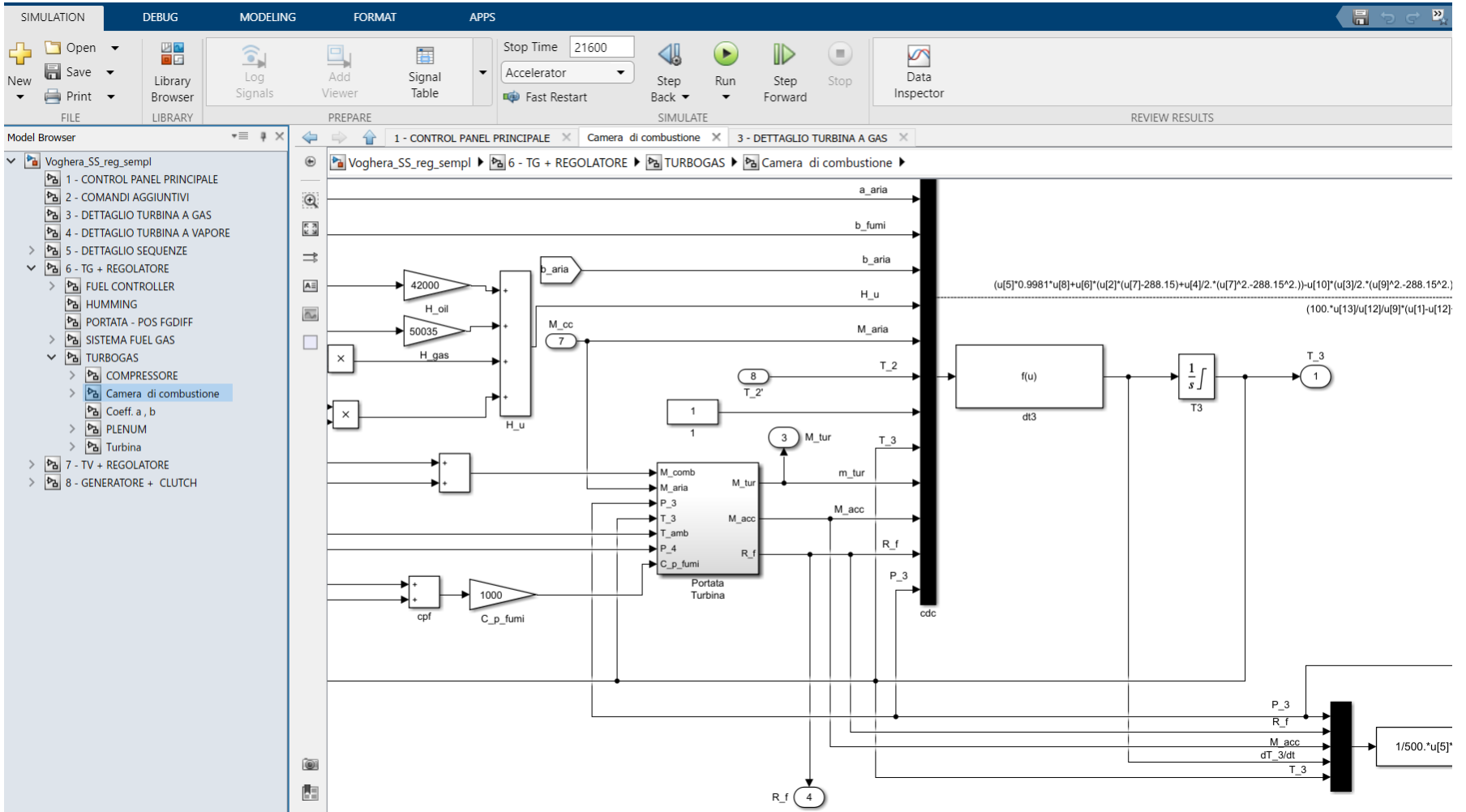
I nomi delle variabili **devono iniziare con un carattere alfabetico**, e successivamente si possono avere caratteri alfanumerici o underscores. La lunghezza del nome di una variabile non deve eccedere **63** caratteri.

I nomi delle variabili **sono case-sensitive**.

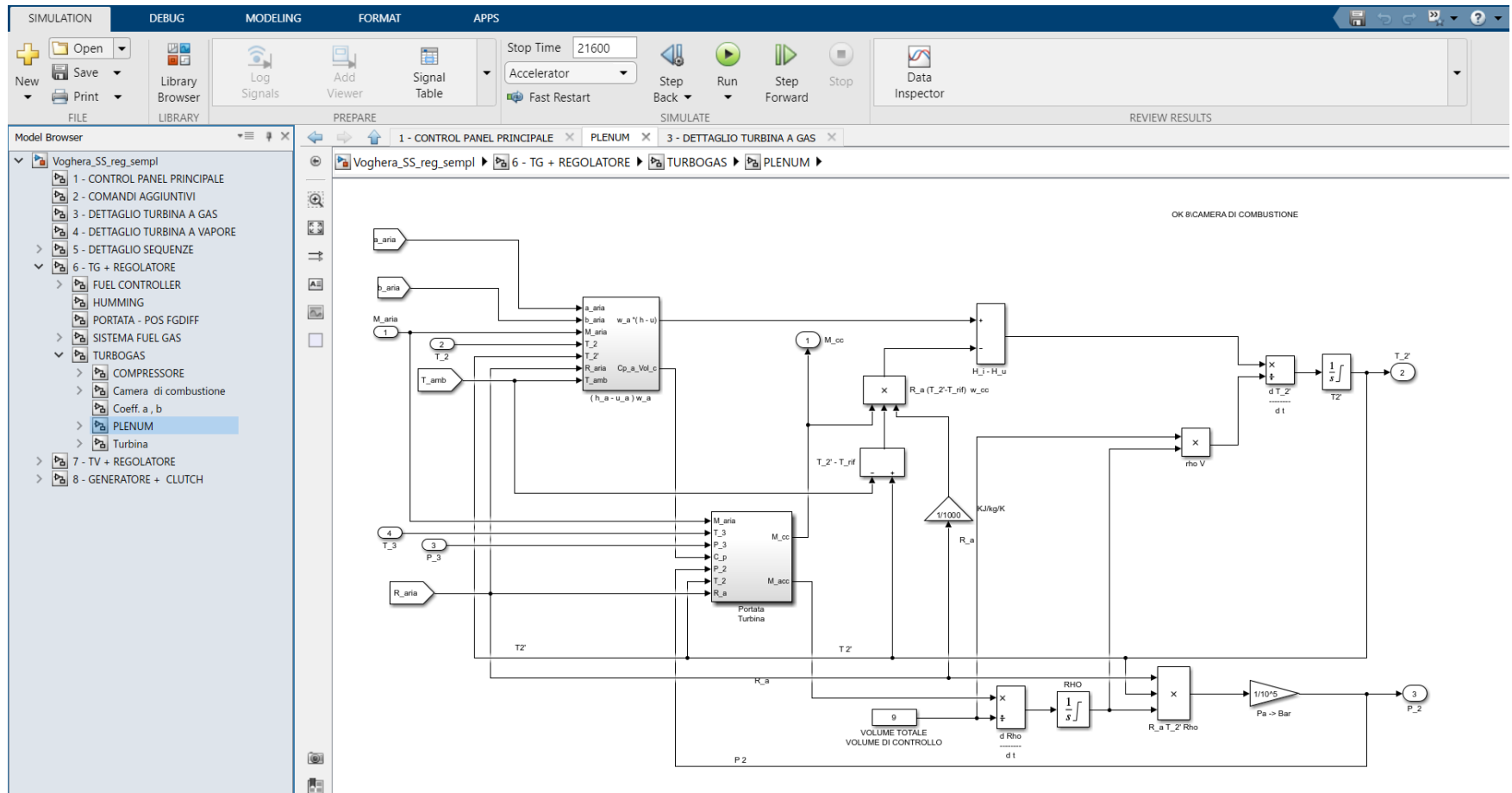
Se si attribuisce ad una variabile il nome di una costante speciale (ad es. $\pi=4$) la costante speciale non può essere più utilizzata finchè la variabile avete il suo stesso nome non venga rimossa dal workspace.

Simulink consente la realizzazione di modelli per la simulazione dinamica per via **grafica**

Voghera_SS_reg_sempl/.../TURBOGAS/Camera di combustione - Simulink academic use



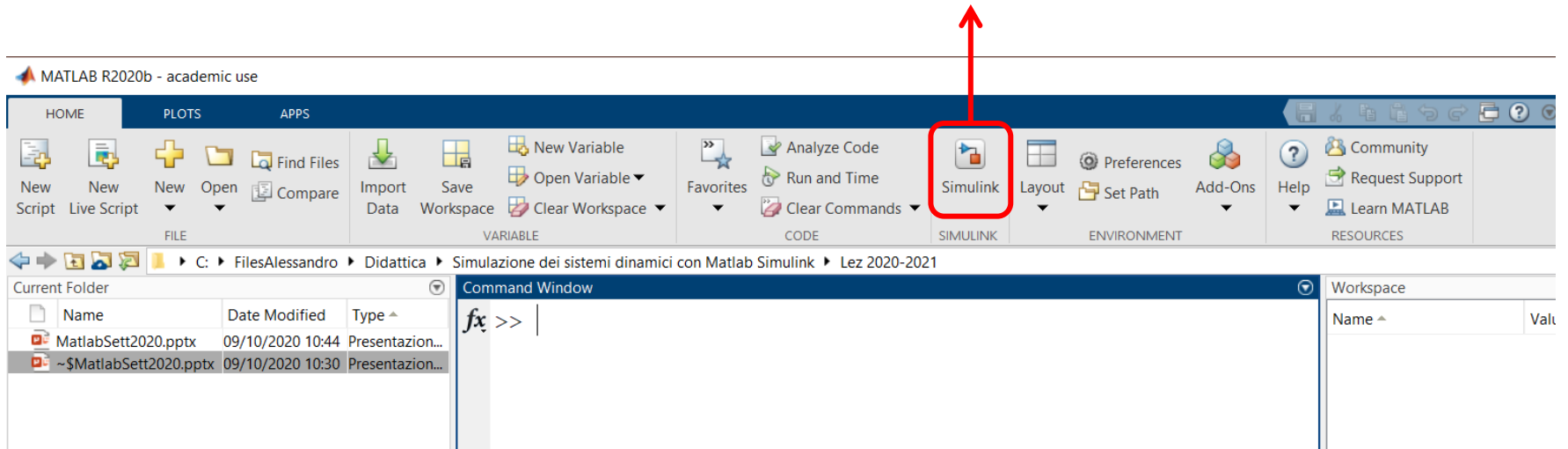
Simulink consente la realizzazione di modelli per la simulazione dinamica per via **grafica**



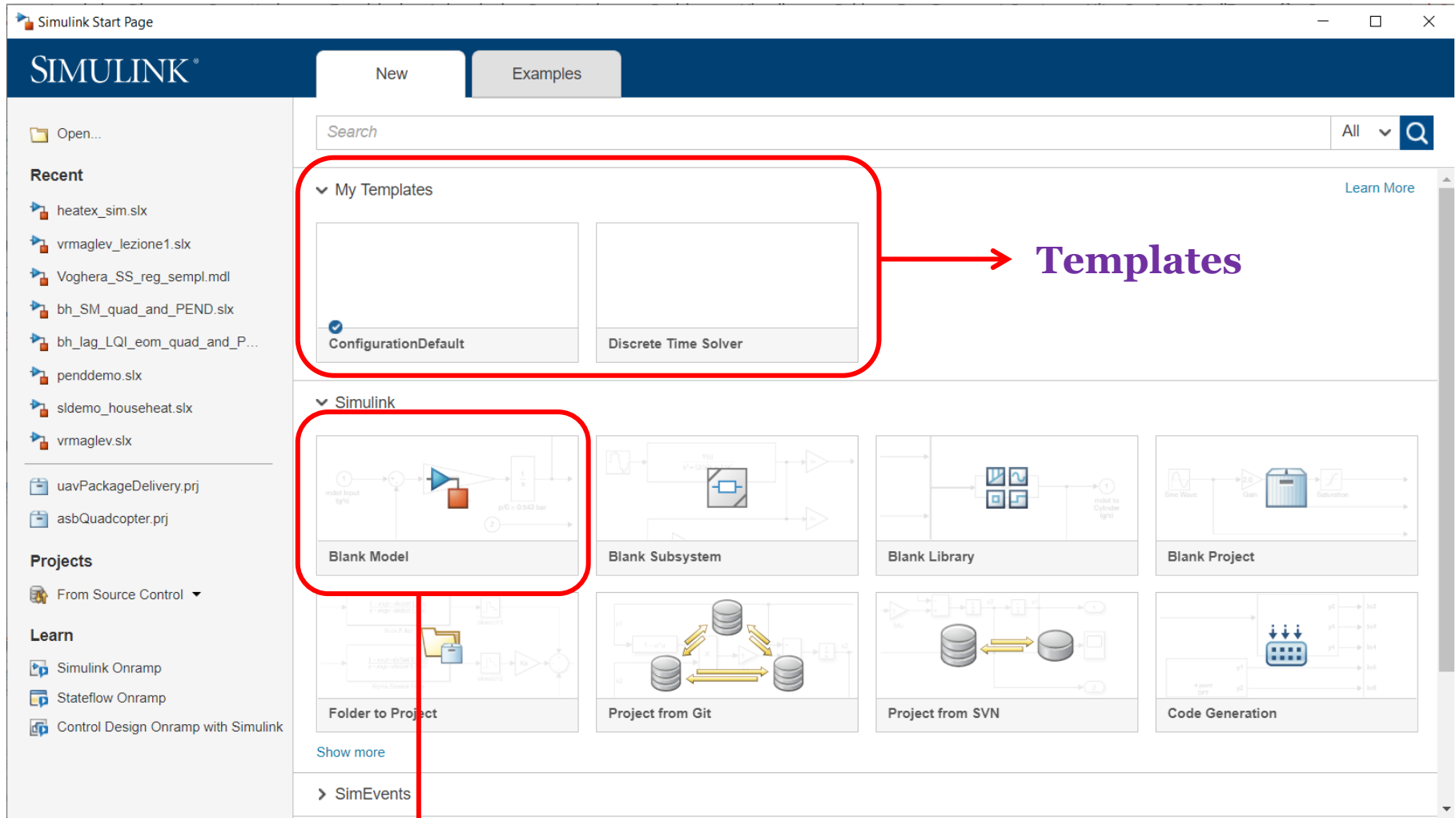
Ciascuna linea di connessione corrisponde ad un **segnale**. I vari blocchi applicano ai segnali in ingresso determinate operazioni matematiche, e producono in uscita segnali risultanti da tali elaborazioni.

Finestra di avvio di Matlab (R2020b)

Avvio SIMULINK



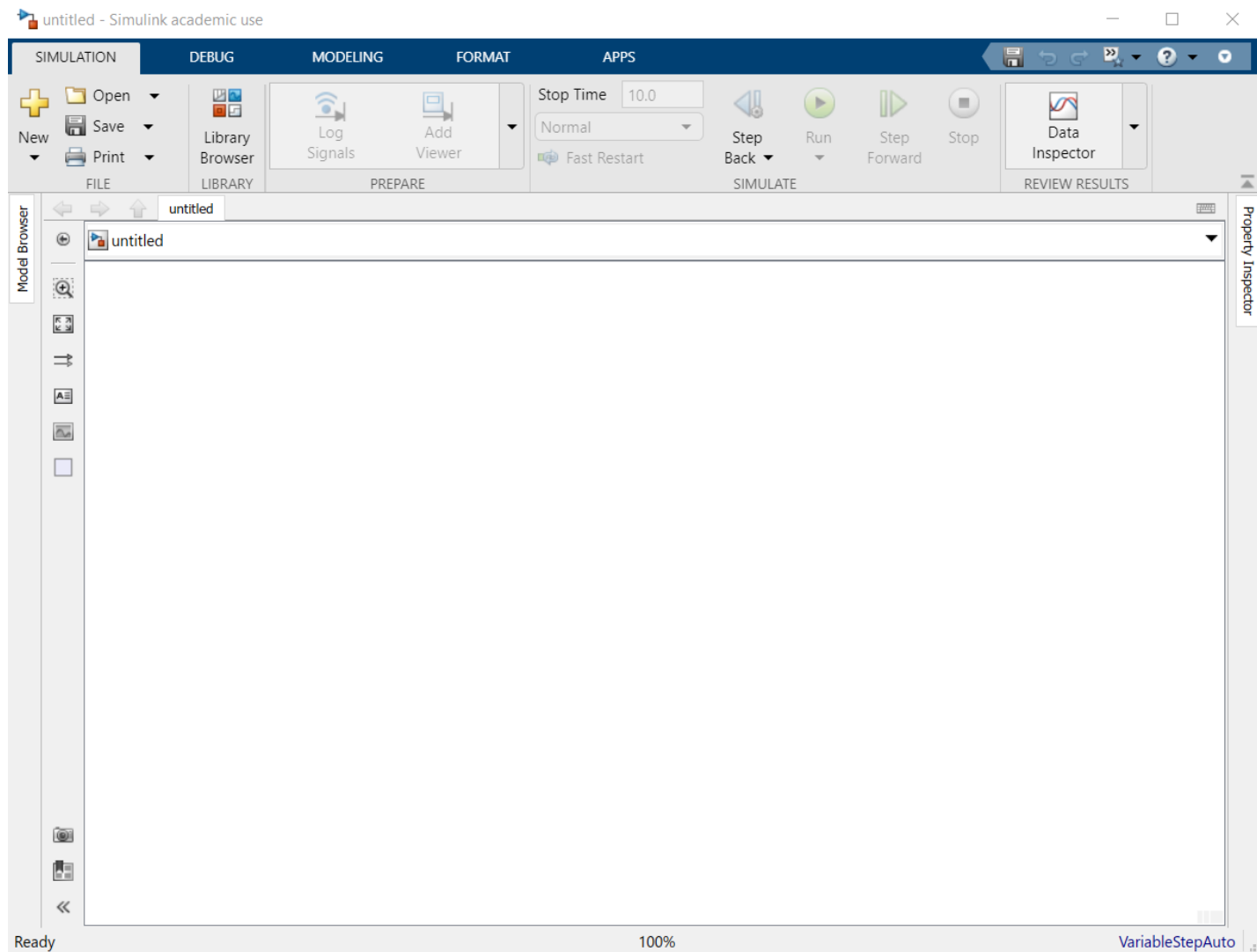
Finestra di avvio SIMULINK



Templates

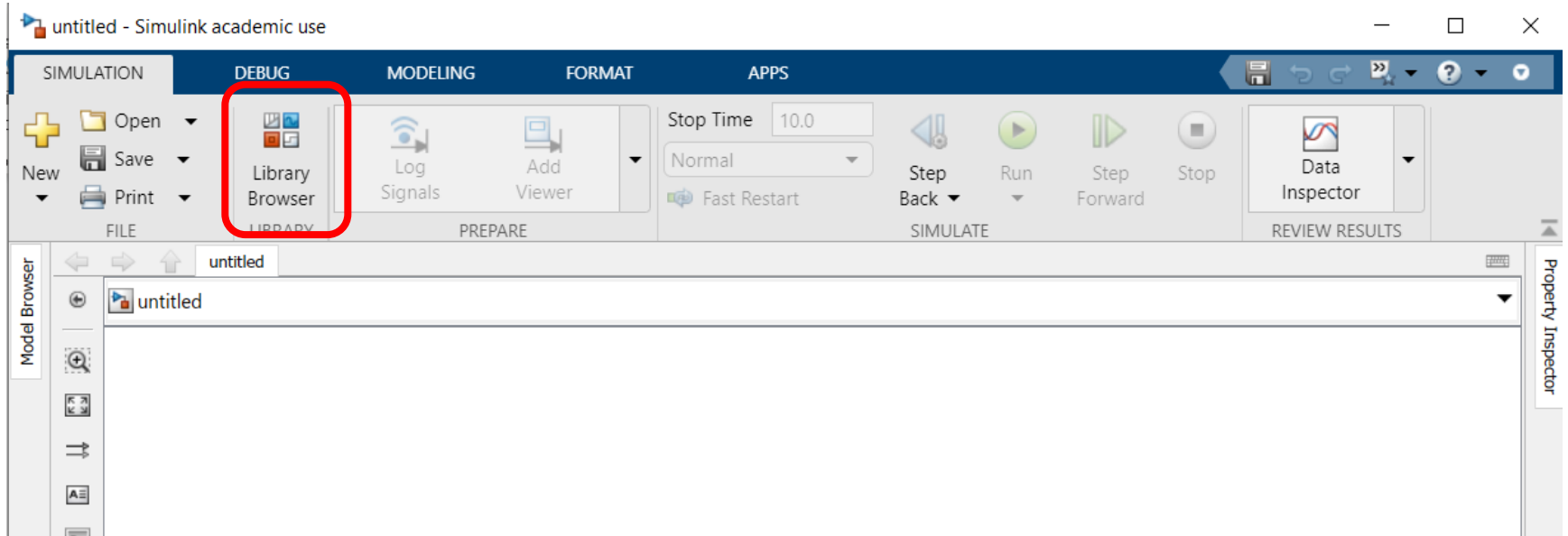
**Apri un modello Simulink vuoto
avente le proprietà di default**

Modello Simulink in bianco

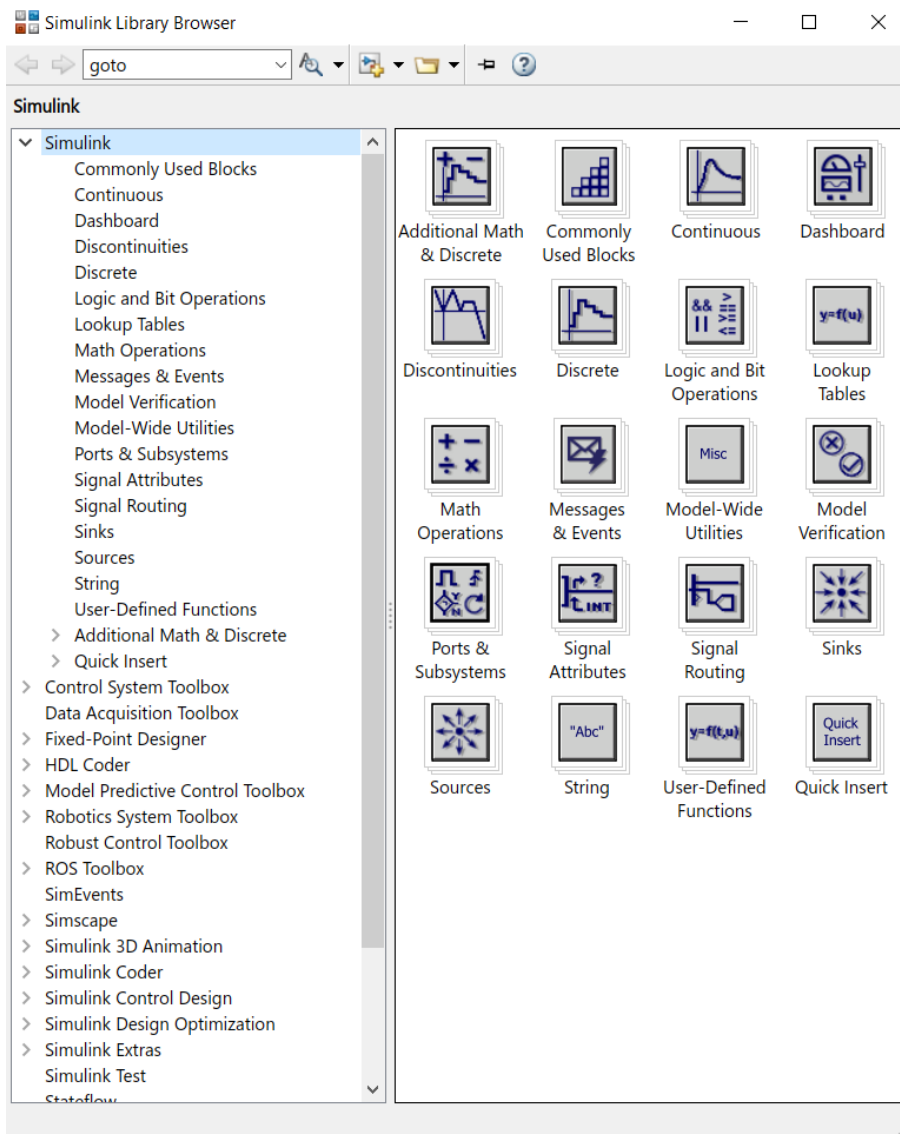


La realizzazione di modelli di simulazione dinamica avviene per via grafica **assemblando fra loro all'interno della pagina di lavoro un certo numero di blocchi Simulink**, in modo da implementare le funzionalità desiderate.

I blocchi Simulink sono allocati all'interno di librerie. E' possibile accedere al «Library browser» cliccando il relativo pulsante nella finestra che ospita il modello in bianco (Menu: «SIMULATION»)



Simulink Library Browser

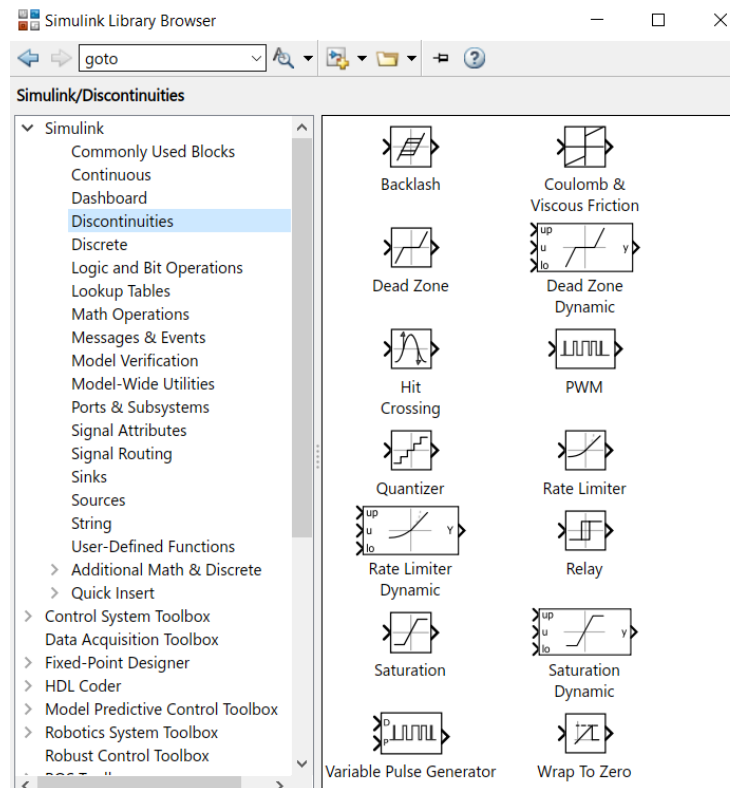


Contiene le librerie di blocchi elementari SIMULINK, da quelli di base fino a quelli più sofisticati orientati a particolari applicazioni

Indaghiamo rapidamente il contenuto di alcune librerie di base che contengono i blocchi Simulink maggiormente impiegati.

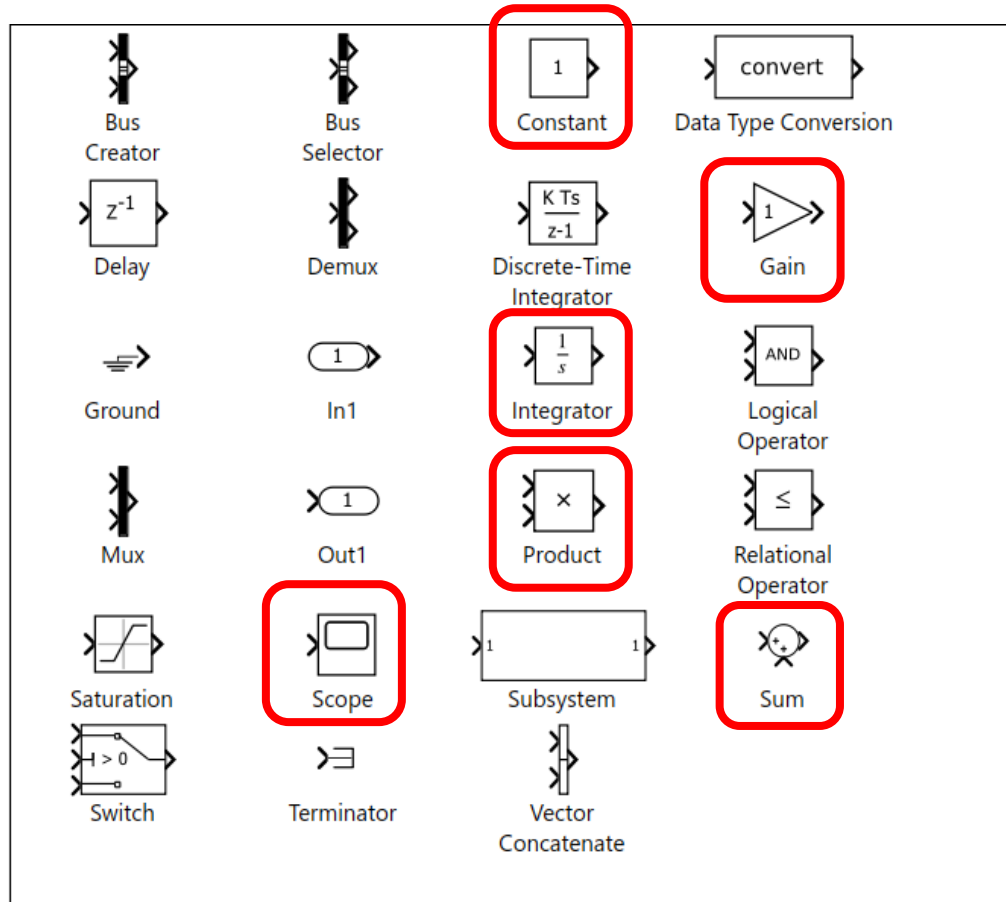
Per accedere al contenuto di una libreria si faccia click sul nome della stessa nella parte sinistra della finestra **Simulink Library Browser**

Contestualmente sarà visualizzato il contenuto della libreria nella parte destra della medesima finestra.



La prima libreria della lista è la Libreria “**Commonly used blocks**”

COMMONLY USED BLOCKS



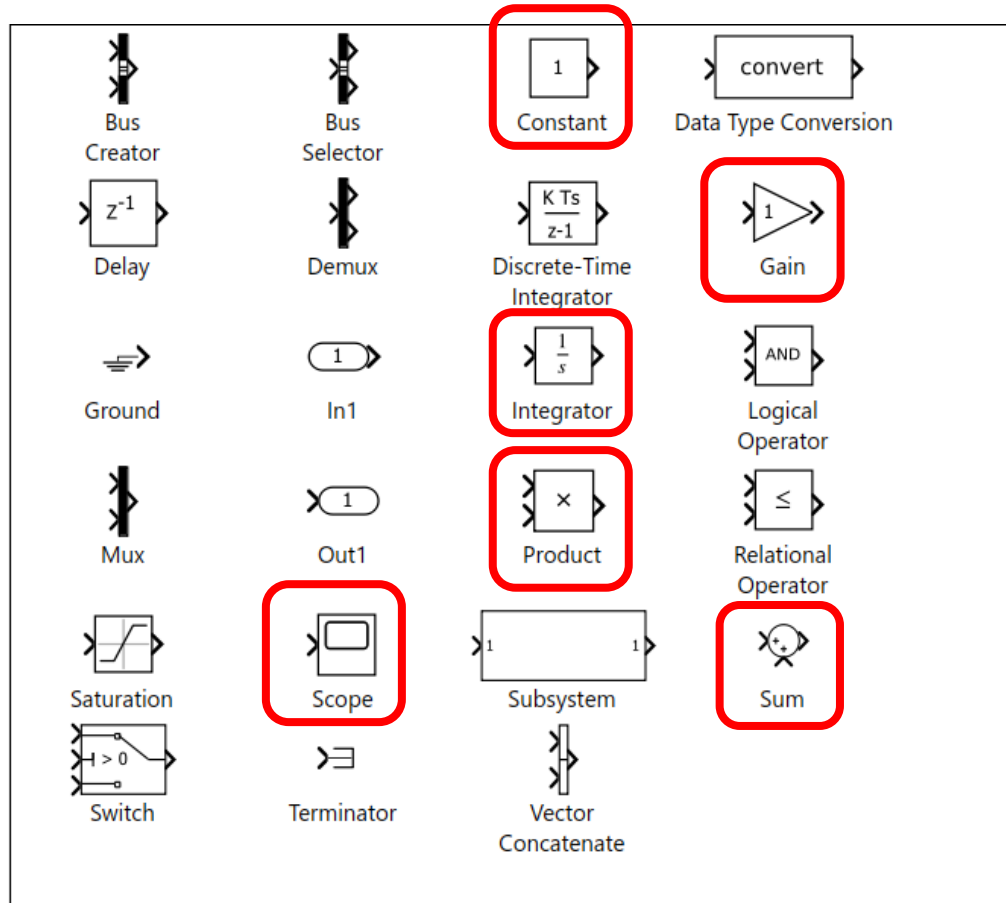
E' una libreria che contiene un insieme di blocchi che implementano funzionalità variegate e che vengono impiegati particolarmente di frequente nei modelli di simulazione.

Nello screenshot a sinistra, che ne mostra il contenuto, sono evidenziati alcuni blocchi di particolare importanza.

Il blocco **Constant** genera al suo terminale di uscita un segnale costante nel tempo.

Il blocco **Gain** moltiplica il segnale connesso al terminale di ingresso per un guadagno costante, e restituisce il risultato nel terminale di uscita.

COMMONLY USED BLOCKS



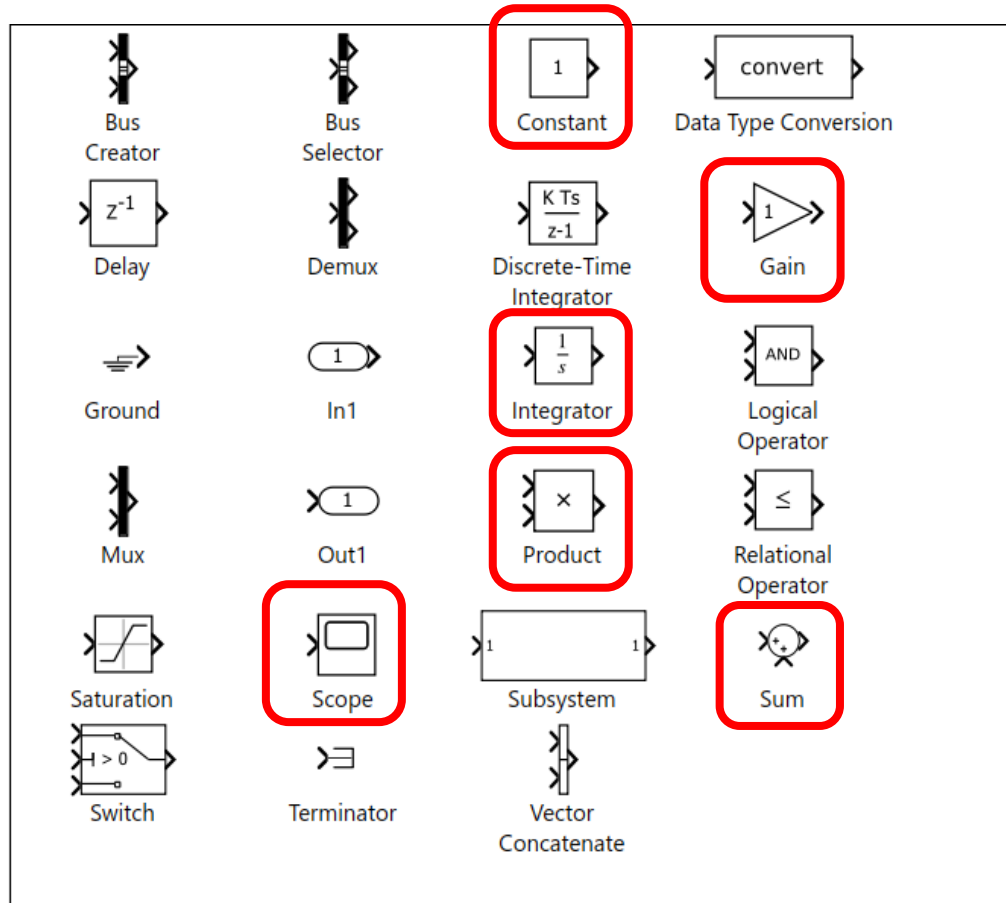
I terminali di ingresso e di uscita compaiono tipicamente sul lato sinistro e sul lato destro del blocco. In dipendenza dalle funzioni, alcuni blocchi avranno solo un terminale di uscita, altri solo un terminale di ingresso, altri ancora avranno sia terminali di ingresso che terminali di uscita.

Il blocco **Integrator** integra il segnale connesso al terminale di ingresso e restituisce il risultato nel terminale di uscita.

Il blocco **Product** moltiplica fra loro i segnali connessi ai due terminali di ingresso, restituisce il risultato nel terminale di uscita.

La prima libreria della lista è la Libreria “**Commonly used blocks**”

COMMONLY USED BLOCKS



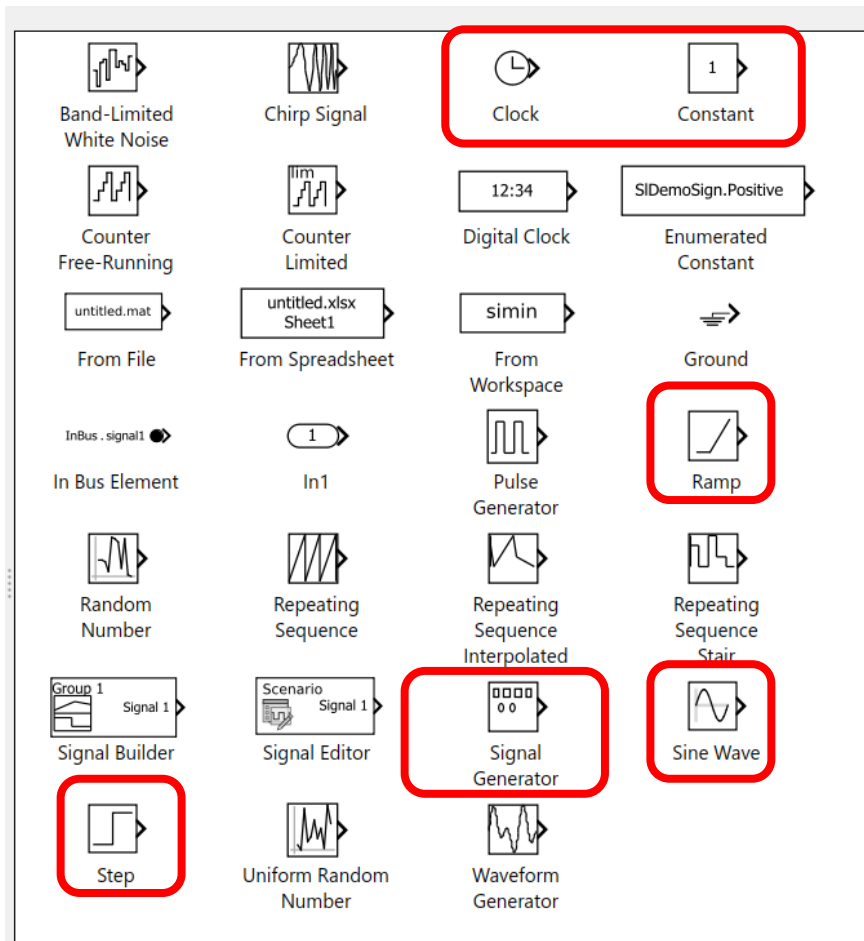
Il blocco **Scope** visualizza in una finestra grafica l'evoluzione temporale del segnale collegato al terminale di ingresso. Come tale, il blocco Scope ha unicamente un terminale di ingresso.

Il blocco **Sum**, infine, **somma** fra loro i segnali connessi ai due terminali di ingresso, restituisce il risultato nel terminale di uscita.

Tutti i blocchi presenti nella libreria “**Commonly used blocks**” sono anche inseriti all'interno di altre librerie contenenti blocchi dalle funzionalità in qualche modo «simili».

I blocchi Simulink si importano nella pagina di lavoro direttamente dalla libreria che li ospita mediante **drag-and-drop**.

SOURCES



Realizziamo alcune simulazioni preliminari in cui si generano e visualizzano alcuni segnali

I blocchi per la generazione di segnali si trovano nella Libreria «*Sources*»

Sono evidenziati il blocco **Clock** che rende accessibile la variabile temporale, il blocco **Constant** già descritto, i blocchi **Ramp**, **Sine Wave** e **Step**, che generano le tre corrispondenti forme d'onda a rampa, sinusoidale, o a gradino, ed il blocco **Signal Generator** che consente di scegliere fra 4 diverse tipologie mediante un menu presente nella **finestra di configurazione del blocco**.

Ogni blocco prevede un set di uno o più parametri di configurazione (ad esempio, per il blocco «Gain» il valore del guadagno). I parametri di configurazione di un blocco sono settati all'interno di una **finestra di configurazione** alla quale si accede facendo **doppio click sul blocco**.

Realizzazione di un modello Simulink

3 fasi

1. Importare nella pagina di lavoro i blocchi elementari Simulink necessari trascinandoli con il mouse dalla rispettiva libreria (drag-and-drop)
2. Parametrizzare i blocchi Simulink nelle rispettive finestre di parametrizzazione, alle quali si accede facendo doppio click con il mouse sopra il blocco stesso.
3. Collegare tra loro i blocchi Simulink tracciando le opportune linee di interconnessione in modo da realizzare le funzionalità desiderate

Esempio introduttivo: costruzione e visualizzazione di un segnale sinusoidale

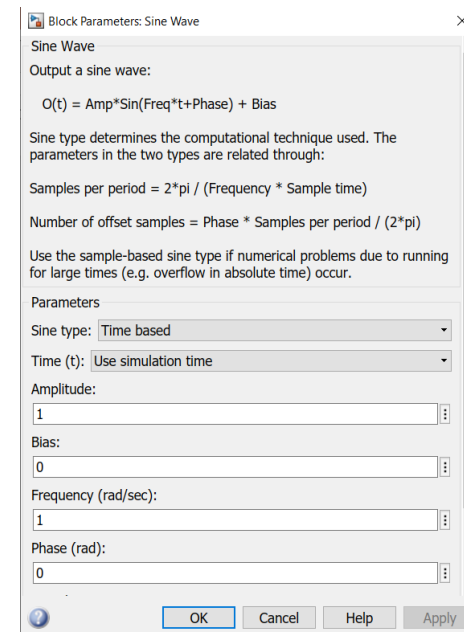
Sono sufficienti due blocchi elementari: un blocco che generi il segnale desiderato, ed un blocco che ne permetta la visualizzazione.

Il primo blocco lo troveremo anche nella libreria “*Sources*” (blocco *Sine Wave*),
Il secondo blocco (blocco *Scope*), si trova nella libreria dei *Commonly Used Blocks* ma anche nella libreria “*Sinks*”.

I blocchi necessari vanno importati nella pagina di lavoro *Untitled* trascinando con il mouse (*drag-and-drop*) l'icona del blocco all'interno della pagina di lavoro. Importiamo il blocco *Sine Wave*.

Si vuole generare il segnale $10 + 5 \sin(t)$.

Per impostare i parametri della sinusoide fare doppio click sul blocco. Si apre una finestra di dialogo all'interno della quale vanno impostati i suoi parametri di funzionamento.



Sine Wave

Output a sine wave:

$$O(t) = \text{Amp} * \sin(\text{Freq} * t + \text{Phase}) + \text{Bias}$$

Puo essere impostato anche un valore costante di **Bias** (v. formula in alto).
Al termine della configurazione premere il tasto OK.

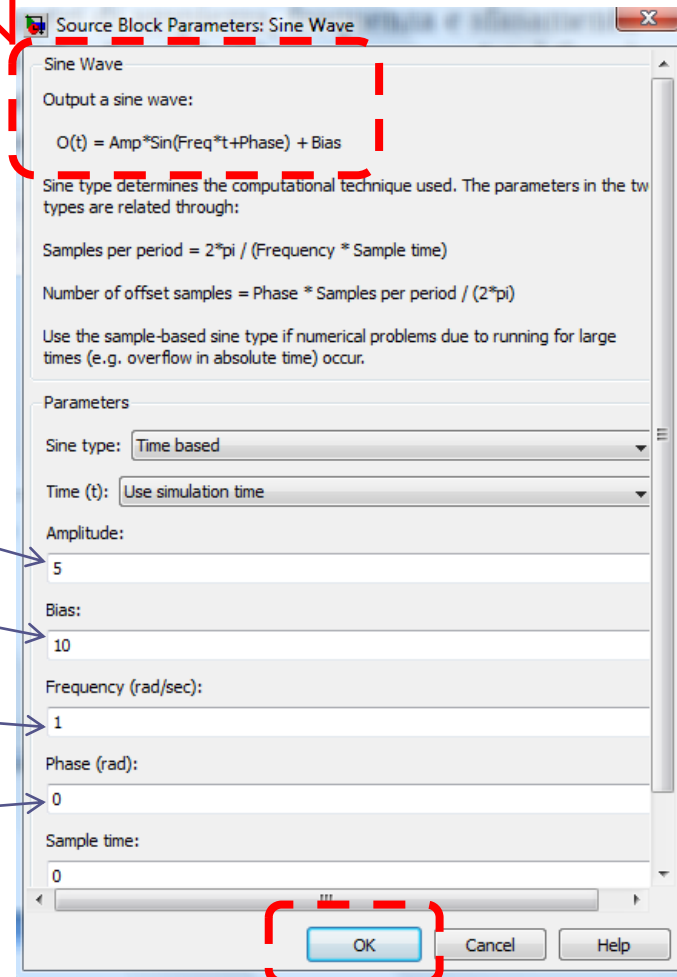
Ampiezza

Bias

Frequenza

Sfasamento

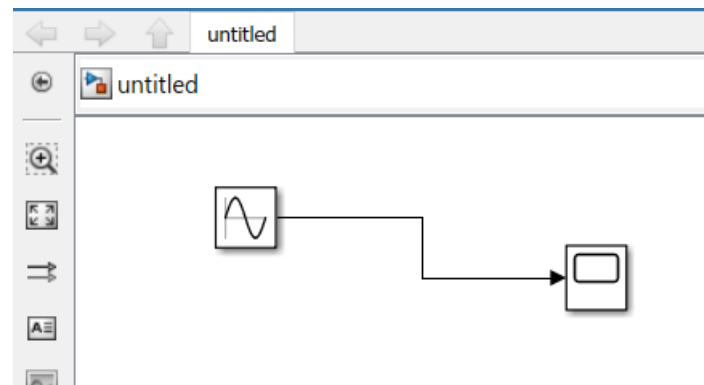
Tasto OK



Si deve ora collegare l'uscita del generatore di funzione "Sine Wave" con l'ingresso del blocco di visualizzazione "Scope".

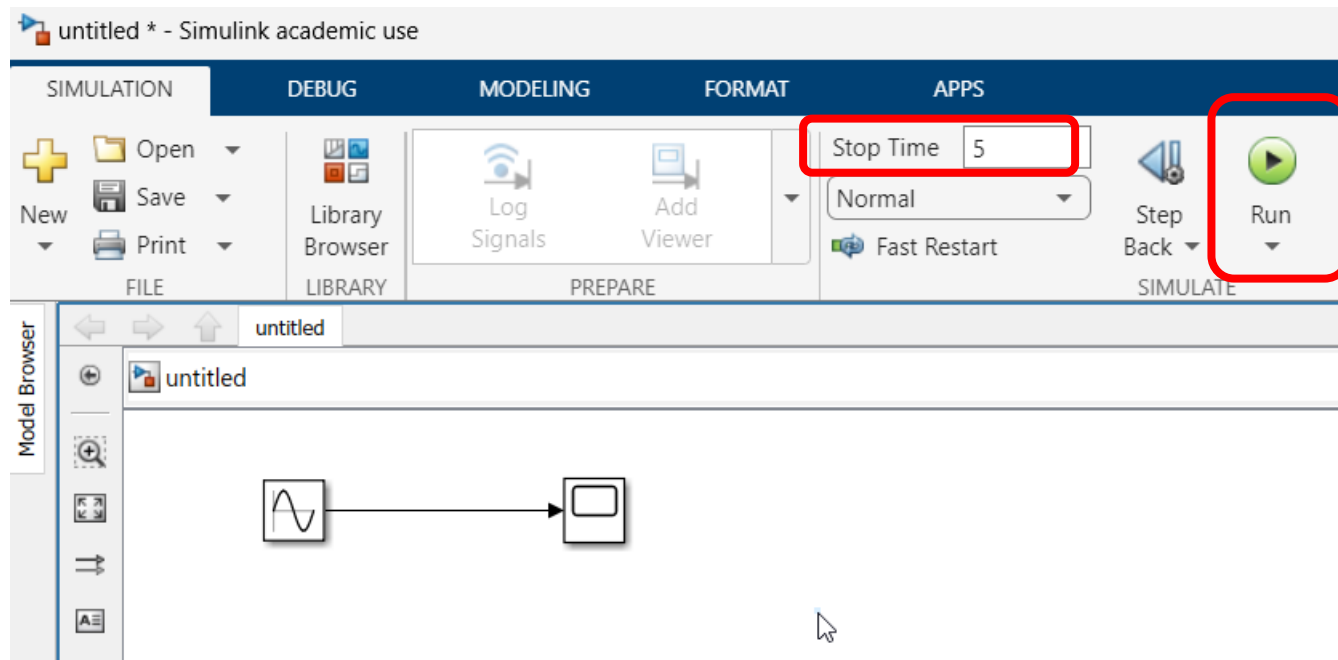
Per effettuare un collegamento tra due blocchi vi è una procedura rapida. Si deve selezionare il blocco di origine (cliccandovi sopra), e si deve successivamente selezionare il blocco di destinazione con il tasto **ctrl** premuto.

Un collegamento correttamente eseguito viene indicato come in Figura

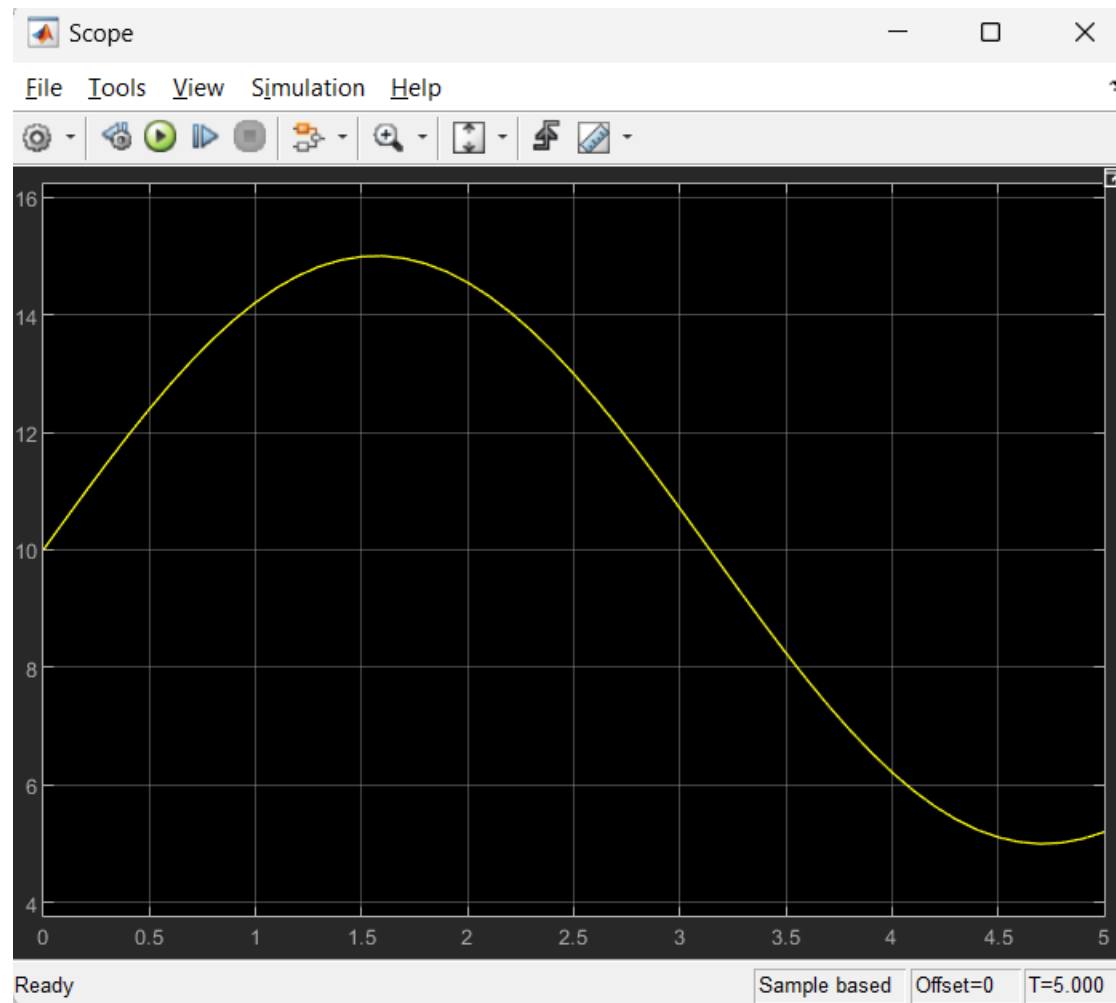


In alternativa, si può portare la freccia del mouse nel punto di inizio della linea di collegamento e quindi "tracciarla" tenendo premuto il tasto sinistro del mouse, fino al punto di destinazione. Le linee di collegamento, così come i blocchi, possono essere rimossi dallo schema con il tasto `cancel`.

Impostare nella casella **Stop Time** la durata della simulazione (il valore di default è 10 secondi), e cliccare sul **pulsante Run** per avviare la simulazione.



Al termine della simulazione fare doppio click sul blocco Scope per visualizzare il segnale in una finestra grafica:



Ora si aumenti la frequenza della sinusoide da 1 rad/s a 2π rad/s (1 Hz).

Si ripeta la simulazione.

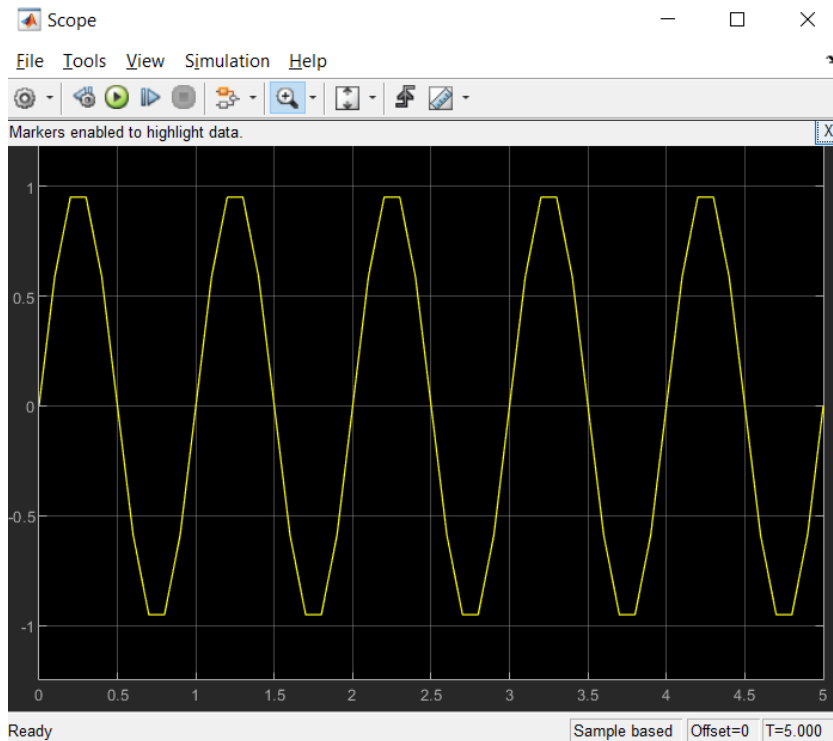
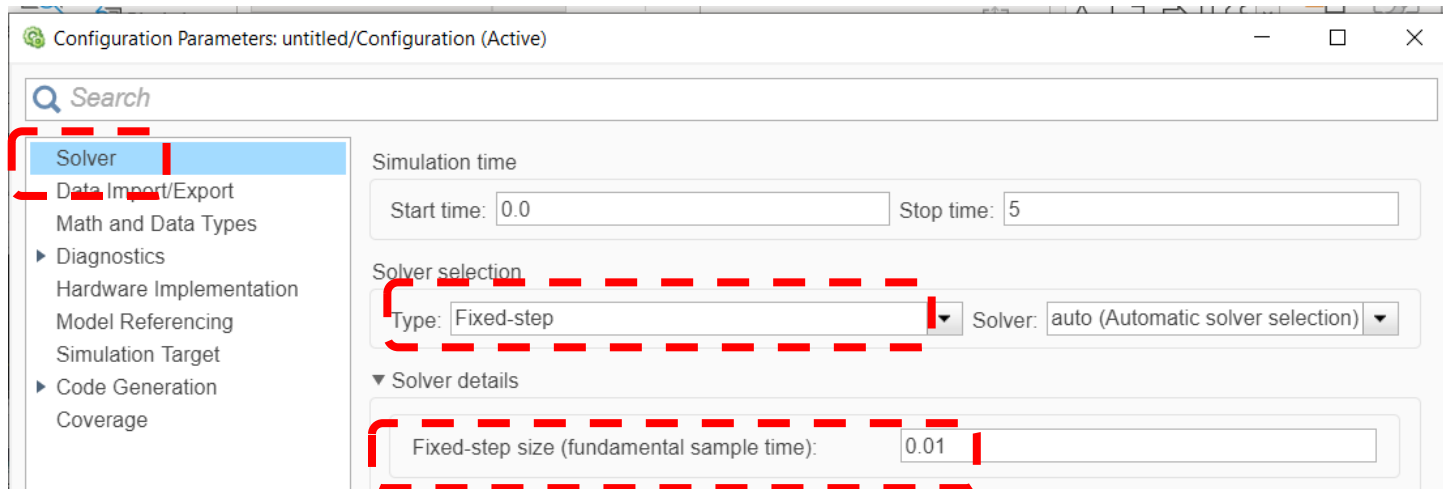


Grafico “spigoloso”

Il grafico è stato realizzato interpolando un numero di punti insufficiente.

Si deve andare a modificare il “solver”, che definisce (fra le altre cose) il passo di discretizzazione temporale che viene impiegato nella esecuzione del modello.

Fare click con il tasto destro in qualunque punto dello schema e scegliere dal menu «Model Configuration Parameters» (Ctrl+E).



Nel menu Solver impostare il Solver Type ed il Fixed Step Size come in figura. In questo modo i segnali della simulazione saranno aggiornati (ricalcolati) ogni 0.01 secondi.

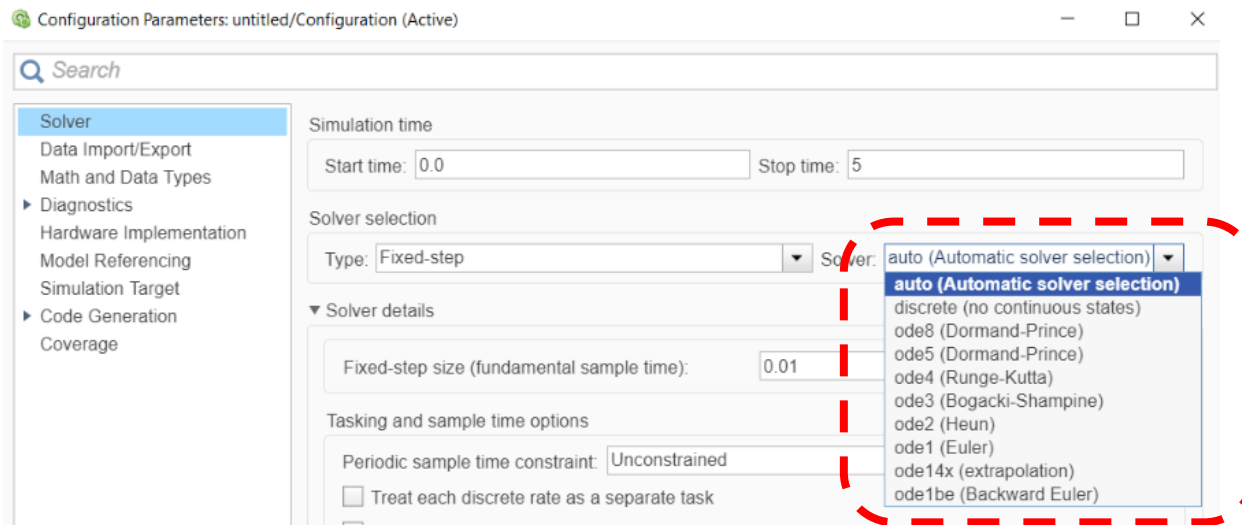
Queste impostazioni vanno eseguite ogni volta che si apre un nuovo modello vuoto. Impareremo successivamente come creare un template di configurazione che consenta di creare direttamente un nuovo modello con le impostazioni desiderate.

Naturalmente se si avesse necessita di generare una sinusoide a frequenza molto più elevata la scelta di 0.01 secondi per il fixed-step size potrebbe diventare non più appropriata, ed il valore dovrebbe essere ulteriormente ridotto.

E' disponibile una ampia varietà di solutori numerici. Nello screenshot sottostante sono riportate le varie opzioni di scelta per i solutori a passo fisso. La scelta del solutore con il quale di risolvono numericamente le equazioni differenziali del modello è ovviamente irrilevante per il semplice esempio in esame che non coinvolge alcun legame differenziale

Una **scelta ottimale per il solutore** bilancia, per il problema in esame, la precisione della soluzione e la mole di calcoli richiesta, che influenza il tempo di simulazione.

L'impostazione di scelta automatica del solver, che è la scelta di default, può dar luogo a soluzioni inaccurate in taluni casi «critici» come ad esempio modelli differenziali in cui intervengono segnali con scale di variazione temporale molto diverse fra loro (problemi «stiff») o modelli con elementi discontinui (non-smooth dynamics) e sono necessari solutori specifici.



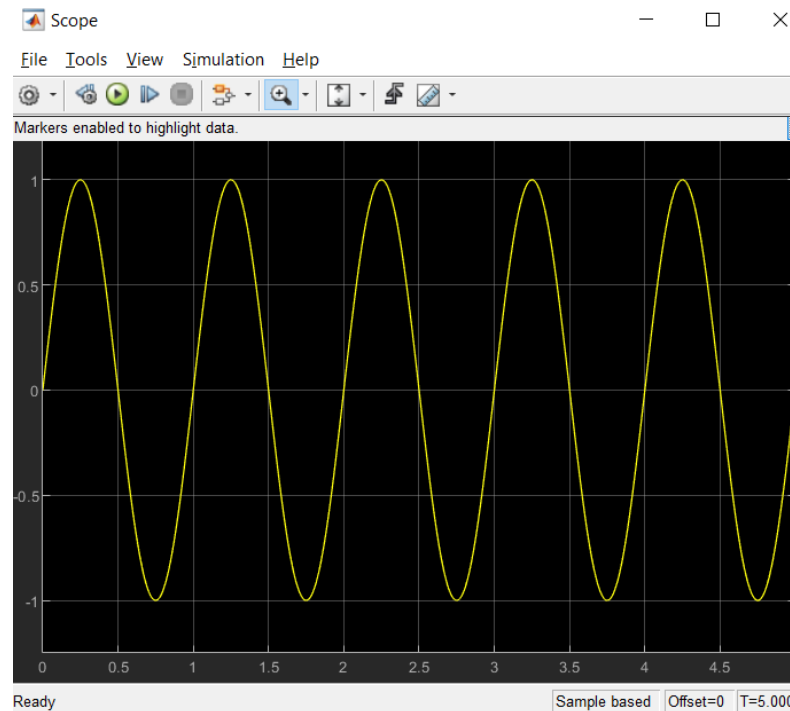
Per una discussione più approfondita in merito alla scelta del solutore:

<https://www.mathworks.com/help/simulink/ug/choose-a-solver.html>

Si ripeta la simulazione e si riaggiorni il grafico

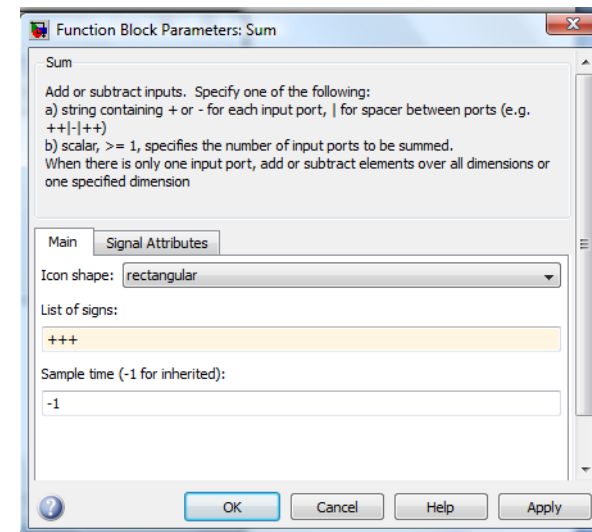
Il grafico della sinusoide è ora correttamente rappresentato.

In base alla scelta fatta per il passo fisso del solutore, vengono ora generati, e interpolati dal grafico, 100 campioni per ogni secondo di simulazione



Per visualizzare un segnale costituito dalla **somma di tre sinusoidi** importiamo nella pagina di lavoro due nuove istanze del blocco elementare Sine Wave, ed importiamo anche un blocco che rappresenti un nodo sommatore (blocco Sum dalla libreria dei Commonly Used Blocks).

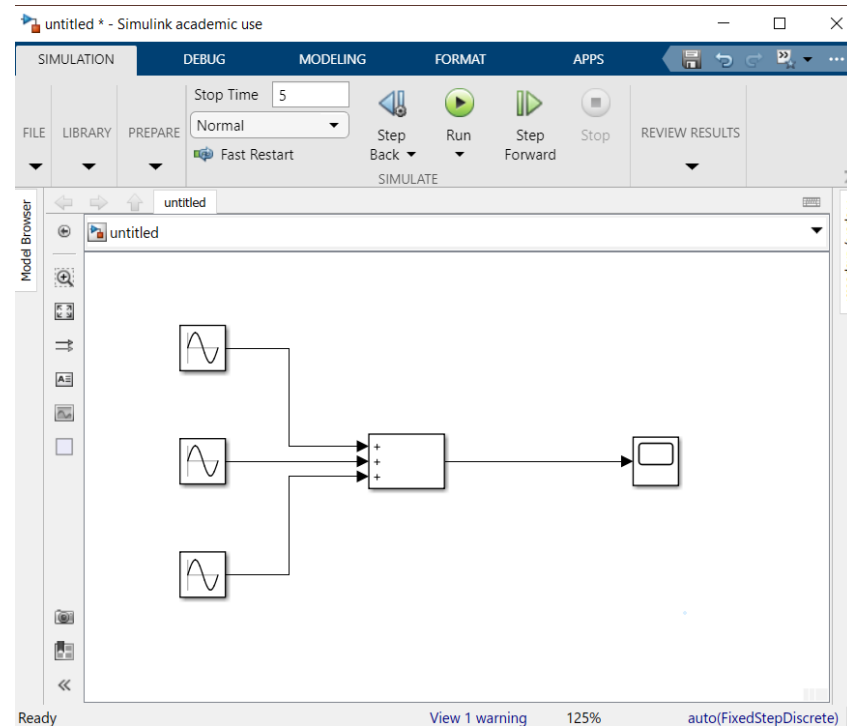
Il blocco Sum è parametrizzato per mezzo di una stringa (es. `++-+ -+`) la cui lunghezza corrisponde al numero di segnali in ingresso al blocco mentre il segno `+` o `-` definisce se il corrispondente ingresso sia da sommare agli altri termini o da sottrarre.



Scegliamo `+++` e impostiamo la Icon shape in rectangular.
L'aspetto del blocco diventa



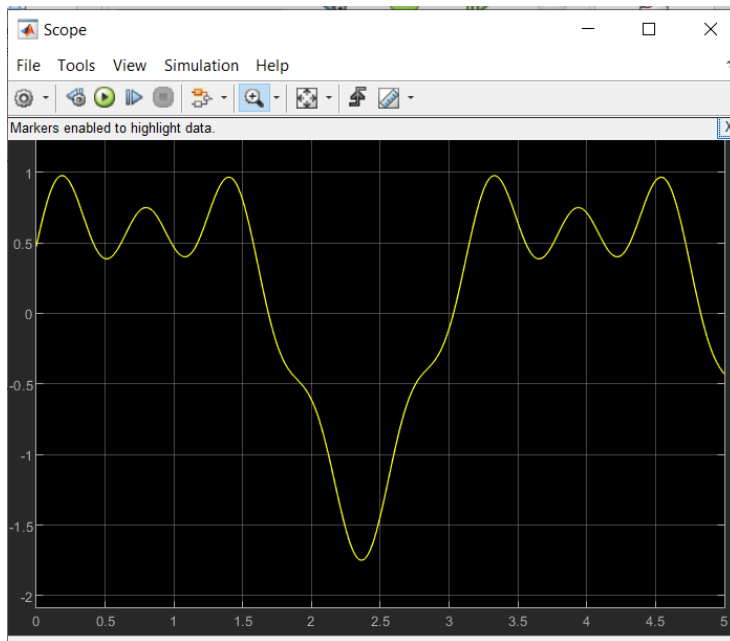
Si realizzi l'interconnessione riportata in Figura.



Si vuole generare il segnale $\sin(2t) + 0.5 \cos(4t) + 0.25 \sin(10t - 0.1)$.

Salvare il file (formato .slx)

	$\sin(2t)$	$0.5 \cos(4t)$	$0.25 \sin(10t - 0.1)$
Amplitude	1	0.5	0.25
Bias	0	0	0
Frequency	2	4	10
Phase	0	$\pi / 2$	-0.1



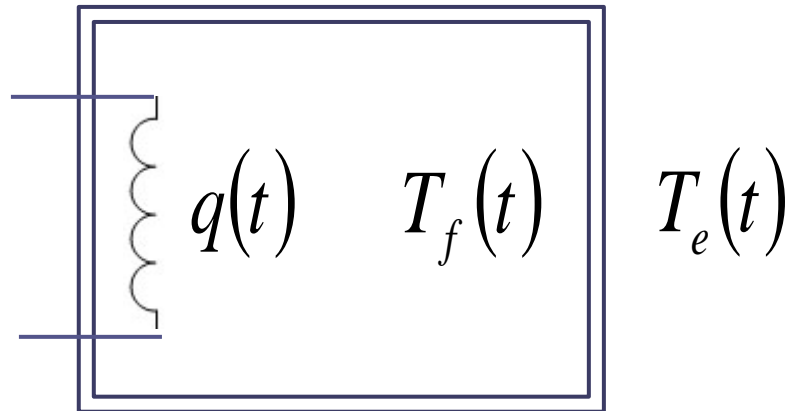
N.B. $\cos(at) = \sin(at + \pi/2)$

File: Crea3Sinusoidi.slx

Realizziamo ora un modello di simulazione dinamica vero e proprio.

Simuliamo un sistema termico

Consideriamo il modello matematico di uno scambiatore di calore (ad es. un **forno**): un volume chiuso circondato da una parete e contenente un fluido all'interno del quale è presente una sorgente di calore $q(t)$



Sia $T_e(t)$ [°C] la temperatura esterna al volume, $T_f(t)$ [°C] la temperatura del fluido interno al volume, e $q(t)$ [J/s] una sorgente di calore interna al volume.

Sia C_f [J/K] la capacità termica del fluido (pari al prodotto fra il calore specifico e la massa contenuta nel volume), e sia K_{ie} (J/K s) il coefficiente di scambio termico tra interno ed esterno.

Modello matematico:

$$C_f \dot{T}_f(t) = K_{ie}(T_e(t) - T_f(t)) + q(t)$$

Modello matematico del fenomeno che si desidera simulare:

$$C_f \dot{T}_f(t) = K_{ie}(T_e(t) - T_f(t)) + q(t)$$

Equazione differenziale lineare del primo ordine

Quantità note:

Parametri fisici del sistema	C_f, K_{ie}
Ingressi esterni	$q(t), T_e(t)$
Condizione iniziale	$T_f(0) = T_{f0}$

Quantità da determinare:

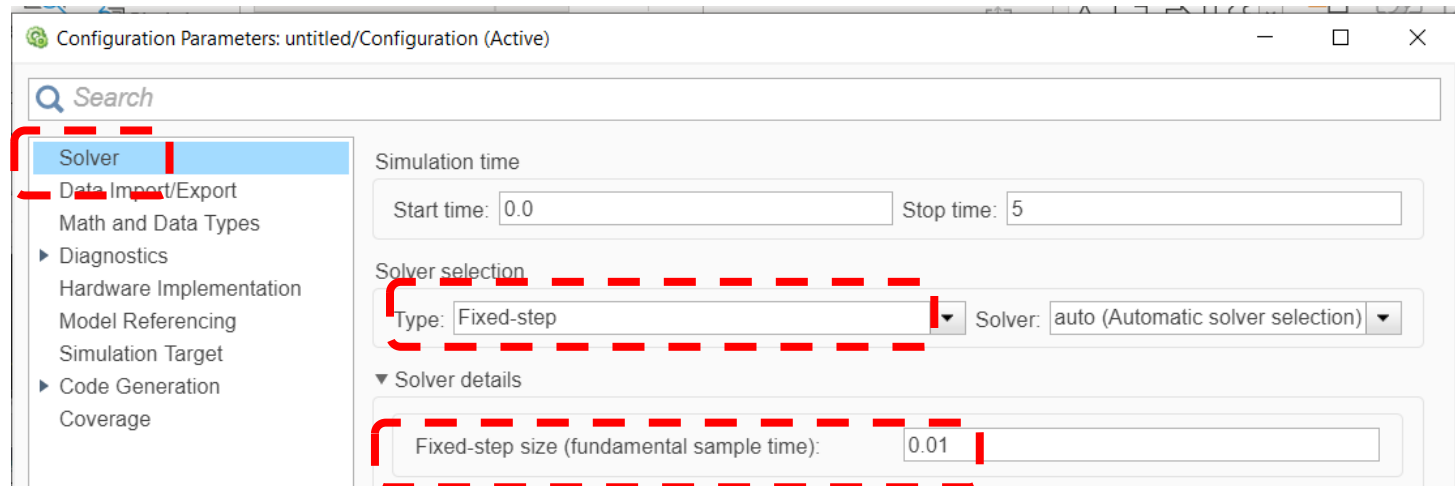
Funzione incognita	$T_f(t)$
--------------------	----------

Simulink risolve **numericamente** l'equazione differenziale, e rende accessibili i valori della funzione incognita in un certo insieme di istanti discreti $T_f(t_1), T_f(t_2), \dots$

Creazione di un template

Creiamo un modello template che sia parametrizzato mediante un solutore a passo fisso, con scelta automatica del particolare solutore, e valore del Fixed Step Size pari a 0.01 secondi.

Apriamo un Blank Model dalla Simulink Start Page e configuriamo le impostazioni desiderate dal Menu «Model Configuration Parameters».



Fatto ciò, dopo avere premuto i pulsanti «Apply» ed «Ok», salvare il modello (pulsante Save) selezionando «Template»

Export untitled to Model Template

Create a template from a model to reuse or share the settings and contents of the model without copying the model each time.

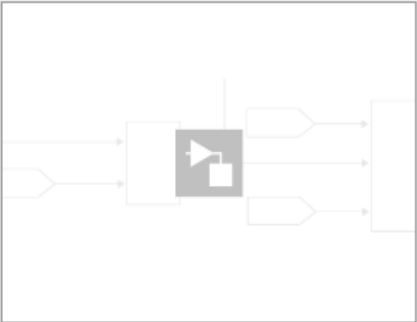
Title: Fixed Step Automatic Solver. Step=0.01.

Author: Alessandro Pisano

Group: My Templates

Description:

Modello configurato con un solutore a passo fisso, scelta automatica del particolare solutore, e fixed step size di 0.01.

Image: 

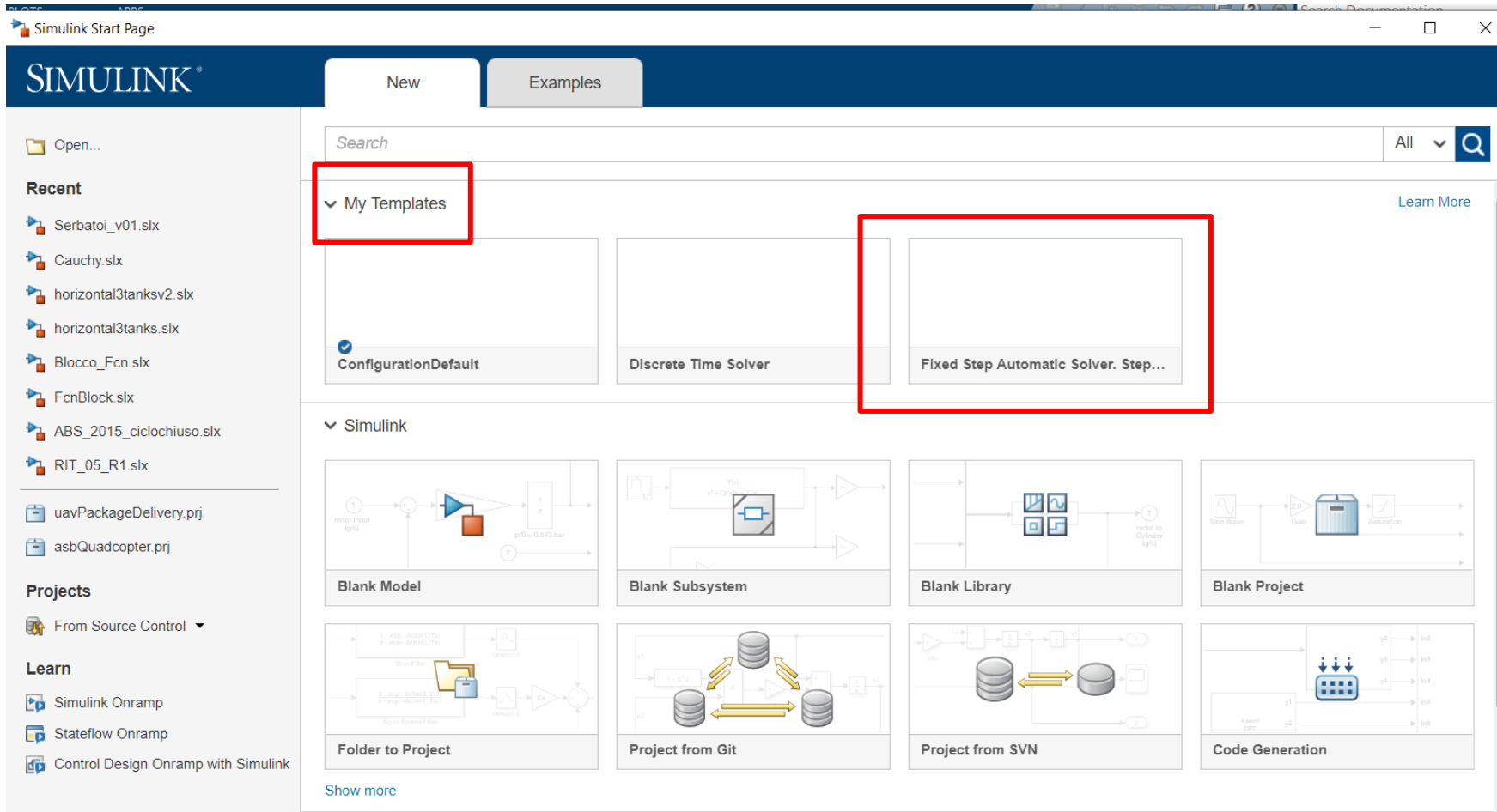
Change...

File location: C:\Users\pisan\Documents\MATLAB\FixedStep_do101.sltx

Browse...

Help Export Cancel

Il Template è ora disponibile nella sezione My Templates della Simulink Start Page.



Per tradurre una equazione differenziale in uno «schema a blocchi» che possa essere implementato in Simulink per modellarla bisogna seguire una procedura sistematica

Passo 1: Riscrivere l'equazione differenziale in **forma esplicita** (cioè isolando alla sinistra dell'uguale la derivata di ordine più elevato della funzione incognita)

$$C_f \dot{T}_f(t) = K_{ie}(T_e(t) - T_f(t)) + q(t)$$

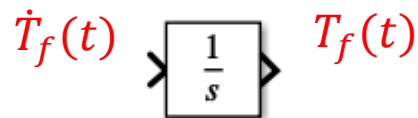


$$\dot{T}_f(t) = \frac{K_{ie}}{C_f}(T_e(t) - T_f(t)) + \frac{1}{C_f}q(t)$$

Passo 2: Importare nella pagina di lavoro (aprirne una ex-novo a partire dal template creato poco prima) un numero di blocchi «Integrator» (libreria Commonly Used Blocks) pari all'ordine della equazione differenziale (1 in questo caso).

Ora completeremo lo schema in modo tale che la temperatura $T_f(t)$ del fluido sia generata al terminale di uscita del blocco integratore

Se all'uscita del blocco integratore è presente la temperatura $T_f(t)$ del fluido , necessariamente all'ingresso del blocco dovrà essere presente il segnale rappresentativo della derivata $\dot{T}_f(t)$ (v. figura)



L'equazione differenziale riscritta in forma esplicita fornisce l'espressione di $\dot{T}_f(t)$ in funzione delle altre costanti e variabili presenti nel modello. Il segnale da applicare in ingresso all'integratore è costruito implementando tale espressione mediante gli opportuni blocchi Simulink

$$\dot{T}_f(t) = \frac{K_{ie}}{C_f} (T_e(t) - T_f(t)) + \frac{1}{C_f} q(t)$$

Si inserisca a monte dell'integratore un sommatore con due terminali positivi di ingresso. Costruiamo separatamente i due contributi $\frac{K_{ie}}{C_f} (T_e(t) - T_f(t))$ e $\frac{1}{C_f} q(t)$ da sommare fra loro.

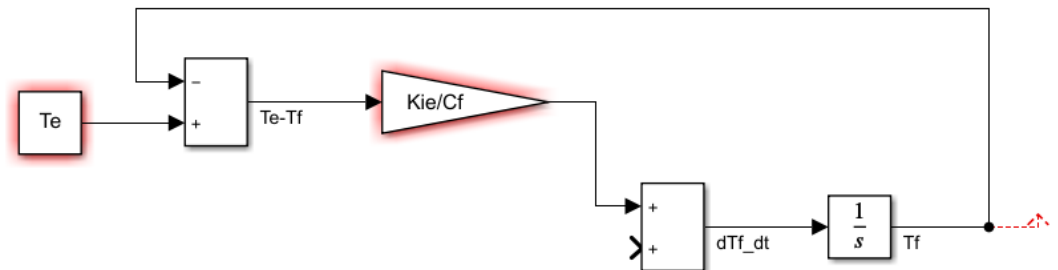
Consideriamo un valore costante T_e per la temperatura esterna $T_e(t)$.

Realizziamo il modello in forma parametrica utilizzando le variabili C_f , K_{ie} , T_e alle quali si attribuirà successivamente un valore mediante un **file script**.

Primo contributo: $\frac{K_{ie}}{C_f} (T_e(t) - T_f(t))$

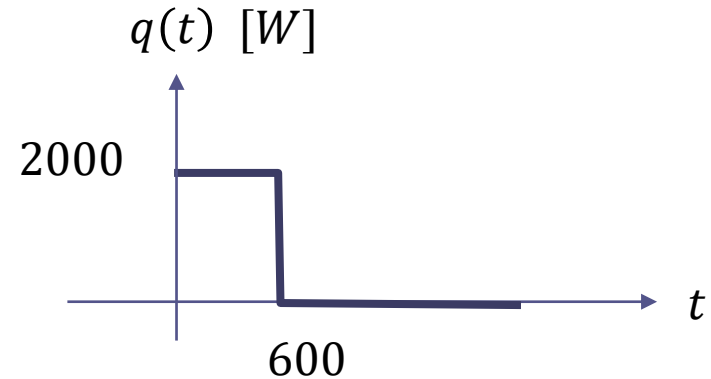
Serve un blocco gain per rappresentare il guadagno $\frac{K_{ie}}{C_f}$, un secondo blocco sommatore per calcolare la differenza $T_e(t) - T_f(t)$ ed un blocco constant per generare la temperatura esterna T_e

La temperatura $T_f(t)$ del fluido è accessibile al terminale di uscita del blocco integratore.

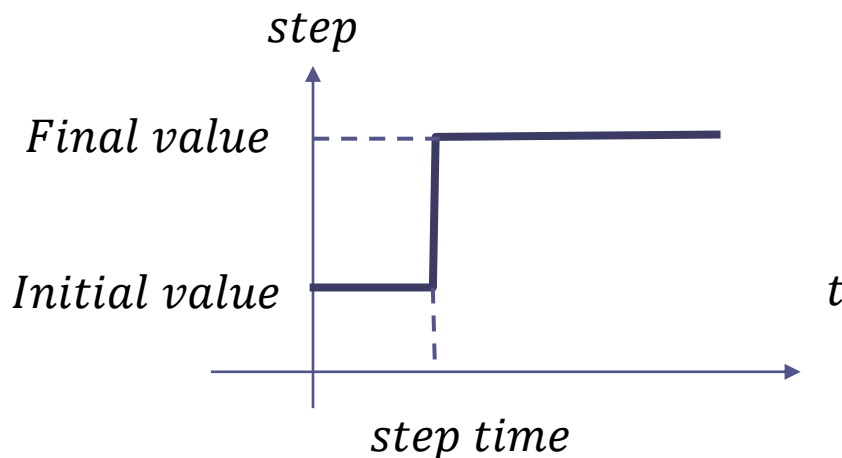


Secondo contributo: $\frac{1}{c_f} q(t)$

Consideriamo il profilo a lato per la sorgente di calore $q(t)$



Realizziamo il segnale $q(t)$ mediante il blocco Step (Libreria Sources). Il blocco step genera un segnale avente la forma riportata nella figura sottostante. I parametri sono i valori iniziale e finale, e l'istante della transizione (step time).

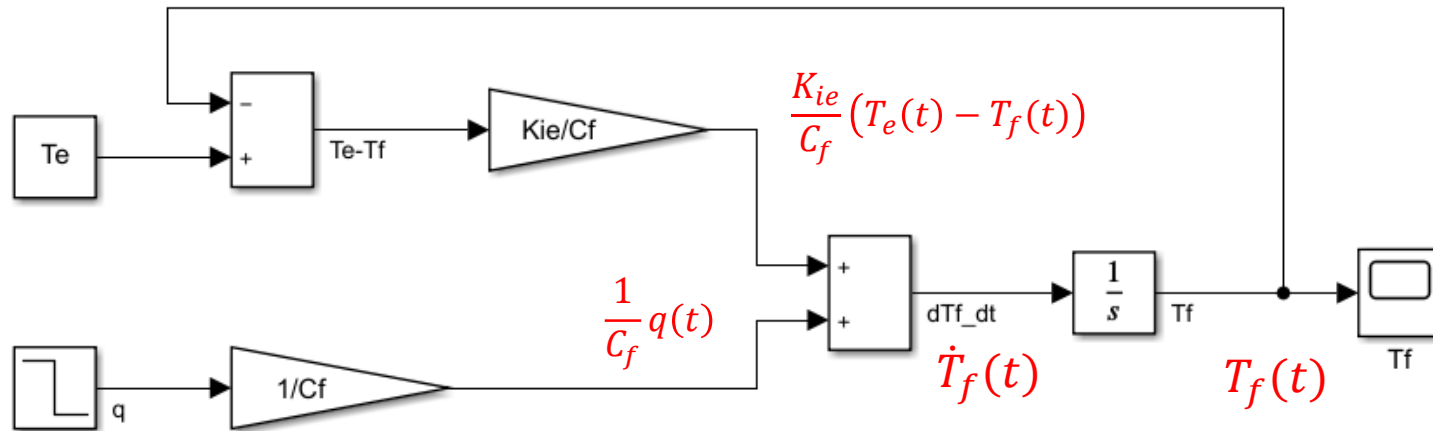


*Parametri di configurazione
del blocco Step*

Initial value = 2000

Final value = 0

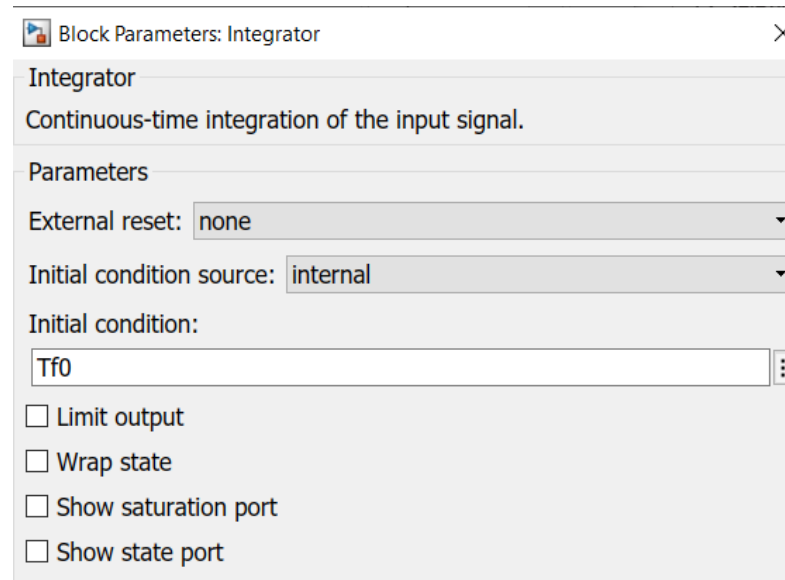
Step time = 600



Si noti come all'ingresso dell'integratore, punto cui corrisponde il segnale dT_f/dt , venga

“costruito” elemento per elemento il segnale $\dot{T}_f(t) = \frac{K_{ie}}{C_f} (T_e(t) - T_f(t)) + \frac{1}{C_f} q(t)$

Il parametro T_{f0} , il valore della temperatura all'istante iniziale, si imposta come parametro di configurazione del blocco Integrator (Initial Condition).

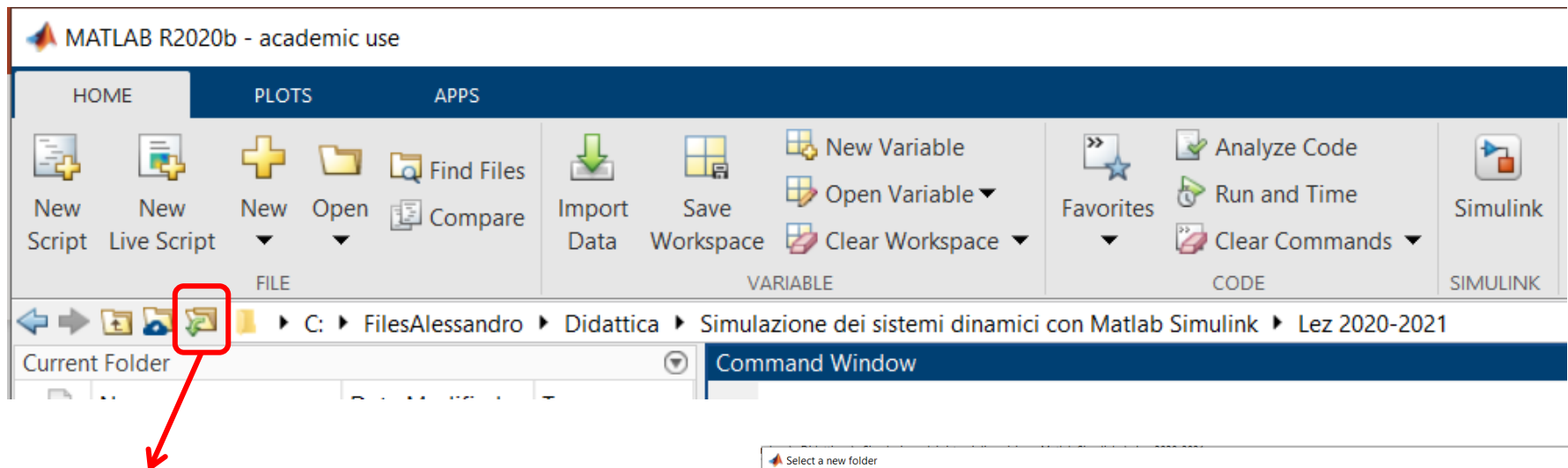


La durata della simulazione si imposta nella casella **Stop Time**. Impostiamo come durata 1200 secondi.

File: Forno.slx

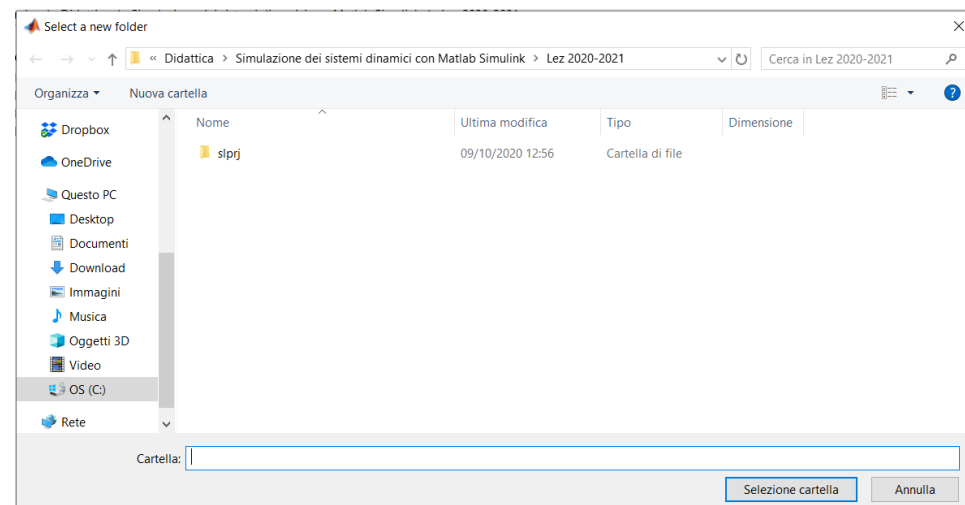
Salviamo nel workspace i parametri necessari alla esecuzione del modello mediante un file **Script**

Scegliere una “current folder” nel PC, all’interno della quale verrà salvato il file



Browse for folder

Si apre la finestra “Select a new folder”. Navigando tra le cartelle del proprio PC sceglierne una (o crearla ex-novo) e poi premere “Selezione cartella”



Ora dalla Home premere il pulsante “New script” per aprire il relativo editor di testo

Iniziamo ad inserire le istruzioni necessarie.

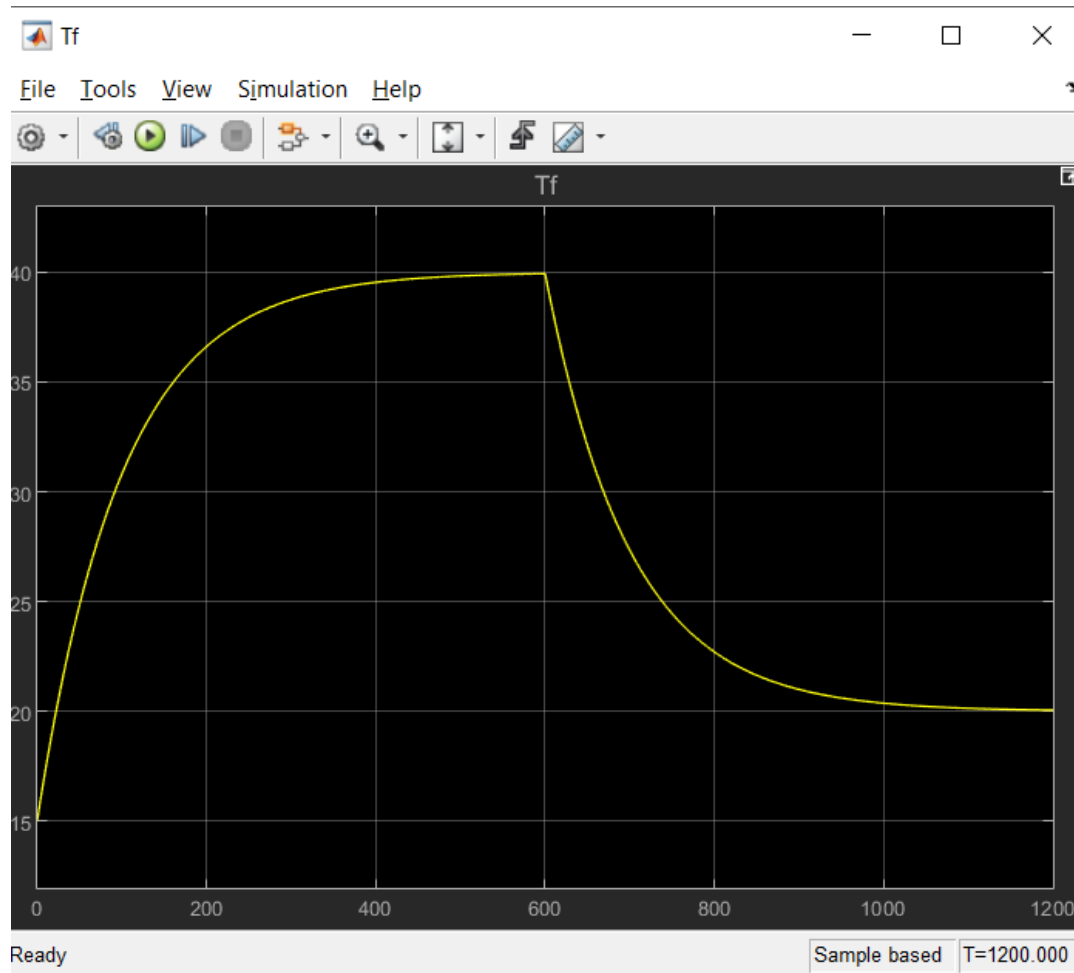
File: Forno_dati.m

```
Tf0=15;      %Temp. iniziale del fluido [°C]
Te=20;      %Temp. esterna [°C]
Cf=1e4;      % J/K per 1 kg di sostanza
Kie=100;     % J / K s , per 1 m^2 di superficie di scambio
q=2000;      % J/s    q=2kW
```

Salviamo il file nella Current Folder (pulsante «Save»).

Che succede se mandiamo in run questo script ? Controllare le variabili presenti nel workspace di Matlab prima e dopo il run.

Si avvii la simulazione. Facendo doppio click sul blocco Scope si visualizza il grafico temporale della temperatura del fluido interno al volume.



Modellazione di un servomotore in corrente continua

Modelliamo il comportamento dinamico di un sistema elettromeccanico: un servomotore in corrente continua



Applicando una differenza di potenziale ai morsetti di alimentazione si provoca la circolazione di una corrente elettrica che induce una coppia di natura elettromagnetica che mette in rotazione l'albero del motore.

La dinamica del sistema è descritta da un **sistema di equazioni differenziali**.

Modello matematico di un servomotore DC

$$V(t) = Ri(t) + L \frac{di(t)}{dt} + K_V \omega(t)$$

$$T_{em}(t) - T_{res}(t) = J \frac{d\omega(t)}{dt} + B\omega(t)$$

$$T_{em}(t) = K_T i(t)$$

VARIABILI DEL SISTEMA

$v(t)$	tensione di alimentazione	$\omega(t)$	velocità angolare
$T_{em}(t)$	coppia elettromagnetica	$T_{res}(t)$	coppia resistente
$i(t)$	corrente di fase		

PARAMETRI ELETTROMECCANICI

K_V	costante di forza contro-elettromotrice	k_T	costante di coppia
R	resistenza dell'avvolgimento	J	momento di inerzia all'albero
L	induttanza di avvolgimento	B	coefficiente di attrito viscoso

Ricaviamo le equazioni differenziali **in forma esplicita**

$$V(t) = Ri(t) + L \frac{di(t)}{dt} + K_V \omega(t)$$

$$T_{em}(t) - T_{res}(t) = J \frac{d\omega(t)}{dt} + B\omega(t)$$

$$T_{em}(t) = K_T i(t)$$



$$V(t) = Ri(t) + L \frac{di(t)}{dt} + K_V \omega(t)$$

$$K_T i(t) - T_{res}(t) = J \frac{d\omega(t)}{dt} + B\omega(t)$$



$$\frac{di(t)}{dt} = \frac{1}{L} [V(t) - Ri(t) - K_V \omega(t)]$$

$$\frac{d\omega(t)}{dt} = \frac{1}{J} [K_T i(t) - B\omega(t) - T_{res}(t)]$$

Così come per la rappresentazione in Simulink di una equazione differenziale, anche per quanto concerne la rappresentazione in Simulink di un sistema di equazioni differenziali il punto di partenza è la **risrittura del sistema di equazioni in forma esplicita**

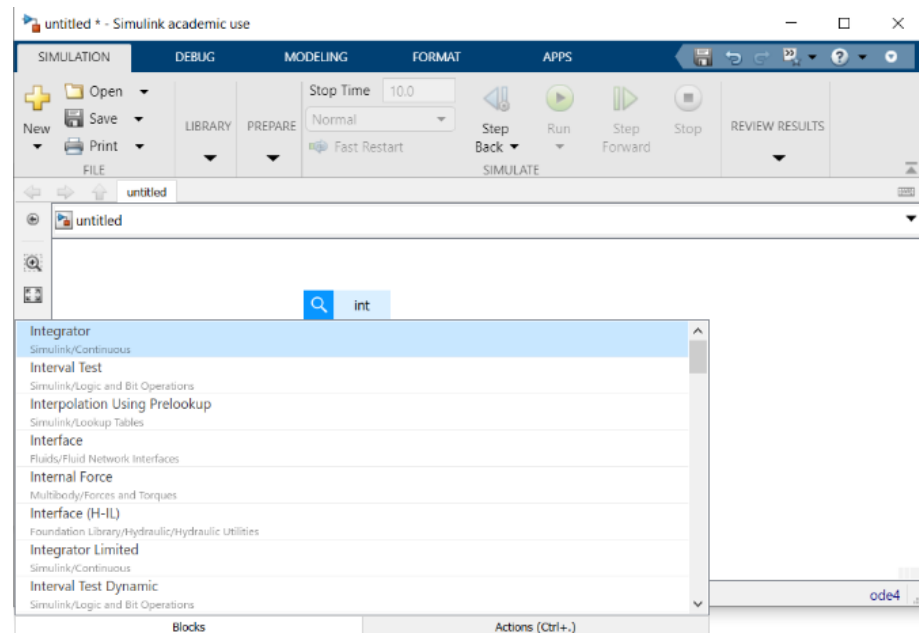
$$\begin{aligned}\frac{di(t)}{dt} &= \frac{1}{L} [V(t) - Ri(t) - K_V \omega(t)] \\ \frac{d\omega(t)}{dt} &= \frac{1}{J} [K_T i(t) - B\omega(t) - T_{res}(t)]\end{aligned}$$

All'interno delle equazioni del modello la tensione applicata $V(t)$ e la coppia resistente $T_{res}(t)$ (che dipende dal carico applicato all'albero del motore) sono grandezze note (ingressi al sistema). Le grandezze incognite sono la velocità angolare $\omega(t)$ e la corrente di fase $i(t)$.

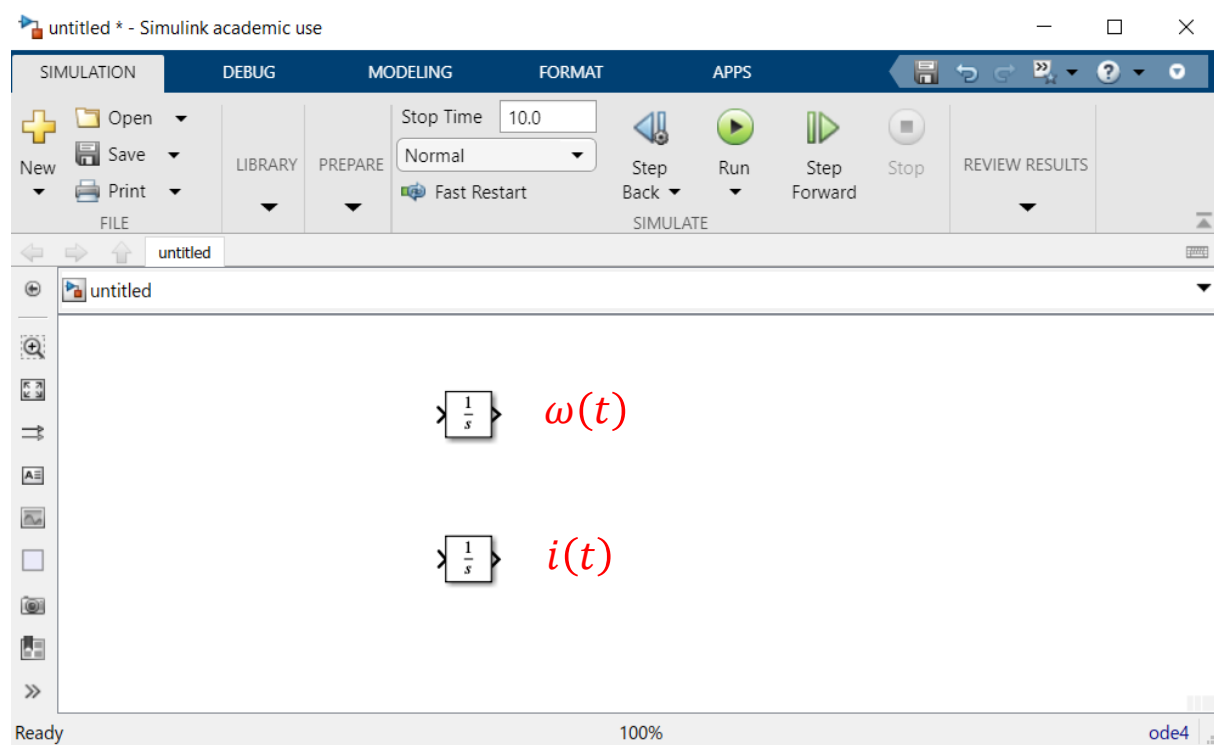
Modelliamo in Simulink il sistema di equazioni. La procedura, come si vedrà, è pressochè identica a quella seguita per modellare il sistema termico ed il sistema massa-molla-smorzatore, con lievi differenze.

Dobbiamo importare nella pagina di lavoro un numero di blocchi «Integrator» pari all'ordine complessivo del sistema (in questo caso 2, perchè il sistema si compone di due equazioni accoppiate del primo ordine)

Per importare dei blocchi all'interno della pagina di lavoro è possibile, in alternativa alla procedura di drag-and-drop dalle rispettive librerie, fare doppio click in un punto qualunque dello schema e scrivere il nome del blocco (o la parte iniziale del nome). In esito a ciò, compare una lista di opzioni da cui scegliere il blocco che ci interessa

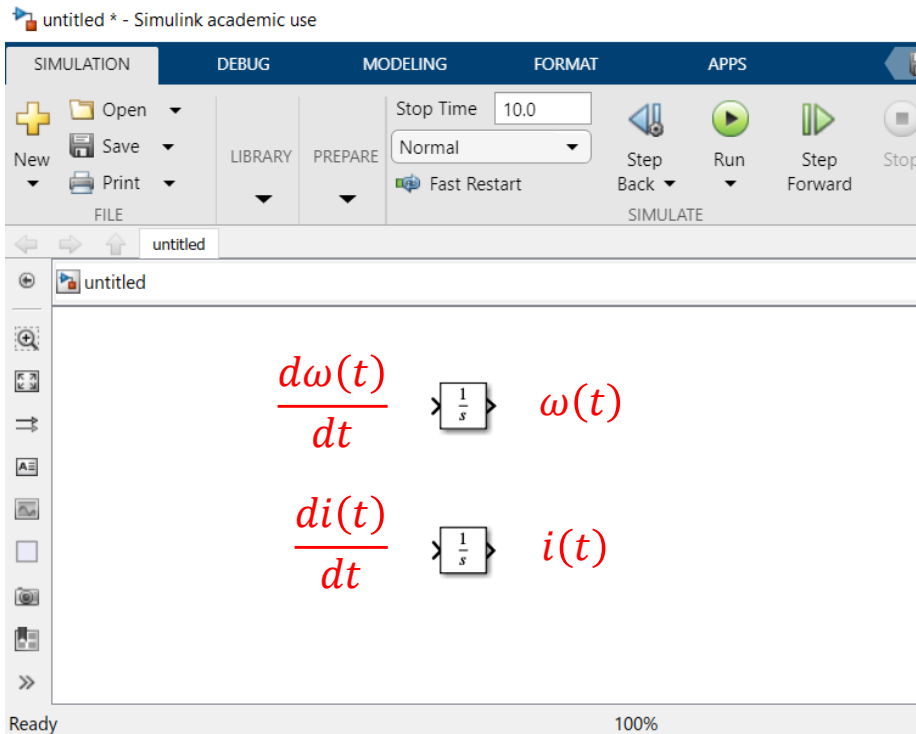


A differenza da quanto fatto in precedenza, non collegheremo più un serie gli integratori, ma li disporremo su righe differenti.



Ora completeremo lo schema in modo tale che le due variabili incognite, la velocità angolare e la corrente di fase, siano generate ai terminali di uscita dei due blocchi integratori

Chiaramente, se ai terminali di uscita degli integratori sono presenti la velocità angolare $\omega(t)$ e la corrente di fase $i(t)$ ai rispettivi terminali di ingresso saranno presenti le rispettive derivate temporali



$$\frac{di(t)}{dt} = \frac{1}{L} [V(t) - Ri(t) - K_V \omega(t)]$$

$$\frac{d\omega(t)}{dt} = \frac{1}{J} [K_T i(t) - B\omega(t) - T_{res}(t)]$$

Per modellare il funzionamento del motore debbono essere «costruite» ed applicate in ingresso ai due integratori le espressioni dei segnali $\frac{d\omega(t)}{dt}$ e $\frac{di(t)}{dt}$, definite dalle equazioni in forma esplicita

Realizziamo ed eseguiamo il seguente Script

```
% Parametri modello motore DC
% Parametri elettromeccanici
R=0.8;      % resistenza
L=1e-3;     % induttanza
KT=0.3;     % costante di coppia
Kv=0.3;     % costante di forza c.e.m.
J=1;        % momento di inerzia
B=0.8;      % coefficiente di attrito viscoso

% Ingressi
V=10;       % tensione di alimentazione
Tres=0;     % coppia resistente

% Condizioni iniziali
omega0=0;   %condizione iniziale della velocita
i0=0;       %condizione iniziale della corrente
```

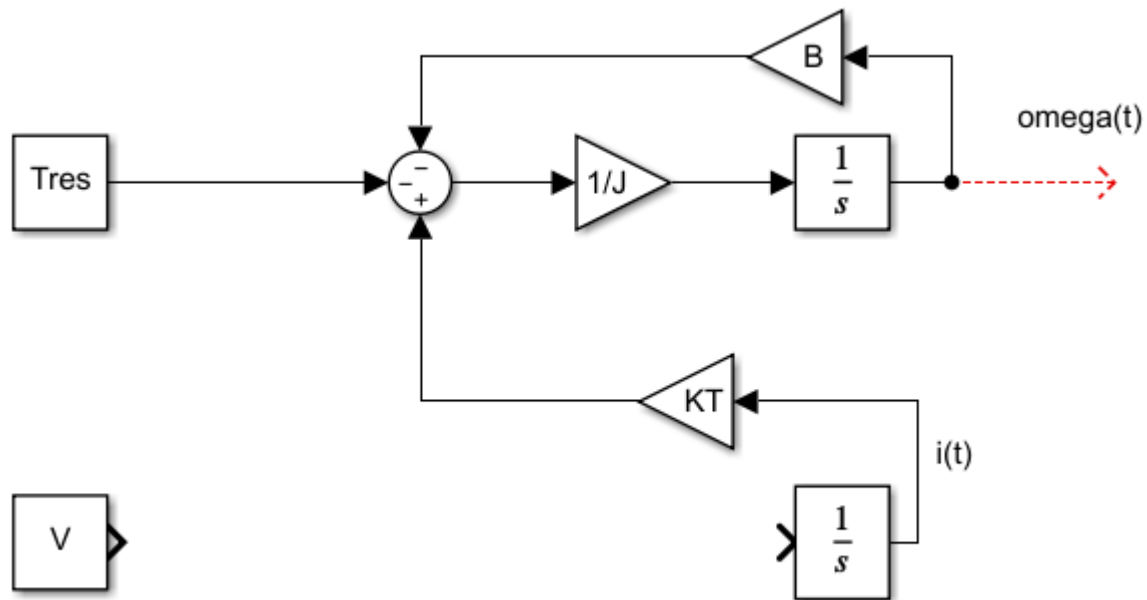
File: MotoreDC_dati.m

Configuriamo i due blocchi integrator inserendo la condizione iniziale del segnale prodotto in uscita. All'interno delle rispettive finestre di parametrizzazione, inserire nel campo «Initial condition» le variabili ω_0 ed i_0

Si desidera realizzare un test di funzionamento a vuoto (cioè con $T_{res}(t) = 0$) con l'applicazione di una tensione di alimentazione $V(t)$ costante.

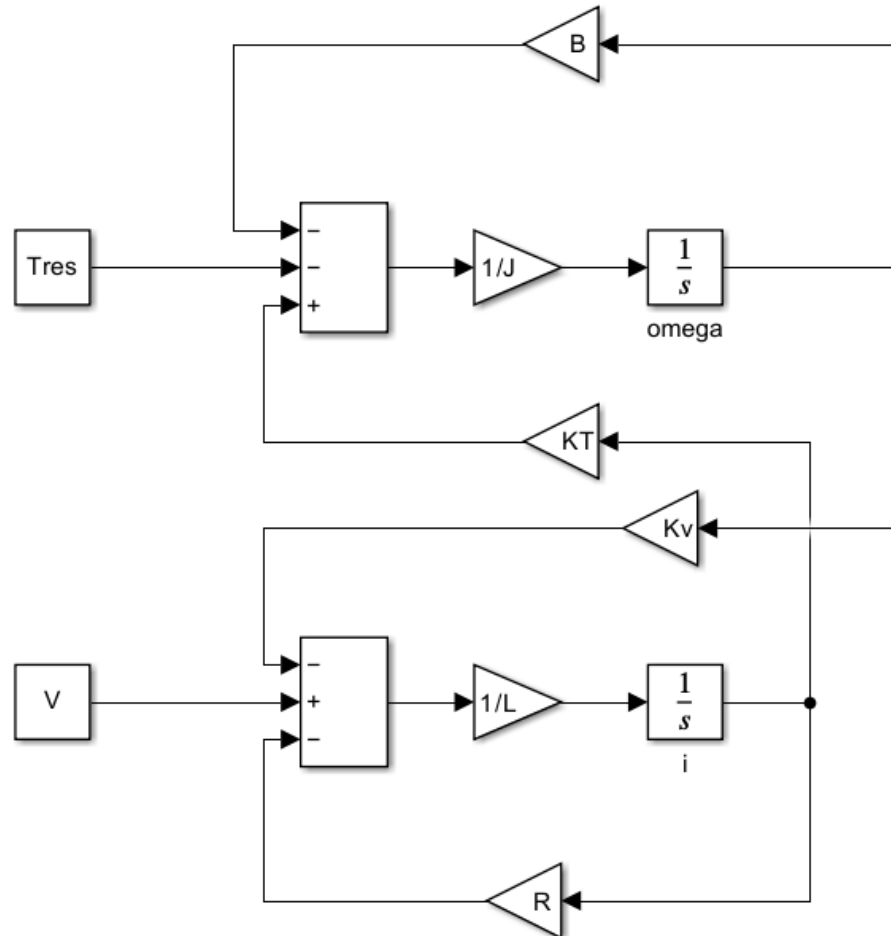
Inseriamo nella pagina di lavoro due blocchi `Constant` che impiegheremo per generare la tensione $V(t)$ applicata al motore e la coppia resistente. Tali blocchi saranno parametrizzati in termini di ampiezza del segnale generato («Constant value») dalle variabili V e T_{res} che sono state definite nello script.

Modello della equazione meccanica $\frac{d\omega(t)}{dt} = \frac{1}{J} [K_T i(t) - B\omega(t) - T_{res}(t)]$



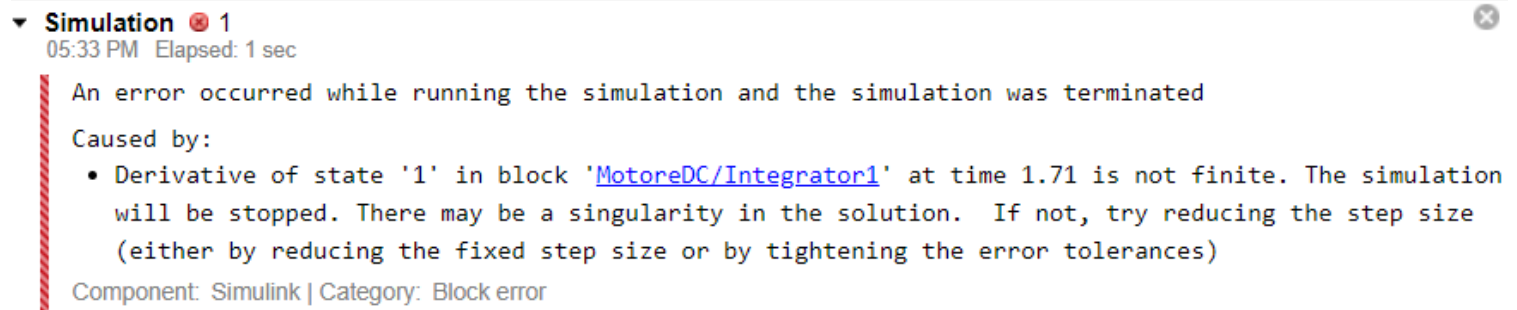
Modello comprensivo anche della equazione elettrica

$$\frac{di(t)}{dt} = \frac{1}{L} [V(t) - Ri(t) - K_V \omega(t)]$$

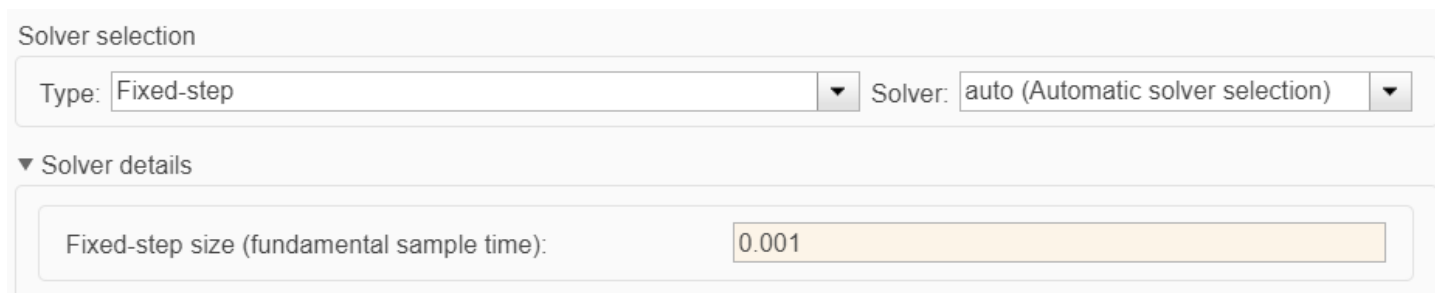


Si desidera realizzare un test di funzionamento a vuoto (cioè con $T_{res}(t) = 0$) con l'applicazione di una tensione di alimentazione $V(t)$ costante di ampiezza pari a 10 V .

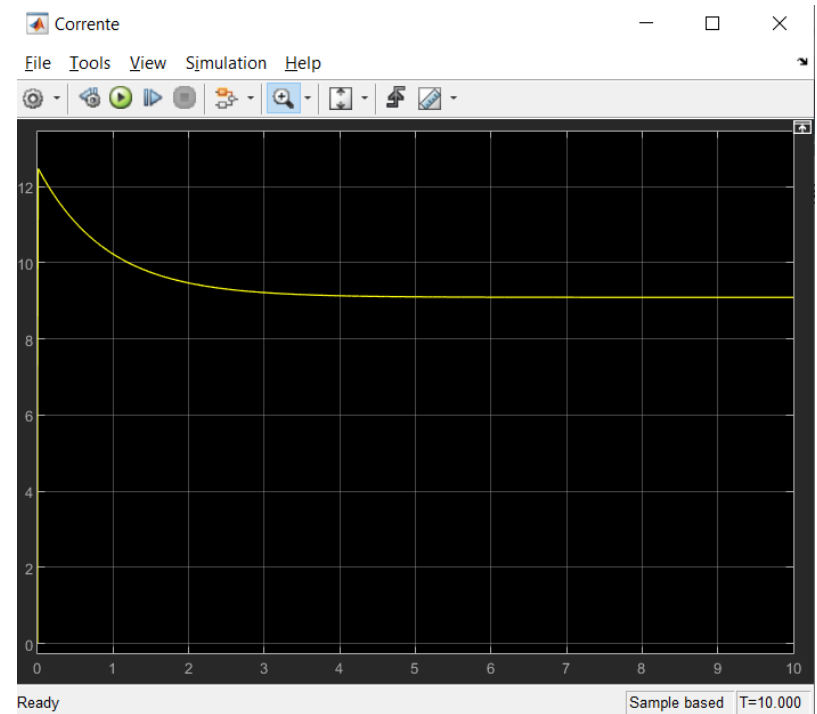
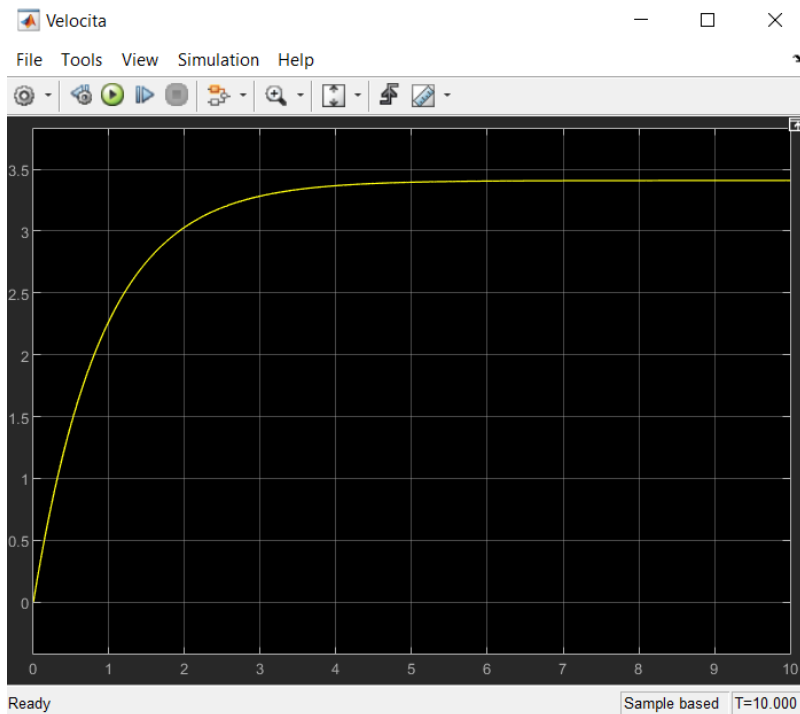
Se si avvia la simulazione utilizzando come fixed step size del solutore numerico il valore di 0.01 (suggerito in precedenza) si ottiene il seguente messaggio di errore



Pur nella sua semplicità, la simulazione numerica di questo modello matematico è resa problematica dal fatto che coesistono nel modello variabili di natura elettrica e di natura meccanica, aventi dinamiche molto differenti. Il solutore numerico con passo 0.01 diverge, e per risolvere il problema bisogna ridurlo fino a 0.001.

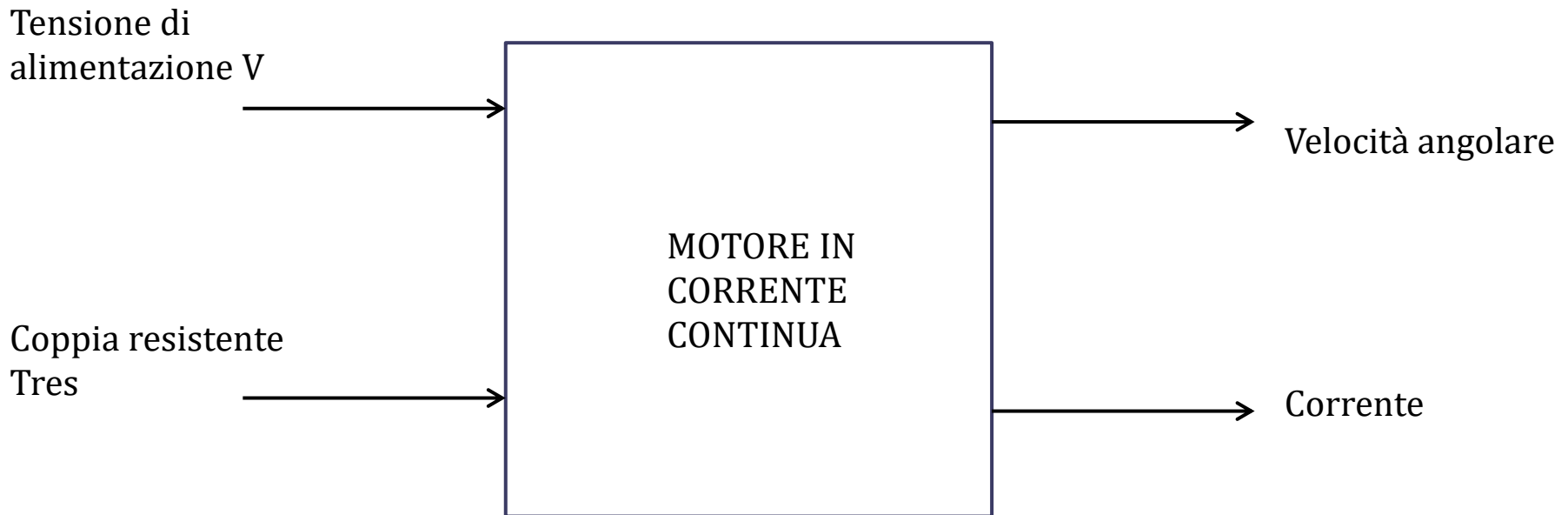


Ora possiamo avviare la simulazione e visualizzare i grafici della velocità angolare e della corrente di fase



Sottosistemi (Subsystem)

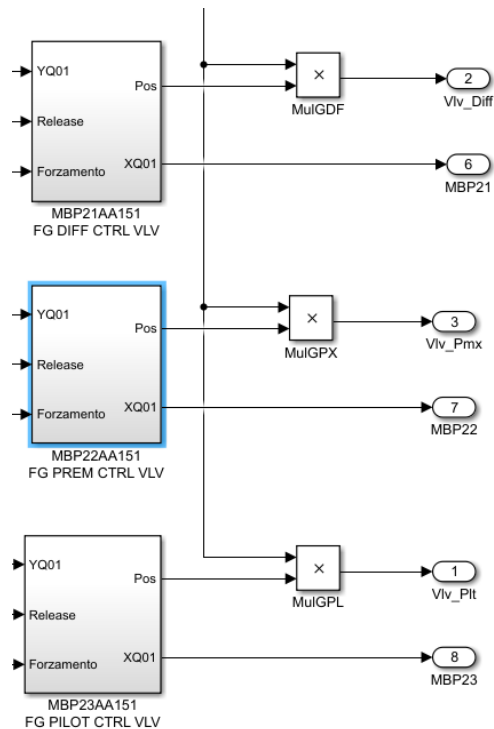
Ora vogliamo compattare il modello simulink del motore DC all'interno di un sottosistema compatto con ingressi e uscite, come mostrato nella figura seguente



Segnali di ingresso

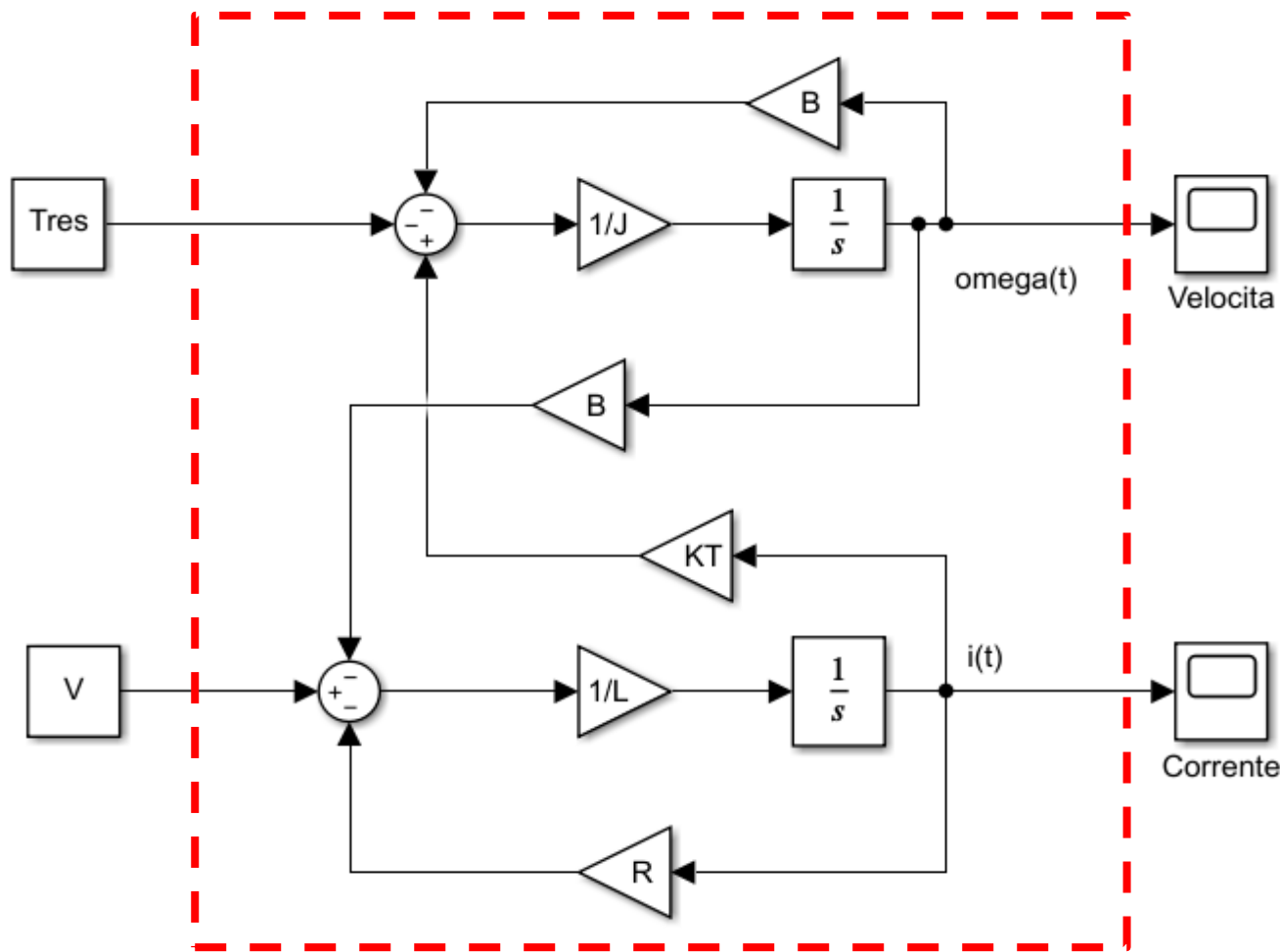
Segnali di uscita

Un sottosistema può essere duplicato all'interno del modello Simulink per rappresentare più istanze di componenti identiche fra loro, o in alternativa importato all'interno di altri modelli

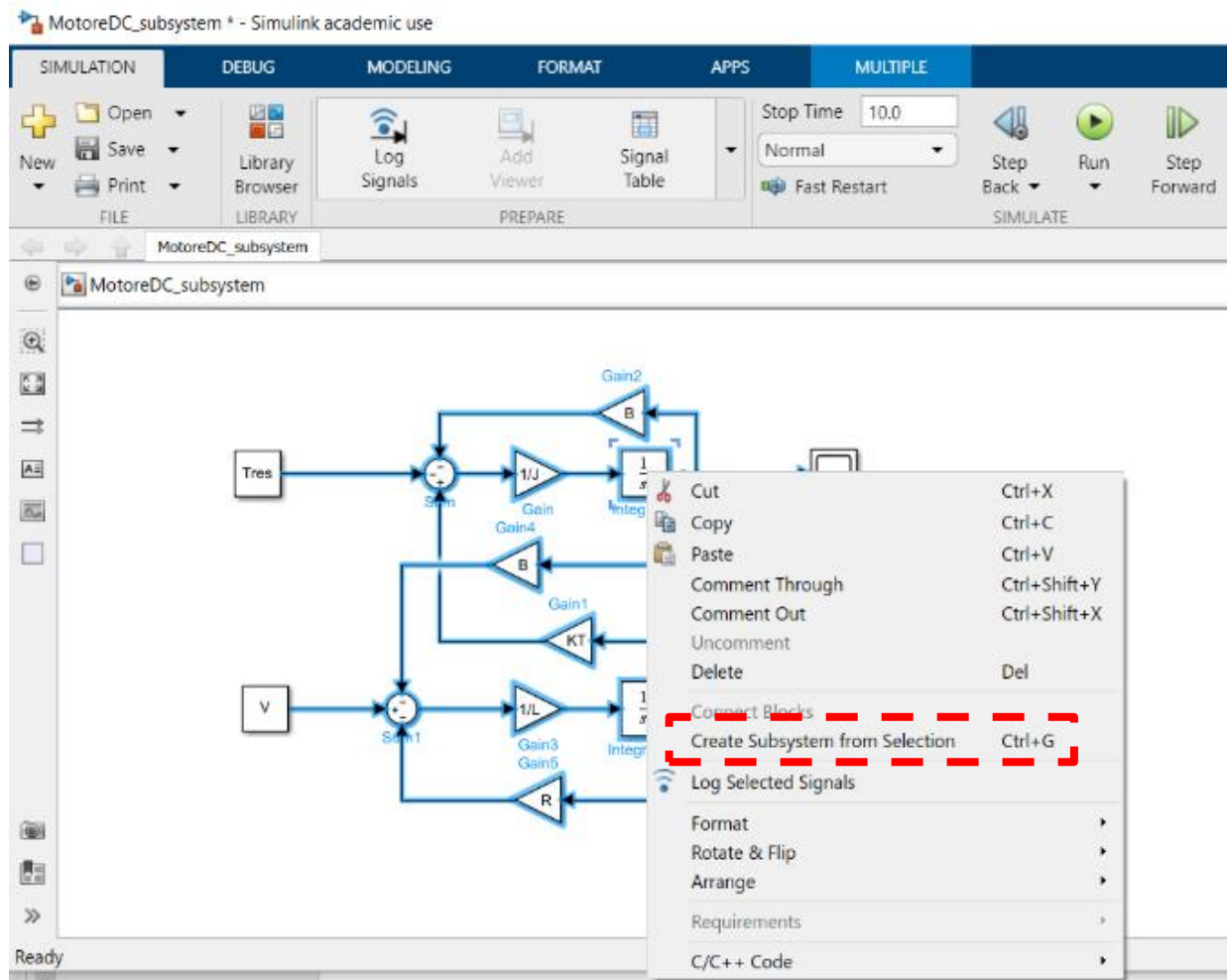


Porzione di un modello di simulazione di un impianto, con tre sottosistemi identici in parallelo che rappresentano 3 valvole motorizzate con le stesse caratteristiche.

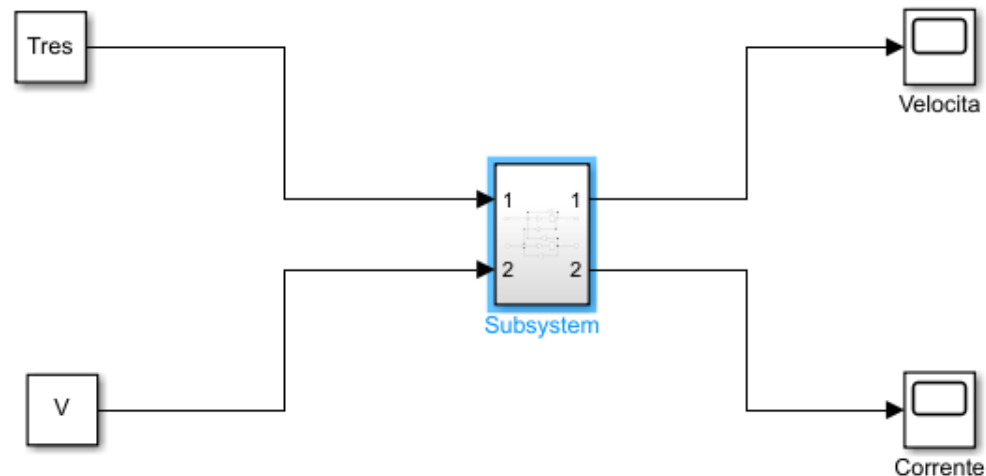
Bisogna tracciare sopra il modello un rettangolo come quello in figura, nel quale «entrino» i segnali di ingresso e dal quale «escano» i segnali di uscita del sottosistema che si sta creando. Ciò può richiedere, in taluni casi, di «ricollocare» i blocchi del modello spostandoli in modo che sia possibile tracciare il rettangolo citato.



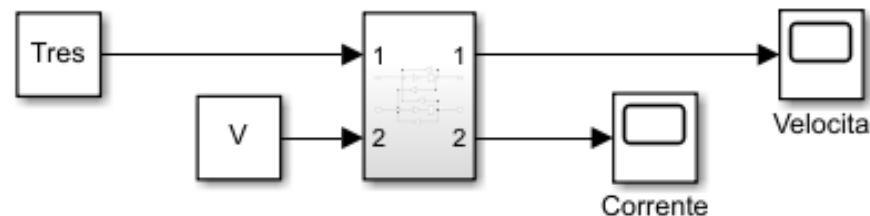
Individuare con il mouse un'area analoga a quella individuata dal rettangolo rosso della slide precedente, successivamente puntare il mouse su un qualunque blocco selezionato interno all'area, premere il tasto destro e selezionare "Create Subsystem from Selection" (in alternativa premere Ctrl+G)



Il risultato è il seguente

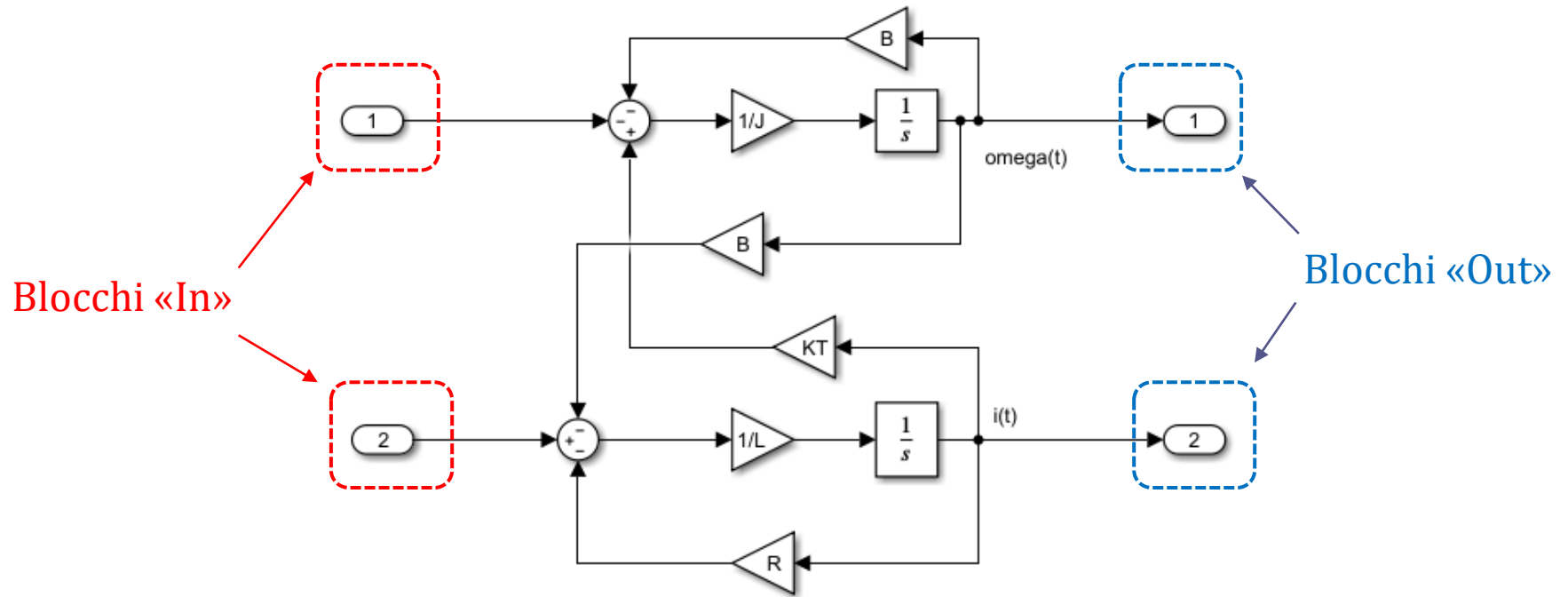


Possiamo riordinare e ridimensionare i blocchi per creare una struttura più ordinata



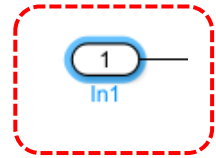
Ora vogliamo che nei terminali di ingresso e uscita del subsystem compaiano dei nomi rappresentativi dei segnali che entrano ed escono dal blocco.

Per accedere al contenuto del Subsystem bisogna fare doppio click su di esso

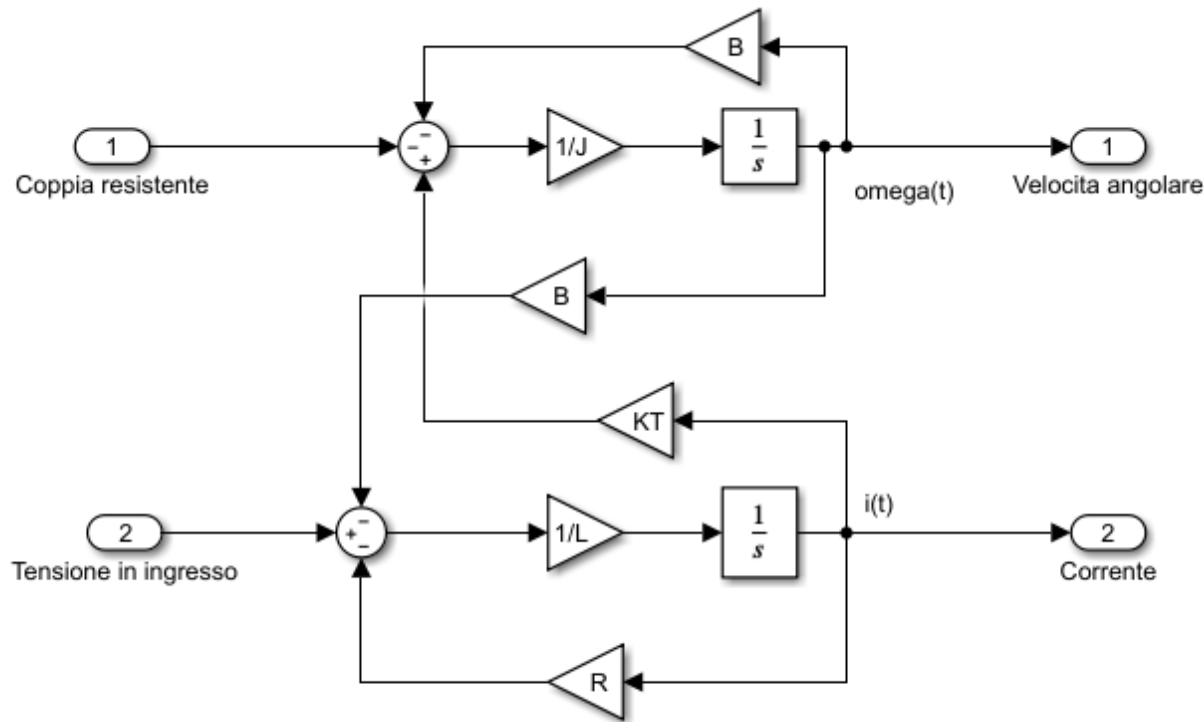


All'interno di un subsystem ciascun segnale di ingresso è rappresentato da un blocco In, mentre ciascun segnale di uscita è collegato ad un blocco Out. Se si aggiungono, o rimuovono, dei blocchi In o Out varierà in maniera corrispondente il numero di terminali di ingresso e uscita del subsystem.

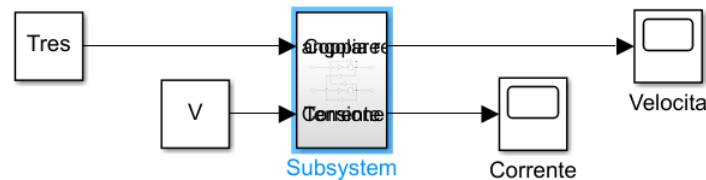
Per assegnare ad un terminale di ingresso o uscita una label si deve selezionare il corrispondente blocco In e Out, e si deve sovrascrivere il nome del blocchetto che compare sotto lo stesso.



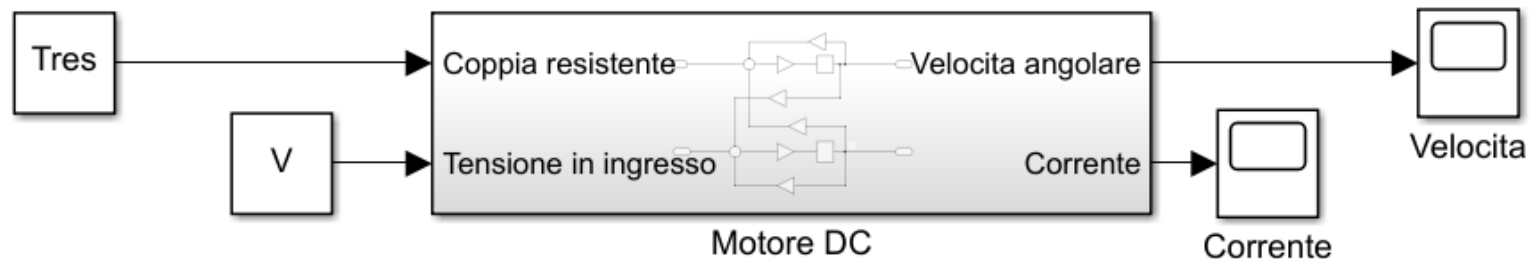
Svolgendo questa operazione per tutti i blocchi In e Out si giunge al seguente schema



L'aspetto del sottosistema risulta modificato.



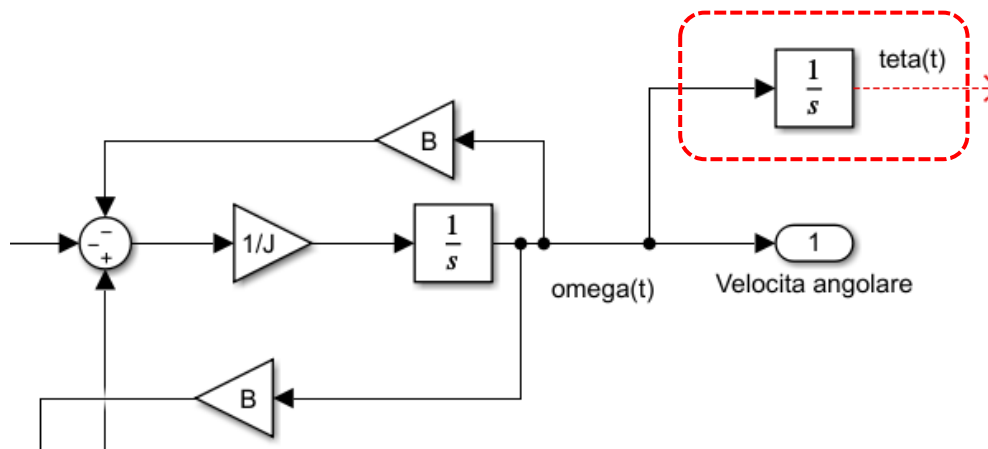
Allargando il sottosistema si visualizzano correttamente le labels. E' possibile attribuire un nome al Subsystem selezionandolo e sovrascrivendo il nome del blocchetto che compare sotto lo stesso.



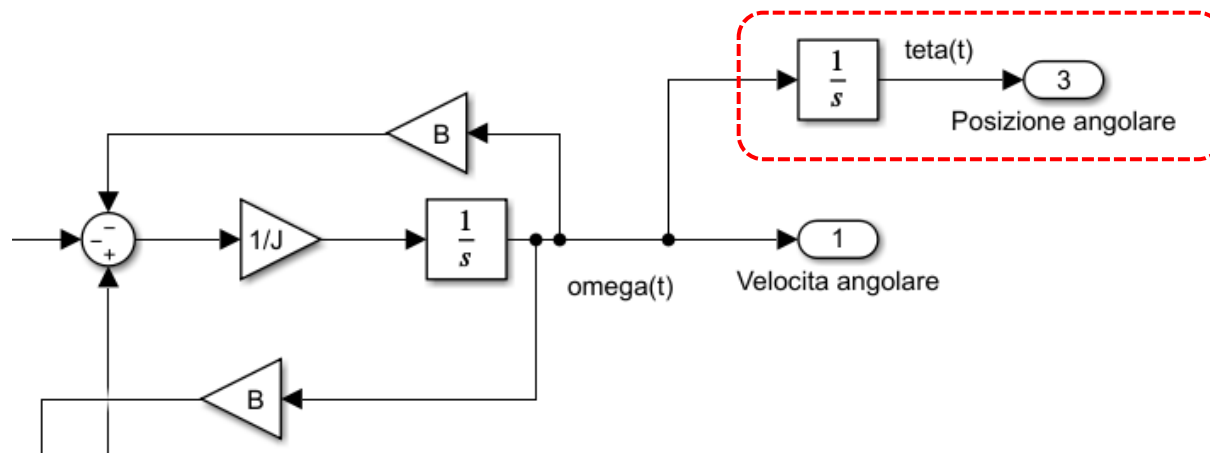
In trasparenza è mostrato il contenuto del sottosistema. Per disabilitare tale visualizzazione cliccare con il tasto destro sul sottosistema e successivamente disselezionare la voce «Content Preview» dal menù «Format».

Modifichiamo il subsystem in modo che venga restituita in uscita anche la **posizione angolare** del motore.

Dobbiamo preliminarmente rendere accessibile la posizione angolare integrando la velocità (si inserisca quindi un nuovo blocco integratore che riceve in ingresso la velocità)



Ora importiamo nello schema dalla libreria «Commonly Used Blocks» una istanza del blocco elementare «Out», e colleghiamone il segnale della posizione angolare al terminale di ingresso del blocco Out in modo che tale segnale venga mandato all'esterno del sottosistema attraverso una nuova porta di uscita. Associamo anche una Label al blocco Out in modo che tale label sia inserita nella porta di uscita



Ecco come si presenta il blocco Subsystem dopo avere apportato tale modifica:

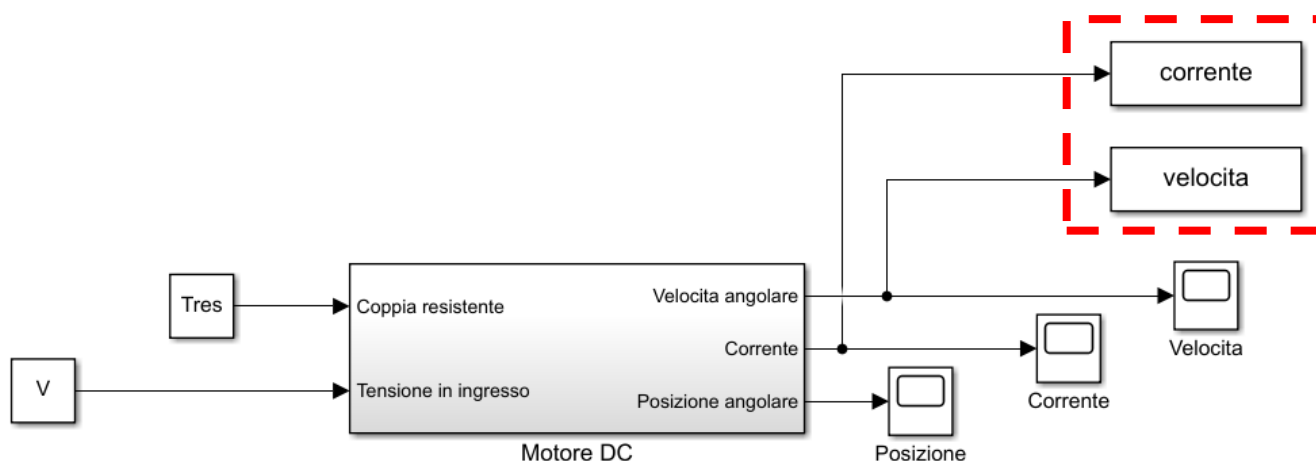


File: MotoreDC_subsystemPosizione.slx

Esportazione verso il workspace di Matlab di dati generati da un modello Simulink

Mostriamo come esportare dati generati da un modello Simulink verso il workspace di Matlab. Ciò ne consente l'elaborazione numerica o, ad esempio, la creazione con il comando **plot** di grafici di qualità corredati da etichette esplicative

L'esportazione di dati da Simulink verso il workspace di Matlab avviene mediante il blocco elementare "To workspace" che si trova nella libreria Sinks. Importiamone 2 istanze onde esportare nel workspace i dati associati alla velocità angolare ed alla corrente di fase.



Blocchi
"To workspace"
collegati ai segnali
da esportare nel
workspace

Files: MotoreDC_subsystem2Workspace.slx / MotoreDC_dati.m

Finestra di parametrizzazione del primo blocco “To Workspace”

Nome che sarà attribuito alla variabile esportata nel workspace

Possibilità di sottocampionare i dati esportati nel workspace attraverso il parametro «Decimation»

Formato di esportazione: modalita di default “**Timeseries**”

Block Parameters: To Workspace

To Workspace

Write input to specified timeseries, array, or structure in a workspace. For menu-based simulation, data is written in the MATLAB base workspace. Data is not available until the simulation is stopped or paused.

To log a bus signal, use "Timeseries" save format.

Parameters

Variable name:

corrente

Limit data points to last:

inf

Decimation:

1

Save format: Timeseries

☒ Log fixed-point data as a TI object

Sample time (-1 for inherited):

-1

OK Cancel Help Apply

Eseguire la simulazione, e verificare quali nuovi variabili compaiano nel workspace dopo che la simulazione è terminata

Al termine della simulazione sono presenti nel workspace le due variabili «corrente» e «velocita» aventi come formato «Timeseries»

Il formato «Timeseries» contiene al proprio interno per ciascun segnale sia gli istanti di campionamento che i valori del segnale.

Variables - velocita

velocita

1x1 double timeseries

Time series name:

Time	Data:1
0	0
1.0000e-03	0.0011
0.0020	0.0037
0.0030	0.0069
0.0040	0.0105
0.0050	0.0141
0.0060	0.0178
0.0070	0.0215
0.0080	0.0252
0.0090	0.0289

Workspace

Name	Value	Size
B	0.8000	1x1
corrente	1x1 double timeseries	1x1
i0	0	1x1
J	1	1x1
KT	0.3000	1x1
KV	0.3000	1x1
L	1.0000e-03	1x1
omega0	0	1x1
R	0.8000	1x1
tout	10001x1 double	10001x1
Tres	0	1x1
V	10	1x1
velocita	1x1 double timeseries	1x1

In una variabile di tipo “Timeseries” avente nome generico “var” si accede al **vettore degli istanti di campionamento** ed al **vettore dei valori del segnale** con le sintassi:

var.Time
var.Data

Potremo quindi accedere ai dati di velocita e corrente con le sintassi

corrente.Time
corrente.Data

velocita.Time
velocita.Data

Si noti che tali vettori sono esattamente gli argomenti da passare alla funzione plot per costruire il relativo grafico.



Una impostazione di default dei modelli Simulink fa si che quando si esportano variabili verso il workspace queste vengano «aggregate» in una unica variabile (della quale si puo scegliere il nome) anzichè essere variabili distinte («corrente» e «velocita») come nell'esempio appena trattato.

Tale impostazione si chiama «**Single Simulation Output**», e si trova nel menù «Data Import/Export» della finestra «Configuration Parameters» del modello (la medesima finestra all'interno della quale si settano i parametri del solutore numerico)

Configuration Parameters: MotoreDC/Configuration (Active)

Search

- Solver
- Data Import/Export
- Math and Data Types
- Diagnostics
- Hardware Implementation
- Model Referencing
- Simulation Target

Load from workspace

☐ Input: [t, u]

☐ Initial state: xInitial

Save to workspace or file

☒ Time: tout

☐ States: xout Format: Dataset

☒ Output: yout

☐ Final states: xFinal ☐ Save final operating point

☒ Signal logging: logout

☒ Data stores: dsmout

☐ Log Dataset data to file: out.mat

☐ Single simulation output: out Logging intervals: [-inf, inf]

Simulation Data Inspector

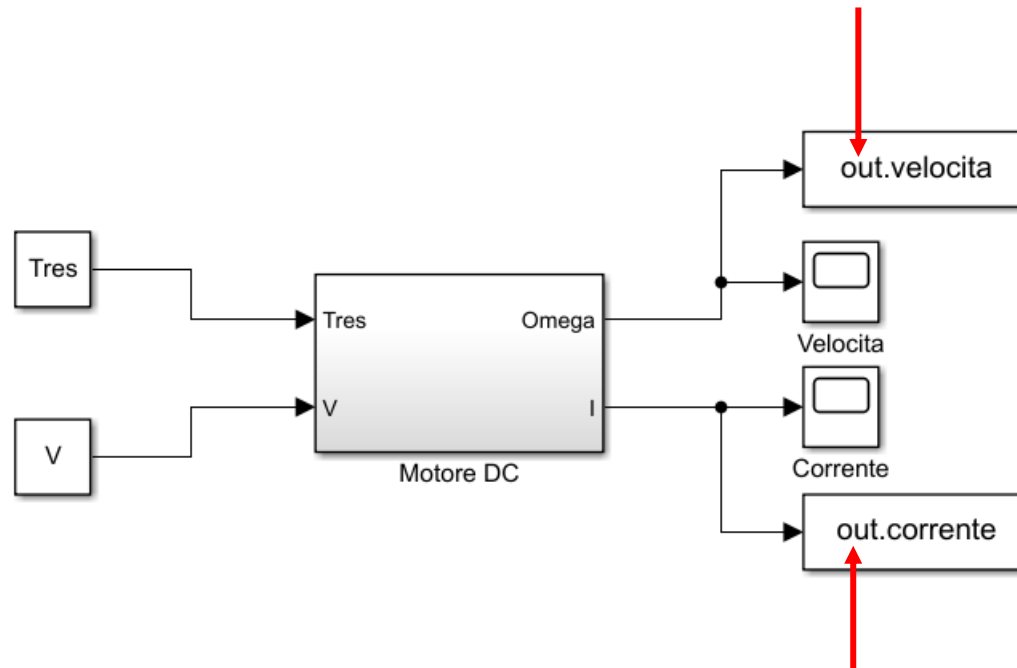
☐ Record logged workspace data in Simulation Data Inspector

► Additional parameters

OK Cancel Help Apply

Se l'impostazione è attiva, viene esportata nel workspace una unica variabile, chiamata in questo caso «out», e le istruzioni di creazione grafici descritte precedentemente non possono essere più impiegate.

Se l'impostazione è attiva, l'aspetto dei blocchi «To workspace» è modificato come mostrato nella Figura seguente

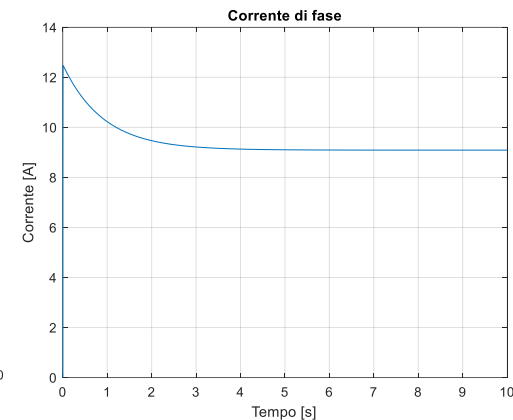
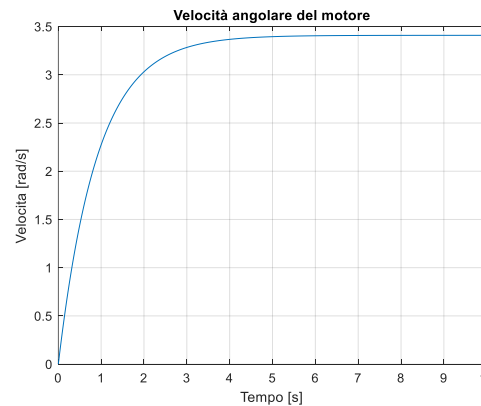


Si presti quindi attenzione, se il contenuto dei blocchi «To workspace» è come quello riportato nella precedente Figura, a disabilitare l'impostazione menzionata.

Dopo avere eseguito la simulazione, si può eseguire il seguente script di creazione dei grafici

```
figure(1)
plot(velocita.Time,velocita.Data),
grid,
title('Velocità angolare del motore')
ylabel('Velocita [rad/s]')
xlabel('Tempo [s]')
```

```
figure(2)
plot(corrente.Time,corrente.Data),
grid,
title('Corrente di fase')
ylabel('Corrente [A]')
xlabel('Tempo [s]')
```



NB Se si omettono le istruzioni `figure(1)` e `figure(2)`, la seconda chiamata alla funzione `plot` sovrascrive il grafico della velocità, che pertanto si «perde»

Analizziamo istruzione per istruzione il precedente codice.

```
figure(1)
```

```
plot(velocita.Time,velocita.Data),
```

```
grid,
```

```
title('Velocità angolare del motore')
```

```
ylabel('Velocita [rad/s]')
```

```
xlabel('Tempo [s]')
```

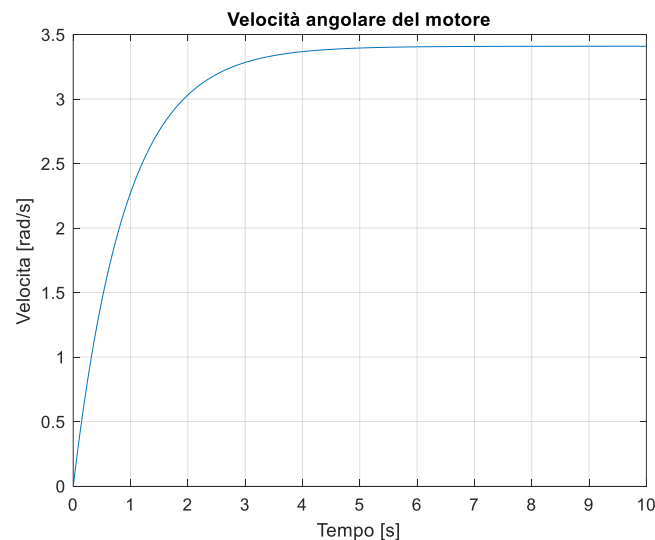
Crea il grafico. Si passano come argomenti di ingresso alla funzione plot i due vettori che contengono gli istanti di campionamento ed i valori del segnale.

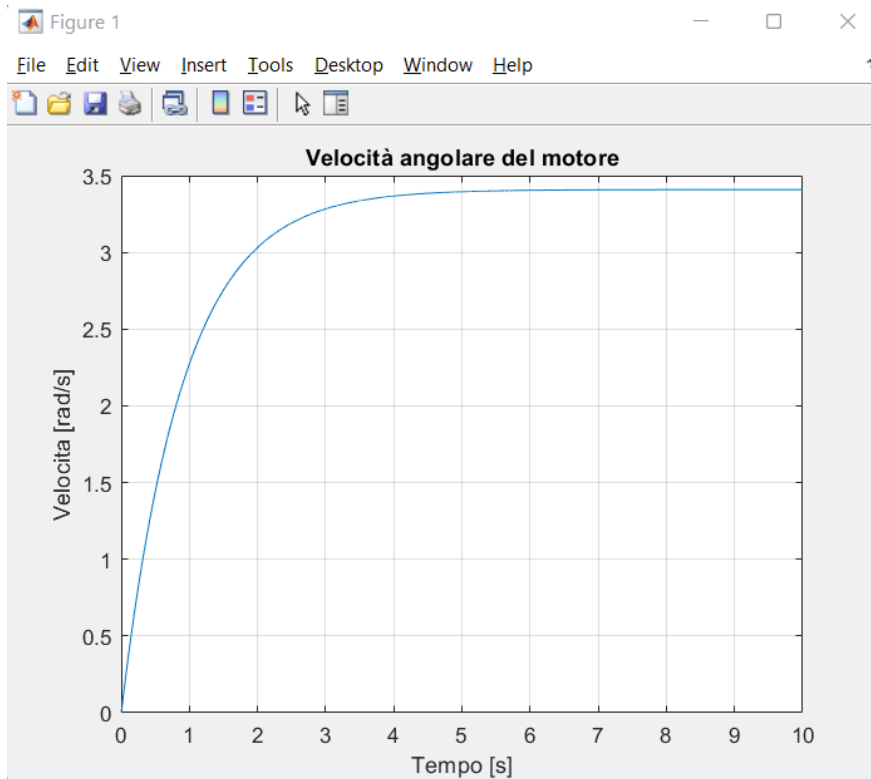
Aggiunge una griglia di sfondo

Aggiunge un titolo

Aggiunge un'etichetta sull'asse delle ordinate

Aggiunge un'etichetta sull'asse delle ascisse





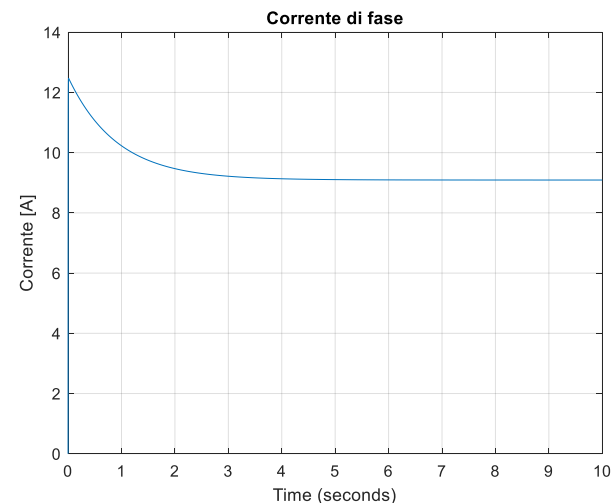
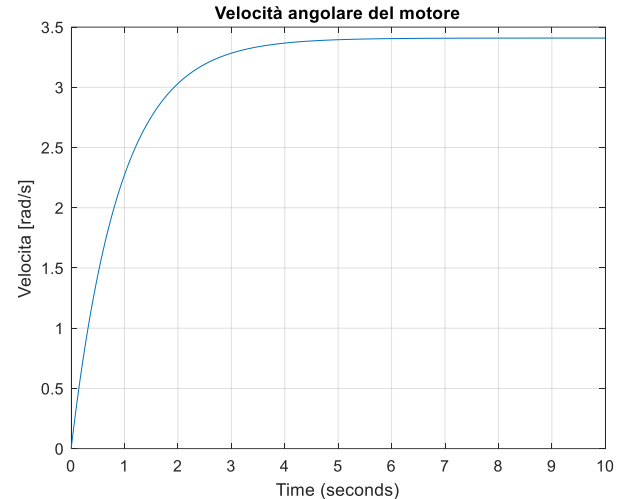
Il grafico può essere esportato in un formato grafico (es jpeg) mediante File->Save as

Il grafico può anche essere direttamente copiato e incollato all'interno di un documento (Edit->Copy Figure)

Per la creazione di **grafici singoli** a partire da variabili di tipo Timeseries, si può utilizzare una **sintassi semplificata** in cui viene passata alla funzione plot la sola variabile di tipo Timeseries.

```
figure(1)
plot(velocita),
grid,
title('Velocità angolare del motore')
ylabel('Velocita [rad/s]')
```

```
figure(2)
plot(corrente),
grid,
title('Corrente di fase')
ylabel('Corrente [A]')
```

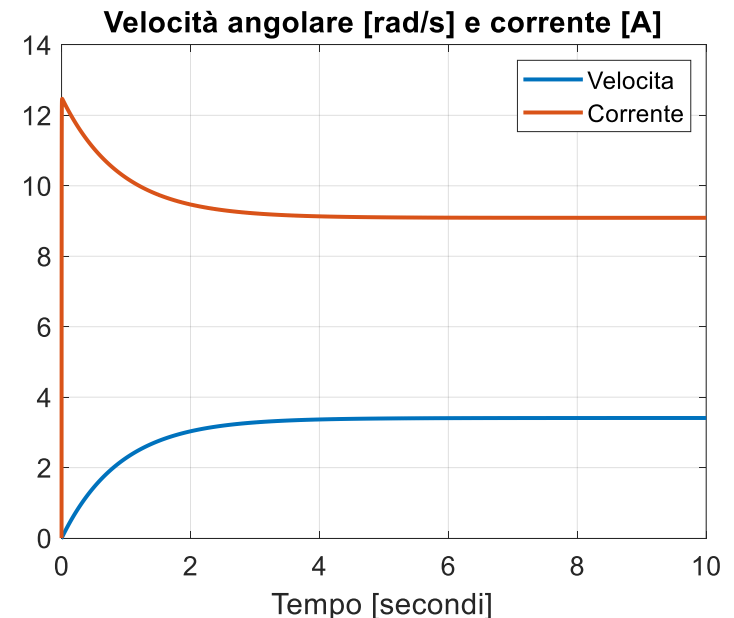


Per la creazione di **grafici con curve sovrapposte** a partire da variabili di tipo Timeseries, si deve invece necessariamente accedere ai vettori con la sintassi `.Time` e `.Data`.

Il seguente codice crea un grafico in cui la velocità e la corrente risultano sovrapposte e inoltre presente una legenda che chiarisce il significato delle curve e consente di distinguerle.

```
figure(3)
plot(velocita.Time,velocita.Data,corrente.Time,corrente.Data,'LineWidth',2),
grid,
title('Velocità angolare [rad/s] e corrente [A]','FontSize',14)
xlabel('Tempo [secondi]','FontSize',14)
set(gca,'FontSize',14)
legend('Velocita','Corrente')
```

Commentiamo queste istruzioni
nella slide successiva



```
figure(3)
plot(velocita.Time,velocita.Data,corrente.Time,corrente.Data,'LineWidth',2),
```

Si passano in ingresso alla funzione `plot` due coppie di vettori, nel medesimo ordine con cui si sarebbero inseriti in caso di grafico singolo. Gli ultimi due argomenti di ingresso impostano lo **spessore delle linee**.

```
grid,

title('Velocità angolare [rad/s] e corrente [A]','FontSize',14)
```

```
xlabel('Tempo [secondi]','FontSize',14)
```

Gli ultimi due argomenti di ingresso alle funzioni `title` e `xlabel` impostano lo la **dimensione del testo**, che di default è eccessivamente piccolo.

```
set(gca,'FontSize',14)
```

Imposta la **dimensione delle cifre negli assi coordinati**. Anche questo parametro ha un valore di default piccolo

```
legend('Velocita','Corrente')
```

La funzione `legend` riceve come argomenti di ingresso le due stringhe che compariranno nella legenda. Ovviamente, l'ordine con cui si passano le stringhe corrisponde all'ordine con cui si sono passati i vettori in ingresso alla funzione `plot`

Gestione di modelli Simulink mediante script (funzione `sim`)

Vogliamo imparare ad **aprire** ed **eseguire** i file Simulink per mezzo di uno script Matlab che impartisca i relativi comandi.

Realizziamo uno script all'interno del quale si operi:

1. la parametrizzazione del modello
2. l'avvio della simulazione, e
3. la successiva creazione di grafici.

Sviluppiamo questo esempio con riferimento al file

`MotoreDC_subsystem2Workspace.slx`

Impiegheremo all'interno di tale Script le funzioni Matlab **`open_system`** e **`sim`** che servono rispettivamente per aprire il file Simulink e per mandarlo in esecuzione.

```
%% Parametrizzazione del modello
```

Inizio di una nuova «Section»

```
clear all,clc
```

```
% Parametri modello motore DC
```

```
% Parametri elettromeccanici
```

```
R=0.8;      % resistenza
```

```
L=1e-3;     % induttanza
```

```
KT=0.3;     % costante di coppia
```

```
KV=0.3;     % costante di forza c.e.m.
```

```
J=1;        % momento di inerzia
```

```
B=0.8;      % coefficiente di attrito viscoso
```

```
% Ingressi
```

```
V=10;       % tensione di alimentazione
```

```
Tres=0;     % coppia resistente
```

```
% Condizioni iniziali
```

```
omega0=0;   %condizione iniziale della velocita
```

```
i0=0;       %condizione iniziale della corrente
```

```
%% Apertura ed avvio del modello
```

Inizio di una nuova «Section»

```
open_system('MotoreDC_subsystem2Workspace')
```

Apri il modello Simulink

```
sim('MotoreDC_subsystem2Workspace')
```

Esegui il modello

(continua)

(continua)

%% Creazione grafici

Inizio di una nuova «Section»

```
figure(1)
plot(velocita),
grid,
title('Velocità angolare del motore')
ylabel('Velocita [rad/s]')
```

```
figure(2)
plot(corrente),
grid,
title('Corrente di fase')
ylabel('Corrente [A]')
```

```
figure(3)
plot(velocita.Time,velocita.Data,corrente.Time,corrente.Data,'LineWidth',2),grid,
title('Velocità angolare [rad/s] e corrente [A]','FontSize',14)
xlabel('Tempo [secondi]','FontSize',14)
set(gca,'FontSize',14)
legend('Velocita','Corrente')
```

Files: MotoreDC_ScriptGestione.m

Lo script riportato nelle due slides precedenti implementa la completa gestione del modello Simulink senza che l'utente debba operare direttamente su di esso.

Lo script salva dapprima nel workspace le variabili simboliche utilizzate dal modello Simulink, quindi apre e manda in run il modello, ed in ultimo crea dei grafici utilizzando le variabili di tipo Timeseries che vengono esportate nel workspace dal modello Simulink al termine della esecuzione.

Si noti come nello Script sono state inserite alcune righe di codice che hanno **due caratteri percentuali consecutivi a inizio riga**.

Tali righe di codice inducono la suddivisione dello script in varie **Sezioni** (Sections)

```
%% Parametrizzazione del modello
```

Prima Section

```
%% Apertura ed avvio del modello
```

Seconda Section

```
%% Creazione grafici
```

Terza Section

Lo script contiene **tre distinte sezioni**

L'utilità delle Sections è duplice. Non solo individua e separa parti dello Script in cui vengono eseguite operazioni di natura differente ma consente facilmente di eseguire la porzione di codice contenuta in una Sezione senza dover necessariamente eseguire tutto lo script.

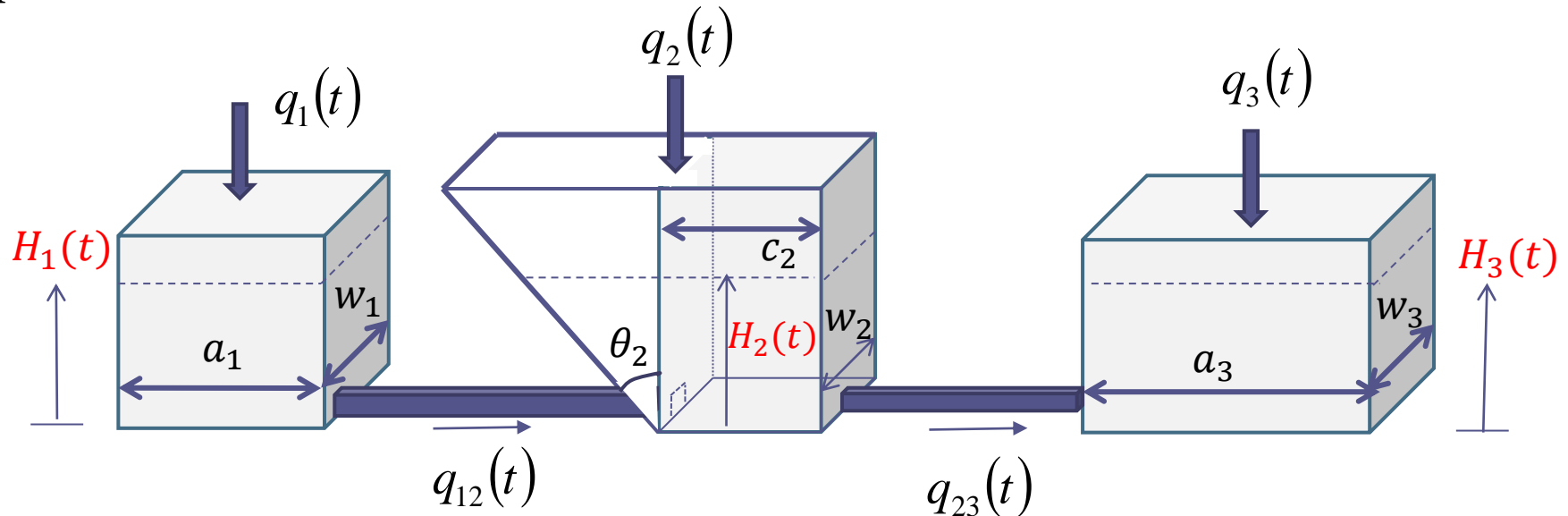
Per fare ciò si deve rendere «attiva» una particolare Sezione cliccando su di essa (la Section attiva è evidenziata con un colore particolare, come mostrato nella figura sottostante) e successivamente premere Ctrl+Invio. In tal modo si eseguono unicamente le istruzioni Matlab contenute nella sezione attiva.

```
omega0=0; %condizione iniziale della velocita
i0=0;      %condizione iniziale della corrente
%% Apertura ed avvio del modello
open_system('MotoreDC_subsystem2Workspace')
sim('MotoreDC_subsystem2Workspace')
%% Creazione grafici
figure(1)
plot(velocita),
grid,
title('Velocità angolare del motore')
ylabel('Velocita [rad/s]')
```

Ad esempio, una volta che il modello sia stato aperto ed eseguito (operazione che può richiedere un tempo anche molto lungo in funzione della complessità del modello) si andrà unicamente a lavorare sulla sezione di creazione dei grafici, modificandola e rieseguendola in maniera selettiva finché non si sia ottenuto l'output desiderato.

Serbatoi idraulici interconnessi

Modelliamo un sistema idraulico composto da tre serbatoi a pelo libero interconnessi da due pipeline



$H_1(t), H_2(t), H_3(t)$

Livelli del liquido nei 3 serbatoi

$q_1(t), q_2(t), q_3(t)$

Portate volumetriche esterne in ingresso nei 3 serbatoi

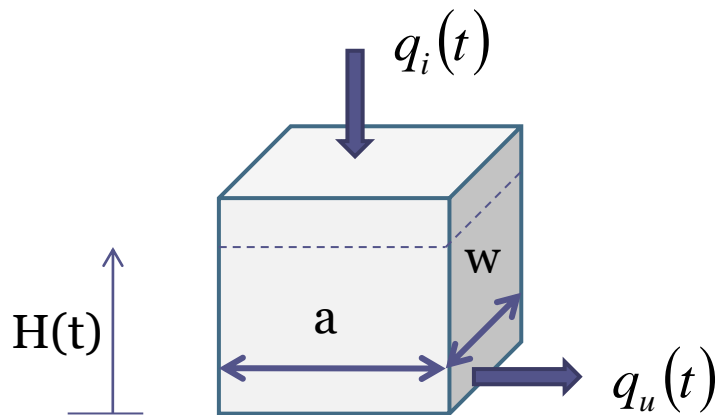
$q_{12}(t)$ ($q_{23}(t)$)

Portata volumetrica che fluisce fra i serbatoi 1 e 2 (**2 e 3**)

$a_1, w_1, c_2, w_2, \theta_2, a_3, w_3$

Dimensioni geometriche dei serbatoi

Serbatoio a sezione costante rettangolare



$V(t)$ = volume di liquido contenuto nel serbatoio (in m^3)

$$V(t) = awH(t)$$

$H(t)$ = livello del liquido nel serbatoio (in m)

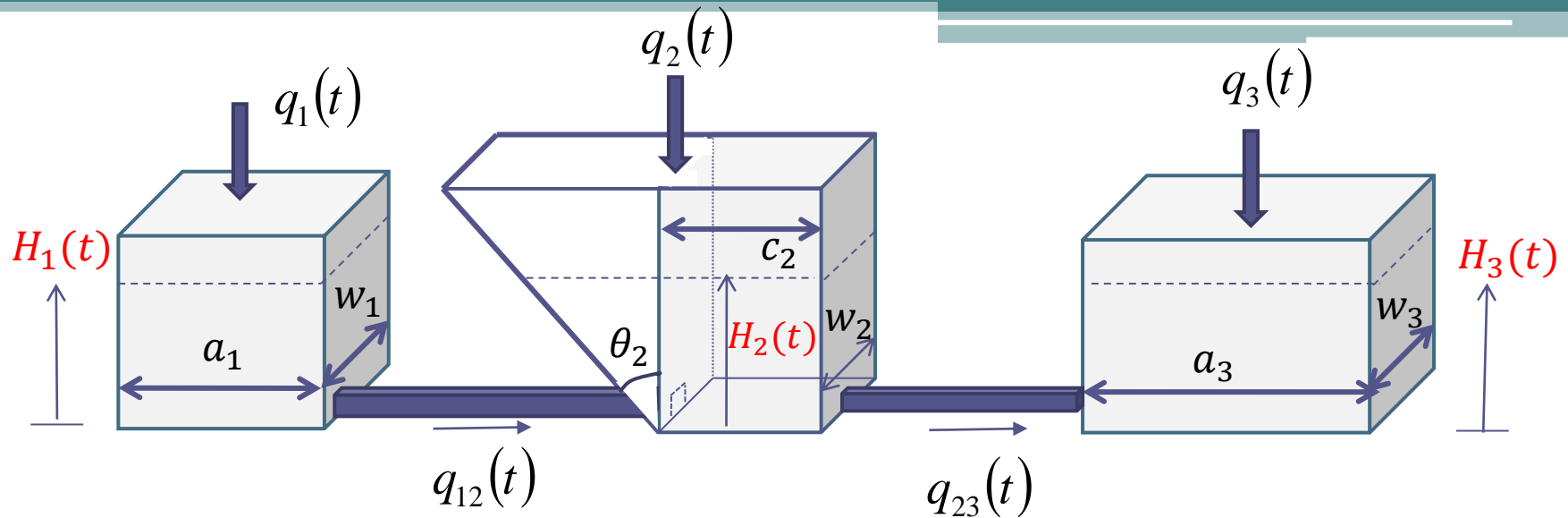
$q_i(t)$ $q_u(t)$ portate **volumetriche** in ingresso ed in uscita dal serbatoio (in m^3/sec)

Conservazione
della massa

$$\dot{V}(t) = q_i(t) - q_u(t)$$

$$\dot{V}(t) = aw\dot{H}(t) = q_i(t) - q_u(t)$$

$$\dot{H}(t) = \frac{1}{aw} q_i(t) - \frac{1}{aw} q_u(t)$$



$$\dot{H}_1(t) = \frac{1}{a_1 w_1} [q_1(t) - q_{12}(t)]$$

$$\dot{H}_2(t) = \frac{1}{\tan(\theta_2) w_2 H_2(t) + c_2 w_2} [q_2(t) + q_{12}(t) - q_{23}(t)]$$

$$\dot{H}_3(t) = \frac{1}{a_3 w_3} [q_3(t) + q_{23}(t)]$$

**Modello in
forma esplicita**

**Sistema di 3 eq.
differenziali non
lineari, accoppiate**

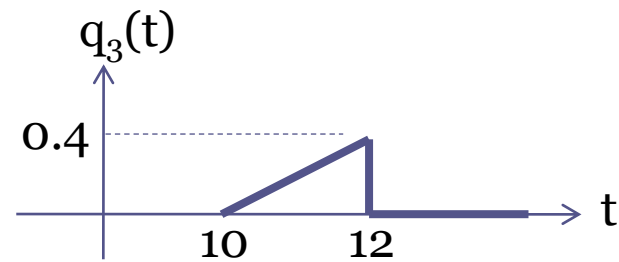
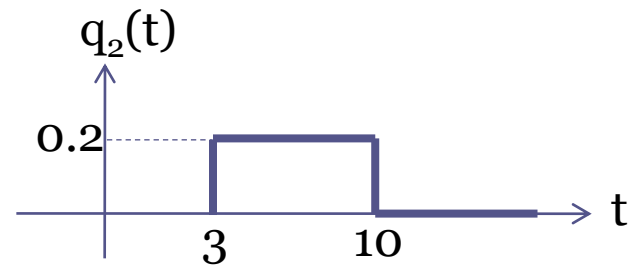
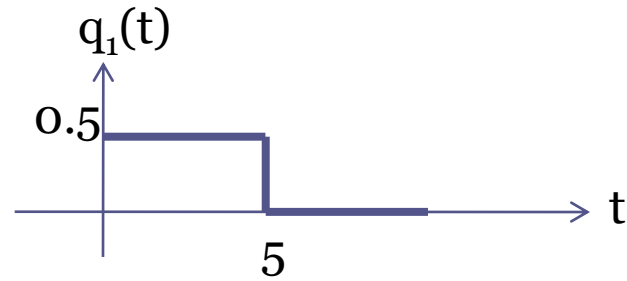
$$q_{12}(t) = C_{12} \sqrt{|H_1(t) - H_2(t)|} \cdot \text{sign}[H_1(t) - H_2(t)]$$

$$q_{23}(t) = C_{23} \sqrt{|H_2(t) - H_3(t)|} \cdot \text{sign}[H_2(t) - H_3(t)]$$

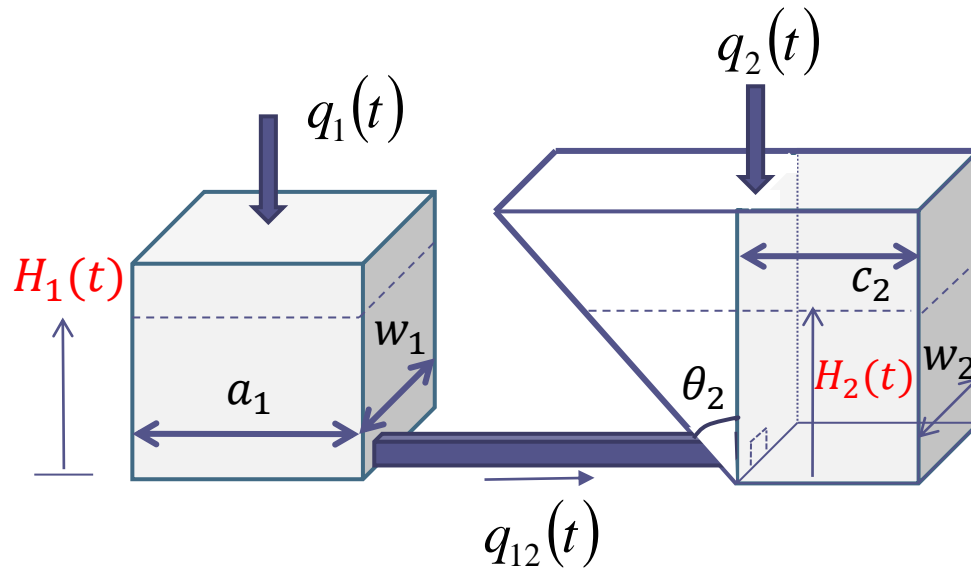
C_{12}, C_{23}

Coefficienti di efflusso

Portate volumetriche in ingresso



Versione semplificata con due serbatoi



**Modello in
forma esplicita**

**Sistema di 2 eq.
differenziali non
lineari, accoppiate**

$$\dot{H}_1(t) = \frac{1}{a_1 w_1} [q_1(t) - q_{12}(t)]$$

$$\dot{H}_2(t) = \frac{1}{\tan(\theta_2) w_2 H_2(t) + c_2 w_2} [q_2(t) + q_{12}(t)]$$

$$q_{12}(t) = C_{12} \sqrt{|H_1(t) - H_2(t)|} \cdot \text{sign}[H_1(t) - H_2(t)]$$

C_{12} Coefficiente di efflusso

Anche se le seguenti slides mostrano una guida passo passo alla modellazione del sistema a 3 serbatoi, procediamo per semplicità a realizzare il modello del sistema semplificato a 2 serbatoi, trascurando sia la portata $q_{23}(t)$ che la dinamica del livello del terzo serbatoio.

File: Serbatoi_dati.m

```
%% Parametrizzazione del modello
% Parametri serbatoio #1
a1=5;    %larghezza serb. S1 [m]
w1=1;    %profondita serb. S1 [m]

% Parametri serbatoio #2
c2=3;
w2=1;
teta2=pi/4;

% Parametri serbatoio #3
a3=4;
w3=2;

% Coefficienti di efflusso
C12=0.5;
C23=0.6;

% Condizioni iniziali
H10=1;
H20=2;
H30=3;
```

**Eseguiamo lo script in
modo da creare nel
workspace le
corrispondenti variabili**

Apriamo un modello Simulink in bianco, e configuriamo il solutore come segue (oppure si apra il template creato in precedenza che già possiede tali impostazioni)

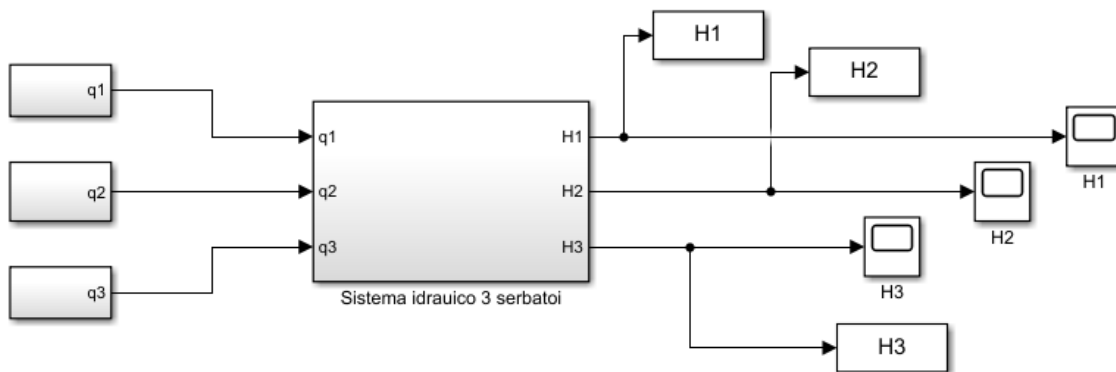
Type: fixed step;

Solver: auto (Automatic Solver selection);

Fixed step size: 0.01 s

Desideriamo strutturare il modello mediante un Subsystem che riceva in ingresso le portate $q_1(t)$, $q_2(t)$, $q_3(t)$ e fornisca in uscita i 3 livelli $H_1(t)$, $H_2(t)$, $H_3(t)$

Obiettivo:



Anche le 3 portate $q_1(t)$, $q_2(t)$, $q_3(t)$ desideriamo che vengano generate all'interno di appositi sottosistemi.

Descriviamo una procedura differente per realizzare un modello contenente dei Subsystems. Importiamo nella pagina di lavoro un blocco «Subsystem» dalla Libreria dei «Commonly Used Blocks»

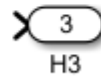
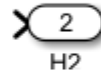
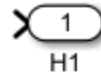
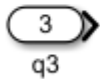
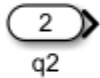


(cliccare il tasto destro del mouse sul Subsystem e disselezionare la voce «Content preview» dal menu format)

Si tratta di un sottosistema già pronto, che ha di default un ingresso ed un'uscita.

Trasformiamolo in un subsystem con tre ingressi e tre uscite **incrementando il numero di blocchi «In» e «Out» presenti all'interno del Subsystem**. Inseriamo anche opportune etichette, come già fatto in precedenza.

Contenuto del Subsystem



Aspetto del Subsystem



Abbiamo creato lo «scheletro» del Subsystem. Ora bisogna completarne il contenuto in modo che questo implementi il sistema di equazioni che definiscono il modello del sistema idraulico a 3 serbatoi.

Importiamo all'interno del Subsystem **tre blocchi integratori**, e disponiamoli come in figura (si impostino anche le relative condizioni iniziali mediante le variabili H10, H20 ed H30). Faremo in modo che alle porte di uscita dei tre integratori vengano generati i tre livelli $H_1(t)$, $H_2(t)$, $H_3(t)$

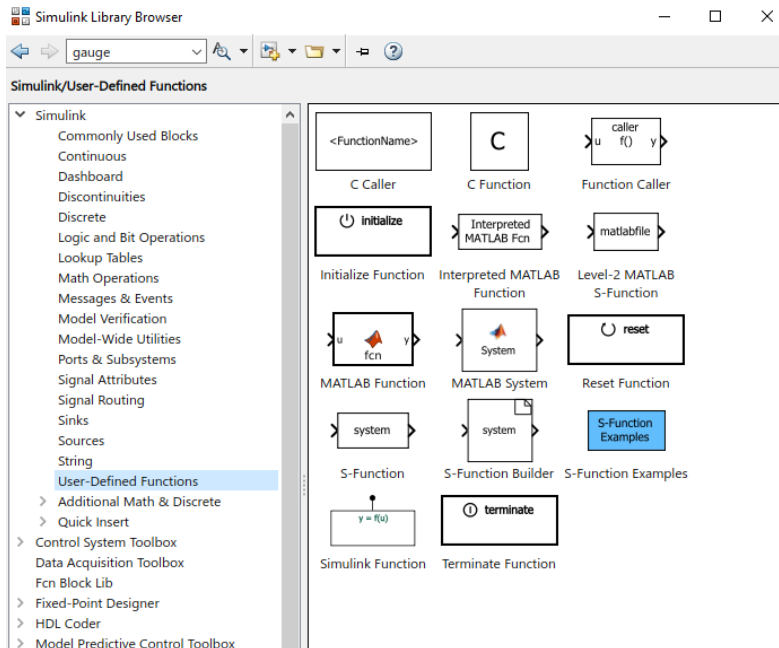


Iniziamo con il costruire i segnali $q_{12}(t)$ e $q_{23}(t)$

$$q_{12}(t) = C_{12} \sqrt{|H_1(t) - H_2(t)|} \cdot \text{sign}[H_1(t) - H_2(t)]$$

$$q_{23}(t) = C_{23} \sqrt{|H_2(t) - H_3(t)|} \cdot \text{sign}[H_2(t) - H_3(t)]$$

Faremo uso del blocco « MATLAB Function » (libreria: User-defined functions)



Consente di scrivere ed eseguire un “function file” Matlab direttamente all’interno di un modello Simulink.

Default:



Se si fa doppio click sul blocco si apre l’editor testuale dove poter scrivere il corpo della funzione

```

MATLAB Function
1  function y = fcn(u)
2
3  y = u;
4

```

Il blocco si programma secondo le medesime procedure e sintassi per la redazione di un Function File (v. la sezione «Function files e anonymous functions» nella parte conclusiva di queste slides). Può essere configurato con un **numero arbitrario di porte di ingresso e uscita**, in relazione al numero di ingressi e uscite della function. Ciascun ingresso o uscita può essere un segnale scalare, vettoriale o matriciale.

All'interno di un blocco «MATLAB function» si possono eseguire complesse operazioni matriciali che rendono particolarmente utile questo blocco elementare nell'ambito di modelli di particolare complessità.

N.B. All'interno di un blocco MATLAB function **non sono visibili le variabili del workspace**, neanche se queste vengono definite come globali.

Ritorniamo al problema della costruzione dei segnali $q_{12}(t)$ e $q_{23}(t)$ mediante uno (o più) blocchi MATLAB function.

Scegliamo di utilizzarne solo uno, in modo che **un unico blocco calcoli e restituisca all'esterno sia $q_{12}(t)$ che $q_{23}(t)$**

$$q_{12}(t) = C_{12} \sqrt{|H_1(t) - H_2(t)|} \cdot \text{sign}[H_1(t) - H_2(t)]$$

$$q_{23}(t) = C_{23} \sqrt{|H_2(t) - H_3(t)|} \cdot \text{sign}[H_2(t) - H_3(t)]$$

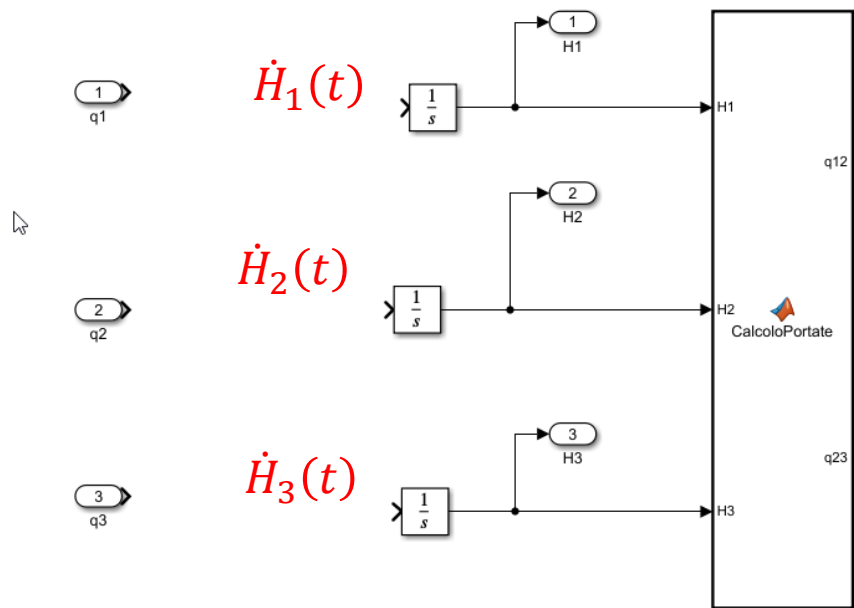
Il blocco riceverà pertanto in ingresso i segnali $H_1(t)$, $H_2(t)$ e $H_3(t)$

Modifichiamo la prima riga come segue:

```
function [q12,q23] = CalcoloPortate(H1,H2,H3)
```

```
q12=
```

```
q23=
```



Poiché le variabili presenti nel workspace non sono visibili dall'interno di un blocco MATLAB Function, dovremo **definire i coefficienti di efflusso internamente al corpo della funzione**. Inseriamo anche contestualmente le espressioni per le due uscite q12 e q23 del blocco

```
function [q12,q23] = CalcoloPortate(H1,H2,H3)
```

```
% Coefficienti di efflusso
```

```
C12=0.5;
```

```
C23=0.6;
```

```
q12=C12*sqrt(abs(H1-H2))*sign(H1-H2);
```

```
q23=C23*sqrt(abs(H2-H3))*sign(H2-H3);
```

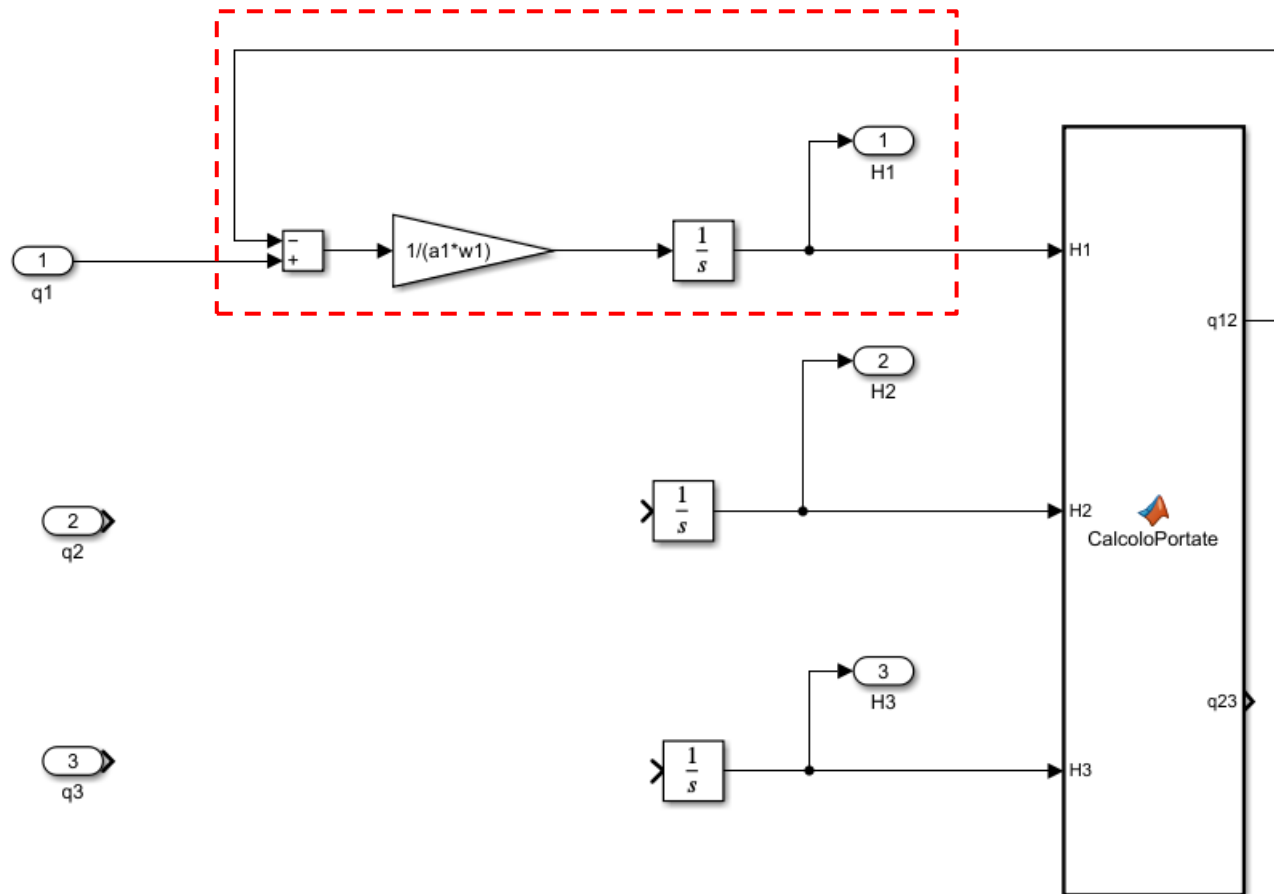
A questo punto possiamo costruire i segnali di ingresso ai vari integratori secondo le equazioni del modello in forma esplicita

$$\dot{H}_1(t) = \frac{1}{a_1 w_1} [q_1(t) - q_{12}(t)]$$

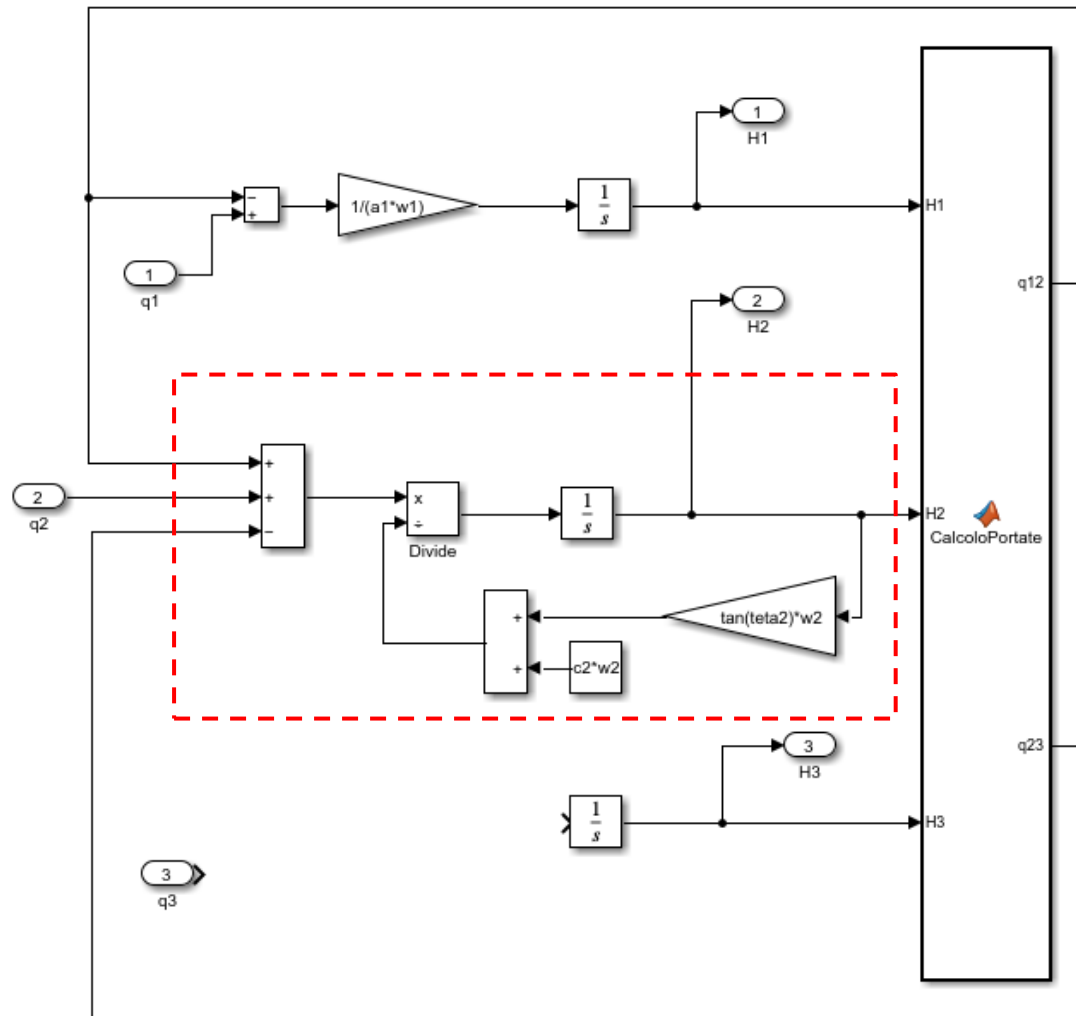
$$\dot{H}_2(t) = \frac{1}{\tan(\theta_2) w_2 H_2(t) + c_2 w_2} [q_2(t) + q_{12}(t) - q_{23}(t)]$$

$$\dot{H}_3(t) = \frac{1}{a_3 w_3} [q_3(t) + q_{23}(t)]$$

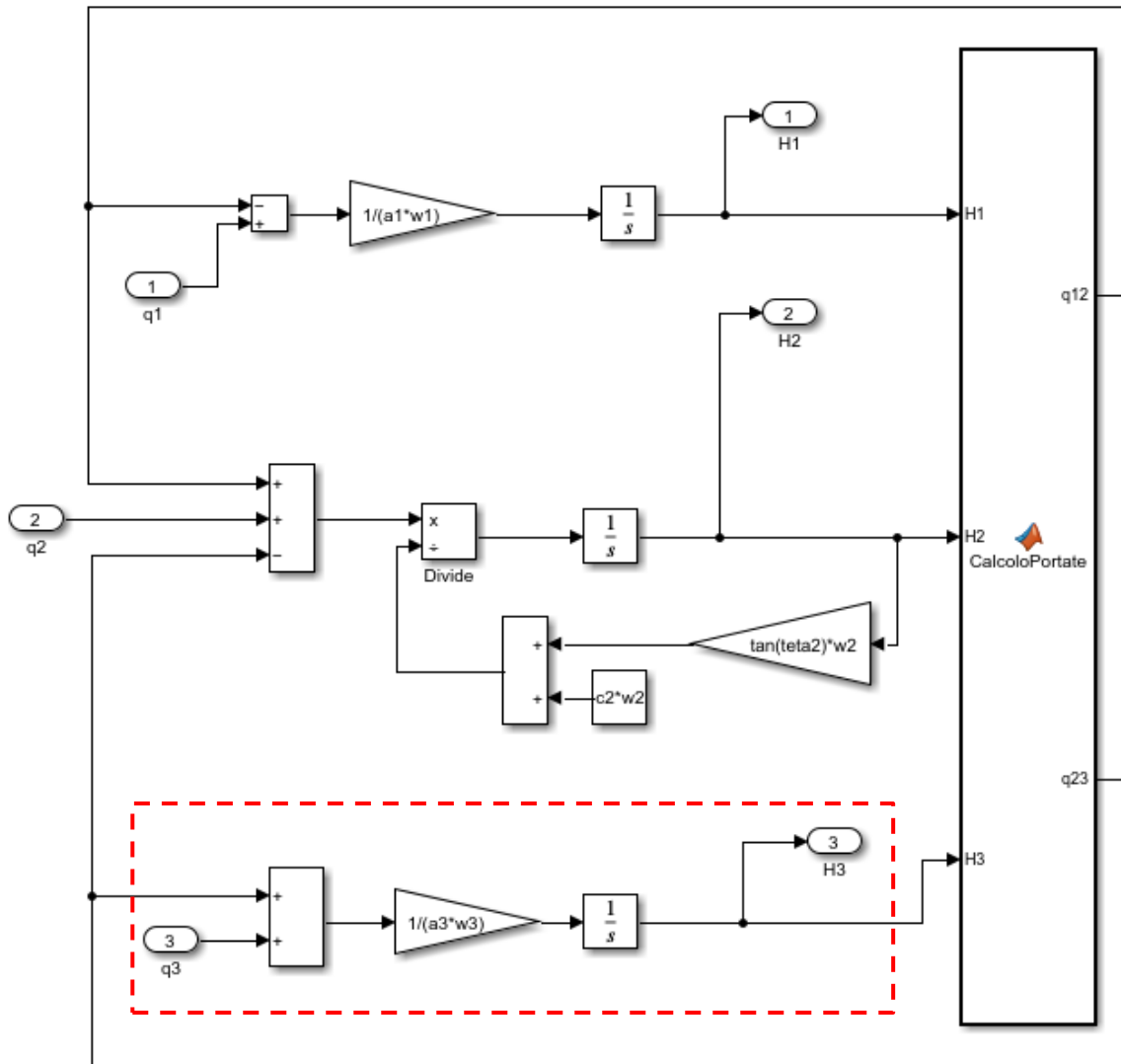
$$\dot{H}_1(t) = \frac{1}{a_1 w_1} [q_1(t) - q_{12}(t)]$$



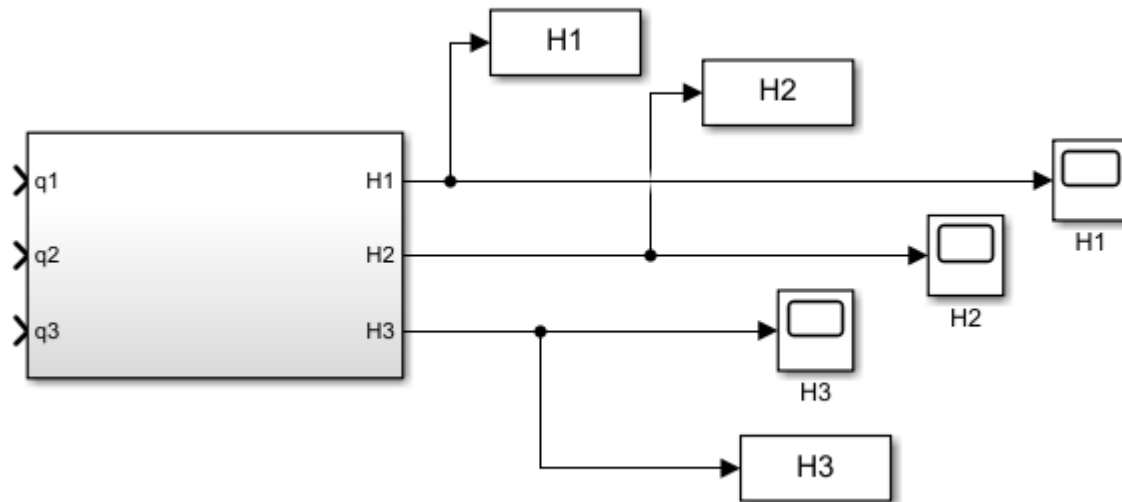
$$\dot{H}_2(t) = \frac{1}{\tan(\theta_2) w_2 H_2(t) + c_2 w_2} [q_2(t) + q_{12}(t) - q_{23}(t)]$$



$$\dot{H}_3(t) = \frac{1}{a_3 w_3} [q_3(t) + q_{23}(t)]$$



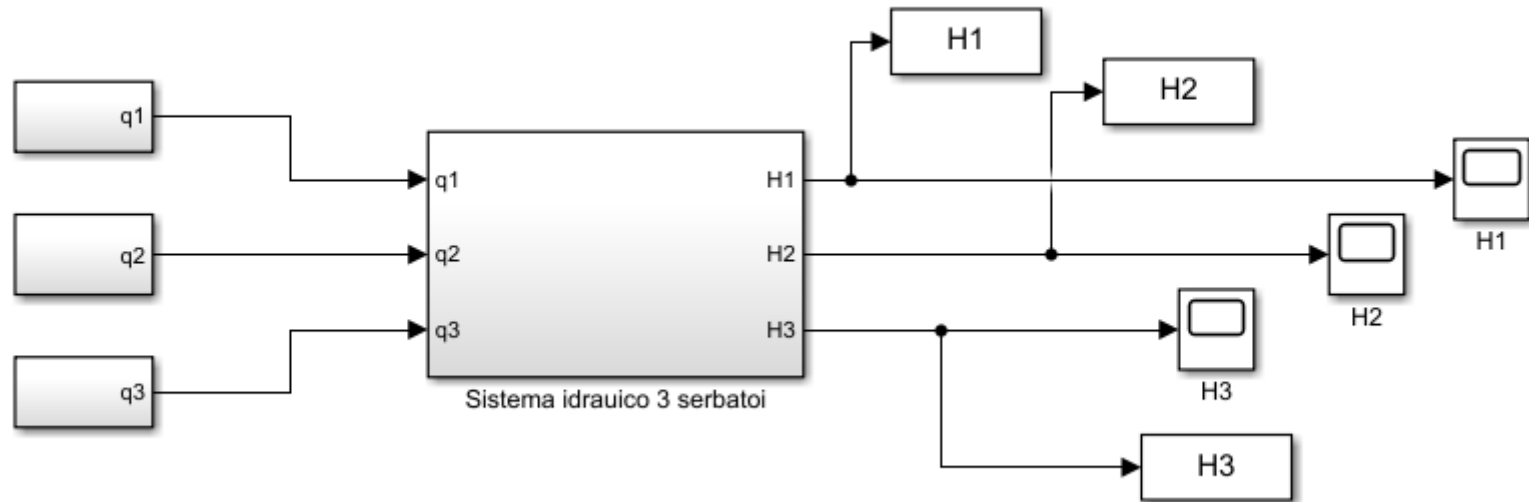
Modello completo all'interno della pagina di lavoro. Sono stati collegati ai tre terminali di uscita del Subsystem i relativi blocchi **Scope** e tre blocchi «**To workspace**» per salvare nel workspace i tre livelli dei serbatoi



Per poter avviare la simulazione devono essere generati i tre segnali $q_1(t)$, $q_2(t)$, $q_3(t)$ che rappresentano le portate esterne in ingresso ai serbatoi.

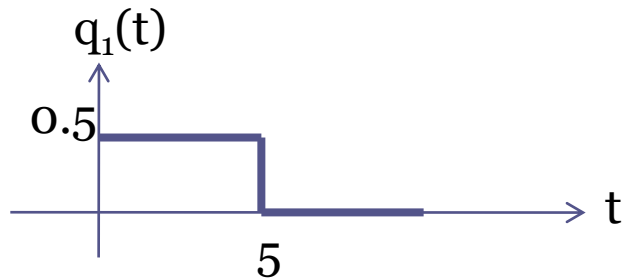
Creiamo tre Subsystem preposti alla generazione di tali segnali.

Come già fatto in precedenza, importiamo tre blocchi «Subsystem» già pronti dalla libreria dei «Commonly used blocks» e modifichiamone il contenuto. In particolare **rimuoviamo i blocchi «In» presenti al loro interno** in modo che i tre Subsystem abbiano unicamente un terminale di uscita. Diamo ai terminali di uscita dei tre Subsystem i nomi q1, q2, q3.

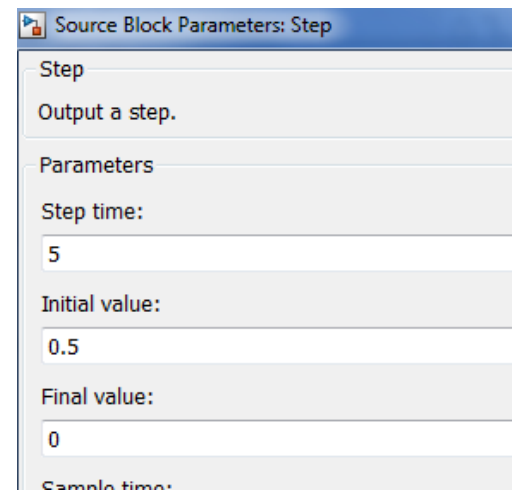
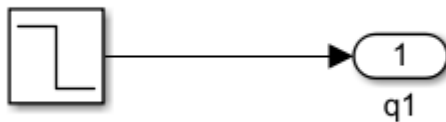


Ora inseriamo all'interno dei Subsystem i blocchi necessari per generare le tre forme d'onda desiderate per i segnali $q_1(t)$, $q_2(t)$, $q_3(t)$

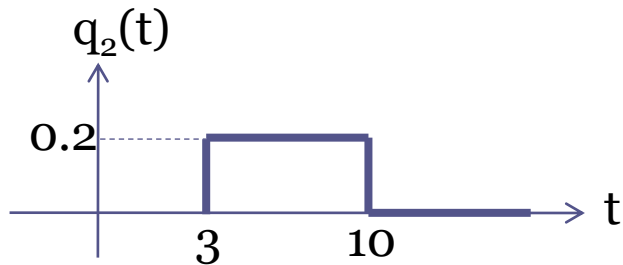
Segnale $q_1(t)$



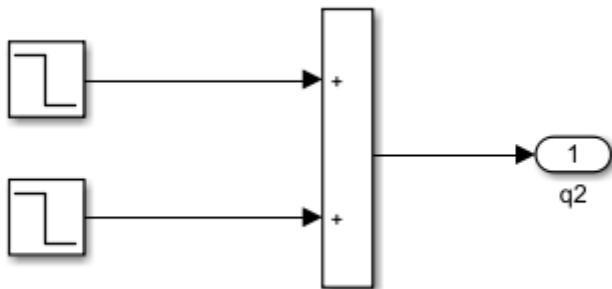
E' sufficiente un singolo blocco «Step» (libreria: Sources, blocco già spiegato in precedenza).



Segnale $q_2(t)$



Possiamo generare questo segnale sommando fra loro le uscite di due blocchi Step parametrizzati opportunamente



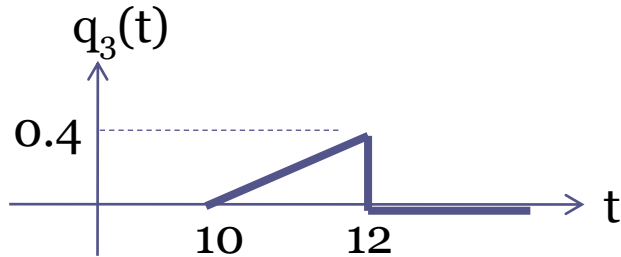
Source Block Parameters: Step1

Step
Output a step.
Parameters
Step time:
3
Initial value:
0
Final value:
0.2

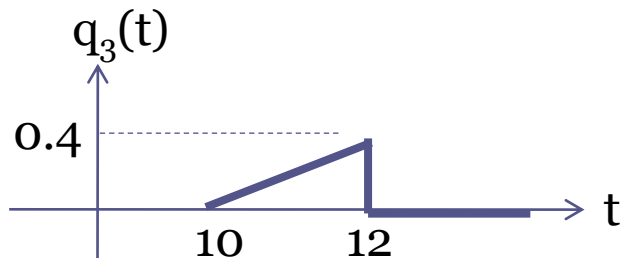
Source Block Parameters: Step4

Step
Output a step.
Parameters
Step time:
10
Initial value:
0
Final value:
-0.2

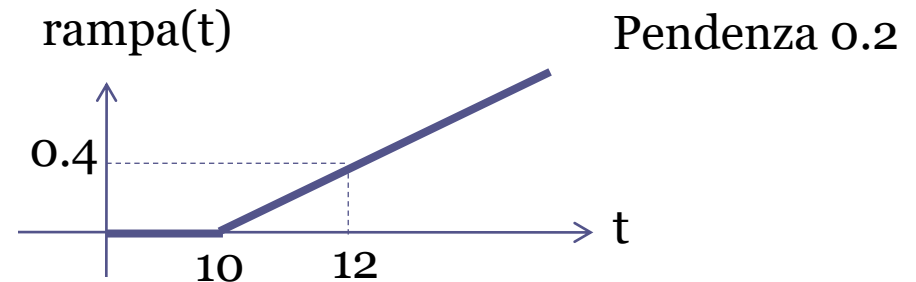
Segnale $q_3(t)$



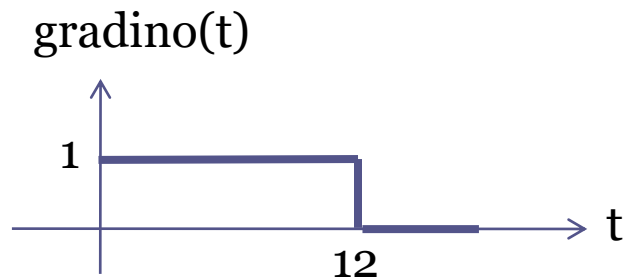
Possiamo generare questo segnale
moltiplicando fra loro un segnale a rampa
 (blocco «Ramp», libreria Sources) **ed un**
segnale a gradino (blocco Step)



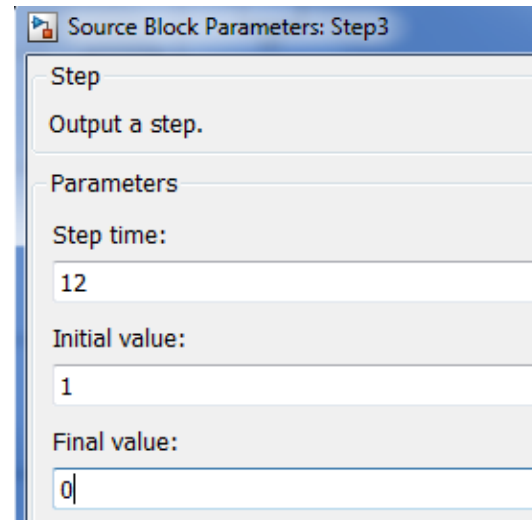
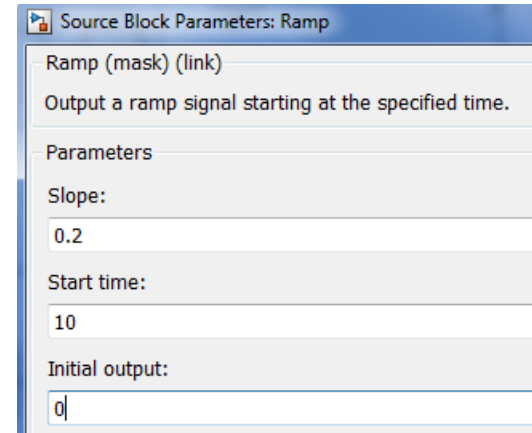
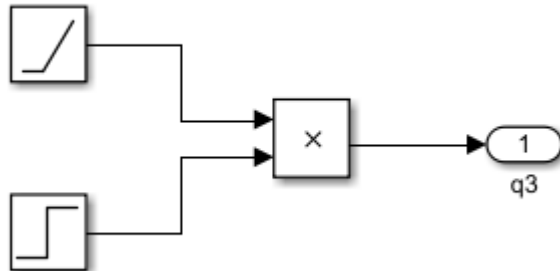
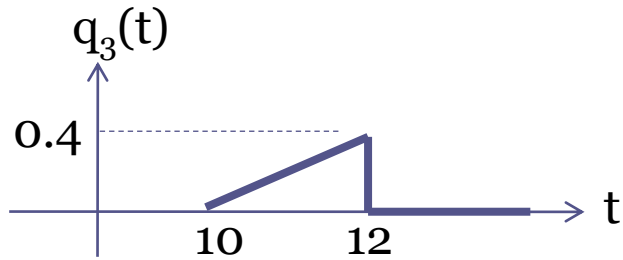
=



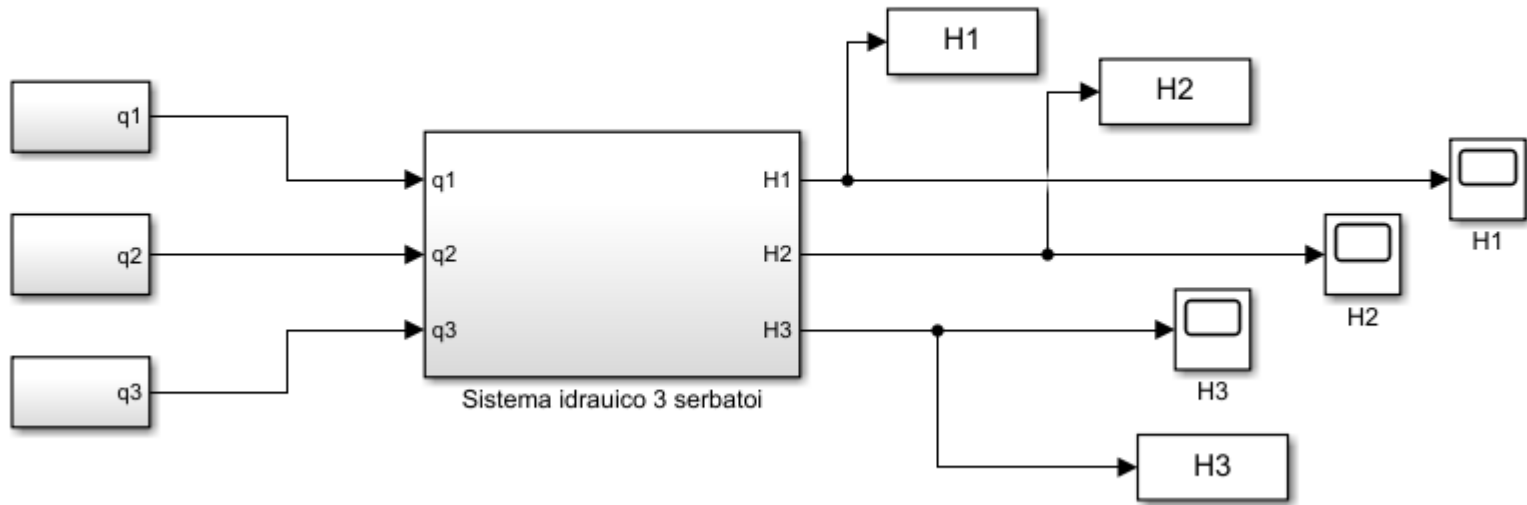
×



Segnale $q_3(t)$



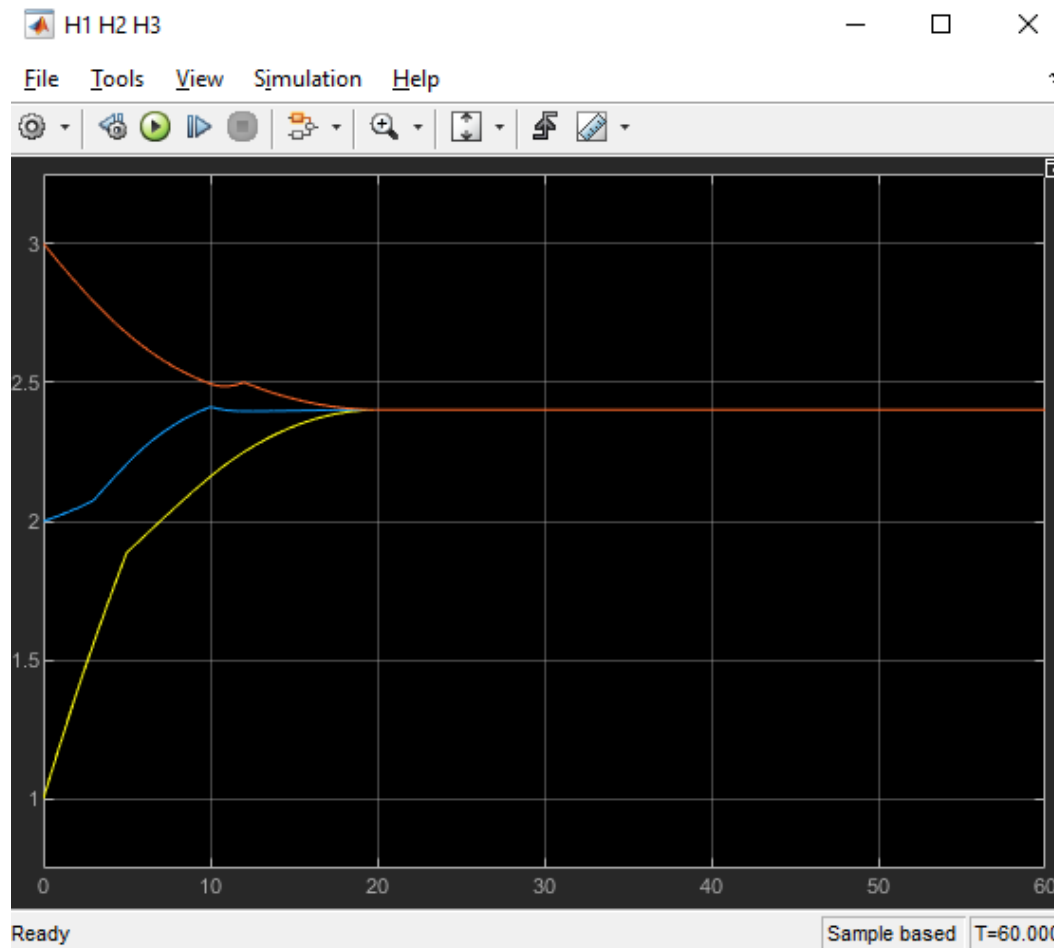
Modello completo



File: Serbatoi_v01.slx

Ora si può fare il run del modello (impostare come durata della simulazione 60 secondi) e visualizzare i segnali $H_1(t)$, $H_2(t)$, $H_3(t)$.

Se si applica in ingresso ad un blocco Scope un segnale vettoriale (si aggregano i tre segnali H1, H2, H3 per mezzo di un blocco Mux, e si colleghi l'uscita del blocco Mux all'ingresso di un blocco Scope) vengono visualizzati **sovrapposti** tutti i segnali



Nella Section finale dello script `Serbatoi_dati.m` sono inserite le istruzioni che attivano il run del modello (comando `sim`) e creano vari grafici con i risultati

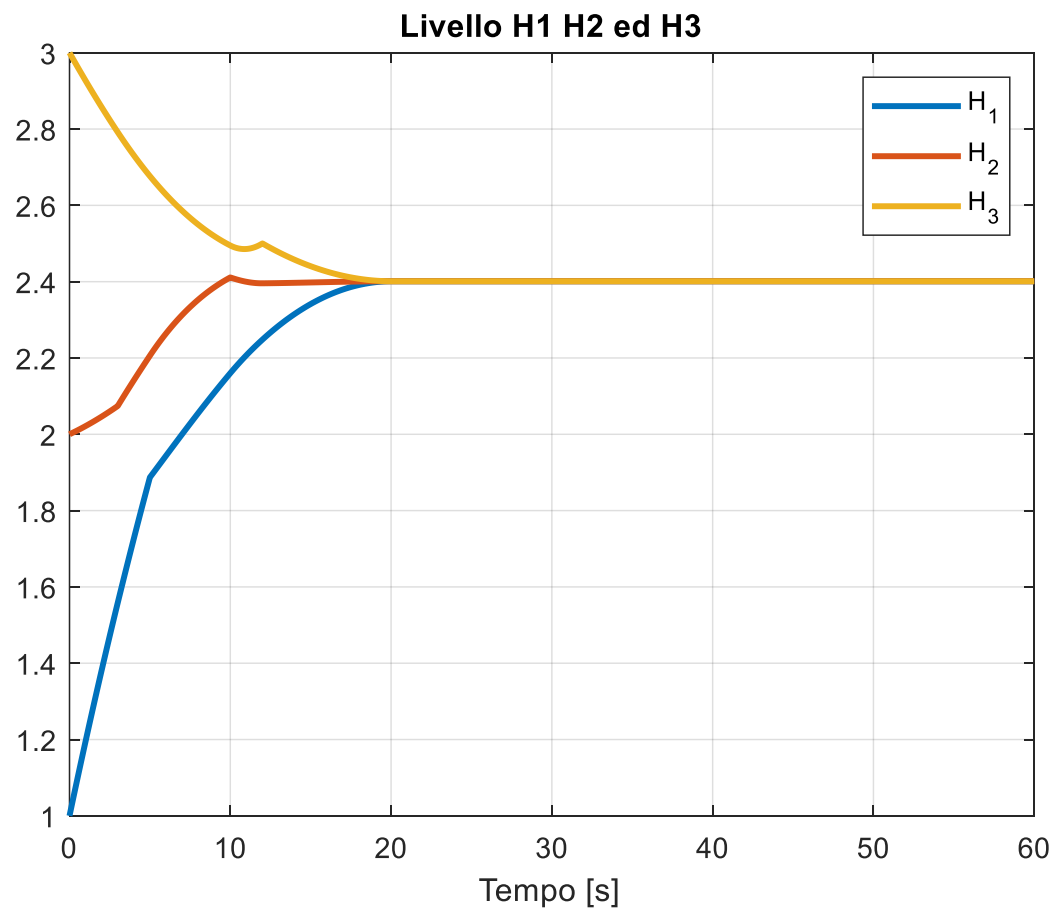
```
%% Run del modello e creazione grafici
sim('Serbatoi_v01')
```

```
figure(1)
plot(H1, 'LineWidth', 2)
grid,
xlabel('Tempo [s]')
title('Livello H1')
```

```
figure(2)
plot(H2, 'LineWidth', 2)
grid,
xlabel('Tempo [s]')
title('Livello H2')
```

```
figure(3)
plot(H3, 'LineWidth', 2)
grid,
xlabel('Tempo [s]')
title('Livello H3')
```

```
figure(4)
plot(H1.Time, [H1.Data H2.Data H3.Data], 'LineWidth', 2)
grid,
xlabel('Tempo [s]')
title('Livello H1 H2 ed H3')
legend('H_1', 'H_2', 'H_3')
```



Gestione di modelli Simulink attraverso un file Script

Ora desideriamo confrontare fra loro i segnali $H_1(t)$ e $H_2(t)$ ottenuti in corrispondenza di 4 valori distinti del coefficiente di efflusso C_{12}

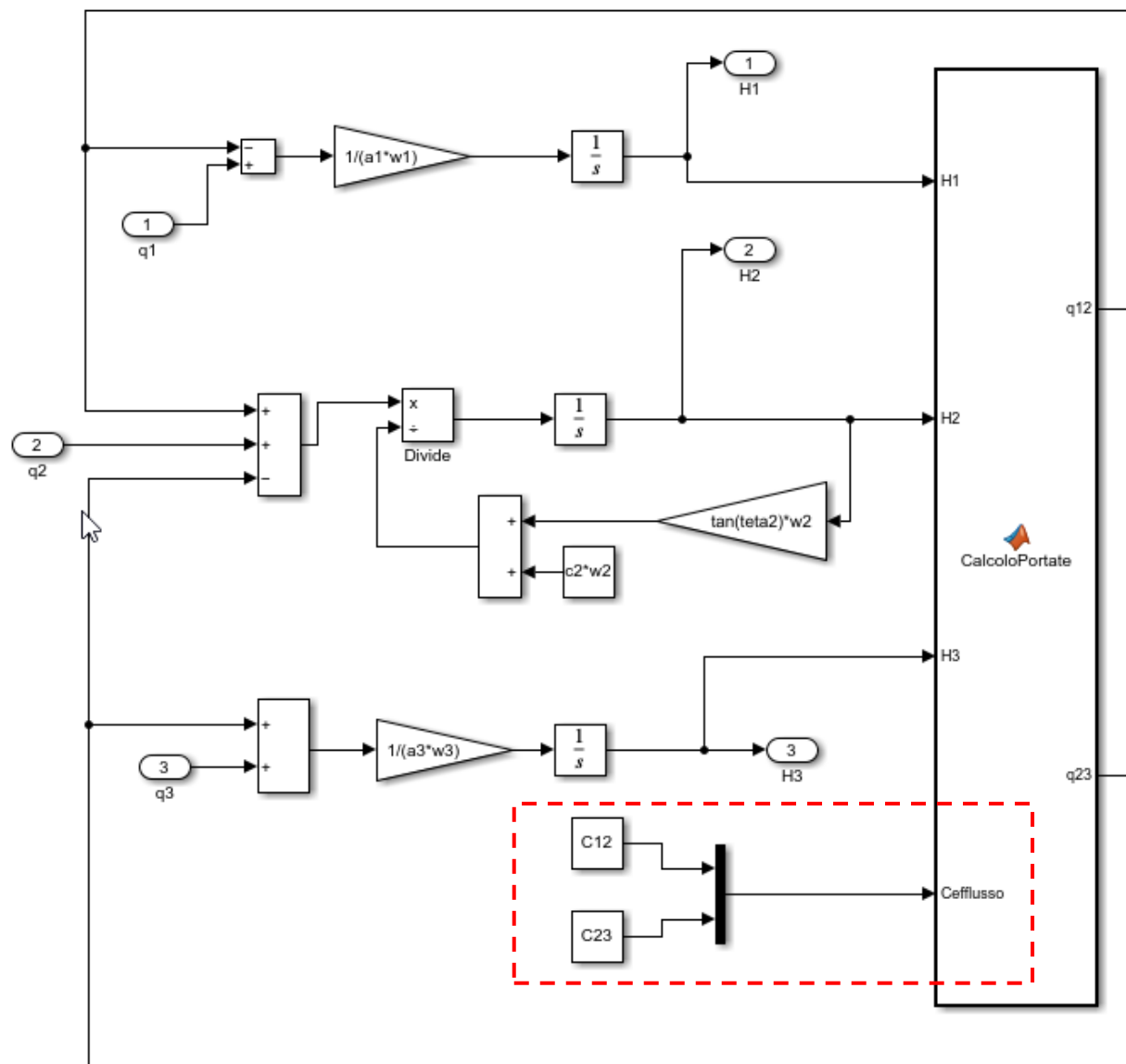
$$C_{12} = 0.5, 0.8, 1.1, 1.5$$

Realizziamo uno script che gestisca in automatico la riparametrizzazione ed i successivi run del modello Simulink e costruisca i grafici richiesti.

E' necessario realizzare una modifica del modello precedentemente realizzato per ovviare al fatto che dall'interno del blocco MATLAB function non sono accessibili le variabili del workspace e pertanto non è possibile, mediante uno script, modificarne il valore in successivi run del programma senza andare esplicitamente a modificare il corpo del blocco MATLAB function

Ciò che facciamo è **passare i parametri C_{12} e C_{23} come ulteriori ingressi al blocco MATLAB function**

Decidiamo di passare in ingresso al blocco MATLAB function entrambi i parametri C_{12} e C_{23} (che in modelli più complessi potrebbero essere anche ben più di due) mediante una **singola variabile addizionale di ingresso**, che dovrà evidentemente essere un segnale vettoriale



```

clear all,clc
%% Esecuzione dei test di confronto
% Parametri Test 1
% Parametri serbatoio #1
a1=5;
w1=1;
% Parametri serbatoio #2
c2=3;
w2=1;
teta2=pi/4;
% Parametri serbatoio #2
a3=4;
w3=2;
% Coefficienti di efflusso
C12=0.5;
C23=0.6;
% Condizioni iniziali
H10=1;
H20=2;
H30=3;

sim('Serbatoi_v02')
H1_test1=H1;
H2_test1=H2;

File: Serbatoi_dati_gestione.m

% Coefficiente di efflusso nel Test 2
C12=0.8;
sim('Serbatoi_v02')
H1_test2=H1;
H2_test2=H2;

% Coefficiente di efflusso nel Test 3
C12=1.1;
sim('Serbatoi_v02')
H1_test3=H1;
H2_test3=H2;

% Coefficiente di efflusso nel Test 4
C12=1.5;
sim('Serbatoi_v02')
H1_test4=H1;
H2_test4=H2;

```


Ad ogni run del modello, che attiviamo con il comando `sim`, il modello Simulink scrive nel Workspace le variabili H1, H2 ed H3

Dopo ogni run del modello i dati esportati nel workspace devono essere salvati in variabili ausiliarie (in questo caso H1_testx e H2_testx con x=1,2,3,4).

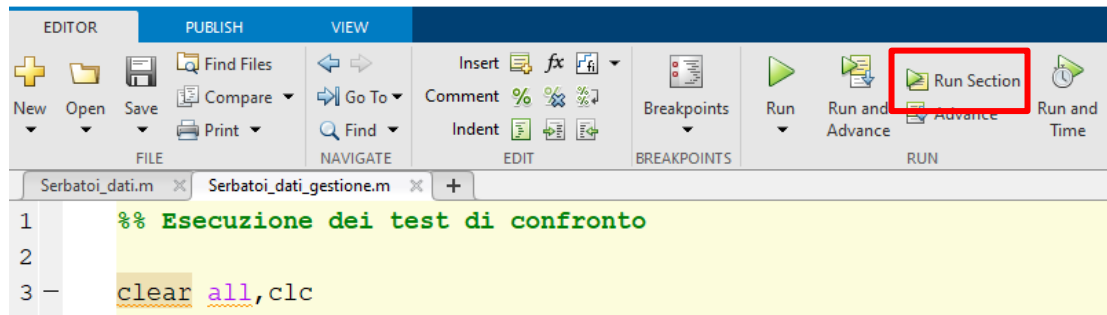
In caso contrario il successivo run del modello sovrascrive i dati della simulazione precedente, che andrebbero persi

Possiamo ora aggiungere allo script una ulteriore Section preposta alla costruzione di grafici di confronto

```
%% Creazione grafici
figure(1)
plot(H1_test1.Time, [H1_test1.Data H1_test2.Data H1_test3.Data H1_test4.Data], 'LineWidth', 2)
grid,
xlabel('Tempo [s]')
title('Livelli H1 nei 4 test')
legend('C_{12}=0.5', 'C_{12}=0.8', 'C_{12}=1.1', 'C_{12}=1.5')

figure(2)
plot(H2_test1.Time, [H2_test1.Data H2_test2.Data H2_test3.Data H2_test4.Data], 'LineWidth', 2)
grid,
xlabel('Tempo [s]')
title('Livelli H2 nei 4 test')
legend('C_{12}=0.5', 'C_{12}=0.8', 'C_{12}=1.1', 'C_{12}=1.5')
```

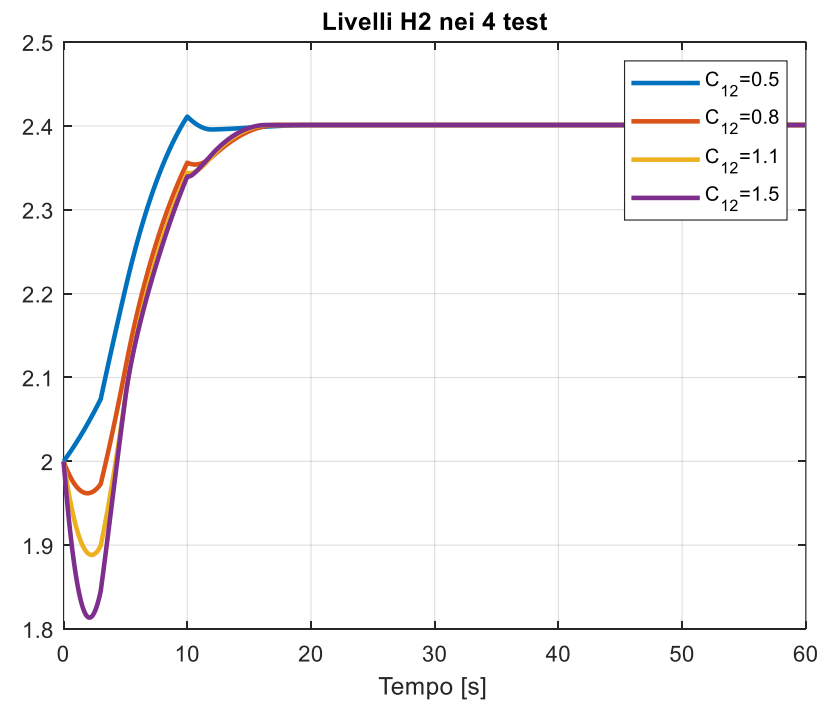
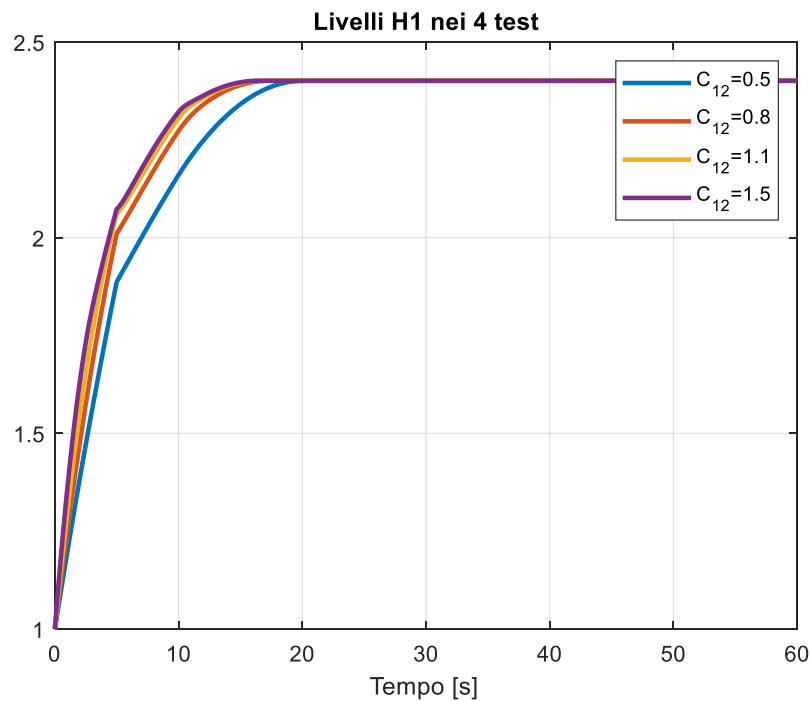
Si esegua la prima Section dello Script (rendere attiva la Section cliccando su di essa, e poi premere «Run Section» o in alternativa Ctrl+Invio)



Workspace		
Name ▲	Value	Size
a1	5	1x1
a3	4	1x1
b2	2	1x1
C12	1.5000	1x1
c2	3	1x1
C23	0.6000	1x1
d2	5	1x1
H1	1x1 double tim...	1x1
H10	1	1x1
H1_test1	1x1 double tim...	1x1
H1_test2	1x1 double tim...	1x1
H1_test3	1x1 double tim...	1x1
H1_test4	1x1 double tim...	1x1
H2	1x1 double tim...	1x1
H20	2	1x1
H2_test1	1x1 double tim...	1x1
H2_test2	1x1 double tim...	1x1
H2_test3	1x1 double tim...	1x1
H2_test4	1x1 double tim...	1x1
H3	1x1 double tim...	1x1
H30	3	1x1
tout	6001x1 double	6001x1
w1	1	1x1
w2	1	1x1
w3	2	1x1

Viene eseguita la sequenza di test, e vengono salvate nel workspace le relative variabili

Eseguendo ora la seconda Section vengono prodotti i seguenti grafici



Il blocco «Fcn» come alternativa al blocco «Matlab Function»

Illustriamo un ulteriore blocco elementare Simulink per calcolare in maniera alternativa le portate che fluiscono fra i due serbatoi.

E' un blocco funzione concettualmente analogo al blocco «Matlab Function», decisamente meno versatile (in quanto può ricevere solo un argomento in ingresso, può unicamente restituire in uscita un segnale scalare, e non può utilizzare al proprio interno un codice Matlab «strutturato»)

Tale blocco si chiama «**Fcn**», ed è disponibile all'interno della libreria «User Defined Functions».

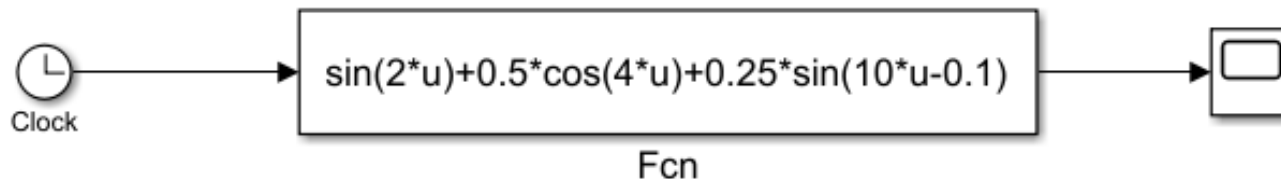
Il blocco Fcn riceve in ingresso un segnale (scalare o vettoriale) e consente di scrivere al suo interno una **espressione** contenente funzioni Matlab che viene applicata al segnale di ingresso (indicato come u nella espressione) ed il cui risultato viene restituito sul terminale di uscita sotto forma di una quantità **scalare**.

L'espressione può includere uno o più dei seguenti elementi

- **u** — The input to the block. **If u is a vector, $u(i)$ represents the i th element of the vector.**
 $u(1)$ or u alone represents the first element.
- **Numeric constants** . (pi, eps, ...).
- **Arithmetic operators** (+ - * / ^).
- **Relational operators** (== != > < >= <=) — The expression returns 1 if the relation is true; otherwise, it returns 0.
- **Logical operators** (&& || !) — The expression returns 1 if the relation is true; otherwise, it returns 0.
- **Parentheses**.
- **Mathematical functions** —
abs, acos, asin, atan, atan2, ceil, cos, cosh, exp, floor, hypot, log, log10, power, rem, sgn
(equivalent to [sign](#) in MATLAB), sin, sinh, sqrt, tan, and tanh.
- **Workspace variables** — Matrix or vector elements must be specifically referenced
(e.g., A(1,1) instead of A for the first element in the matrix).

Nella sua modalità di impiego «standard» il blocco riceve in ingresso un segnale scalare che all'interno dell'espressione viene denotato con «u»

Per costruire la forma d'onda $\sin(2t) + 0.5 \cos(4t) + 0.25 \sin(10t - 0.1)$ (esempio trattato in precedenza) si può utilizzare la seguente configurazione

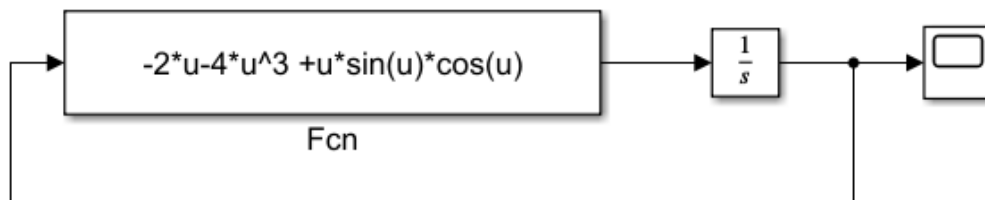


Per risolvere il problema di Cauchy

$$\dot{x}(t) = -2x(t) - 4x^3(t) + x(t) \sin(x(t)) \cos(x(t))$$

$$x(0) = 1$$

si può utilizzare la seguente configurazione



File: Cauchy.slx

E' anche possibile passare in ingresso al blocco Fcn un **segnale «vettoriale»** cioè un array di segnali $s_1(t), s_2(t), \dots, s_n(t)$

$$\begin{bmatrix} s_1(t) \\ s_2(t) \\ \vdots \\ s_n(t) \end{bmatrix}$$

Questa è la modalità di impiego che ci serve nel presente esempio per generare mediante il blocco Fcn i segnali

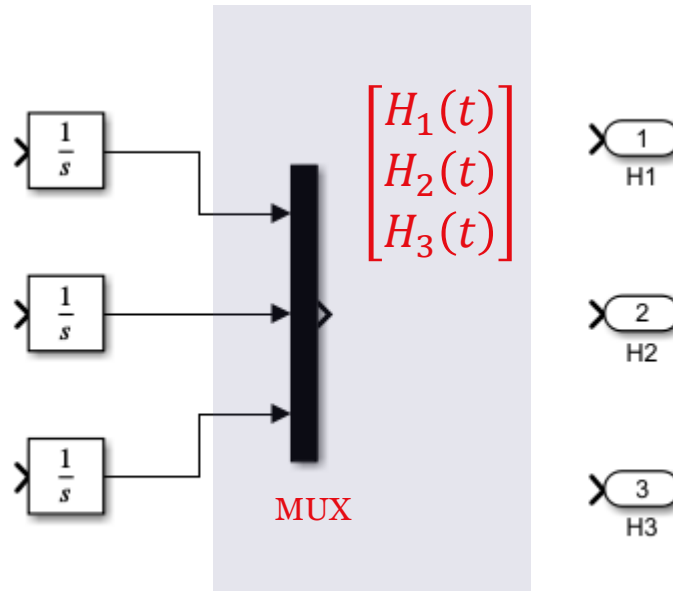
$$q_{12}(t) = C_{12} \sqrt{|H_1(t) - H_2(t)|} \cdot \text{sign}[H_1(t) - H_2(t)]$$

$$q_{23}(t) = C_{23} \sqrt{|H_2(t) - H_3(t)|} \cdot \text{sign}[H_2(t) - H_3(t)]$$

Salviamo il file **Serbatoi_v02** come **Serbatoi_v03** e dopo aver cancellato il blocco Matlab Function «Calcolo Portate» creiamo all'interno del Subsystem «Sistema idraulico 3 serbatoi» un array di segnali che contenga i tre livelli dei serbatoi

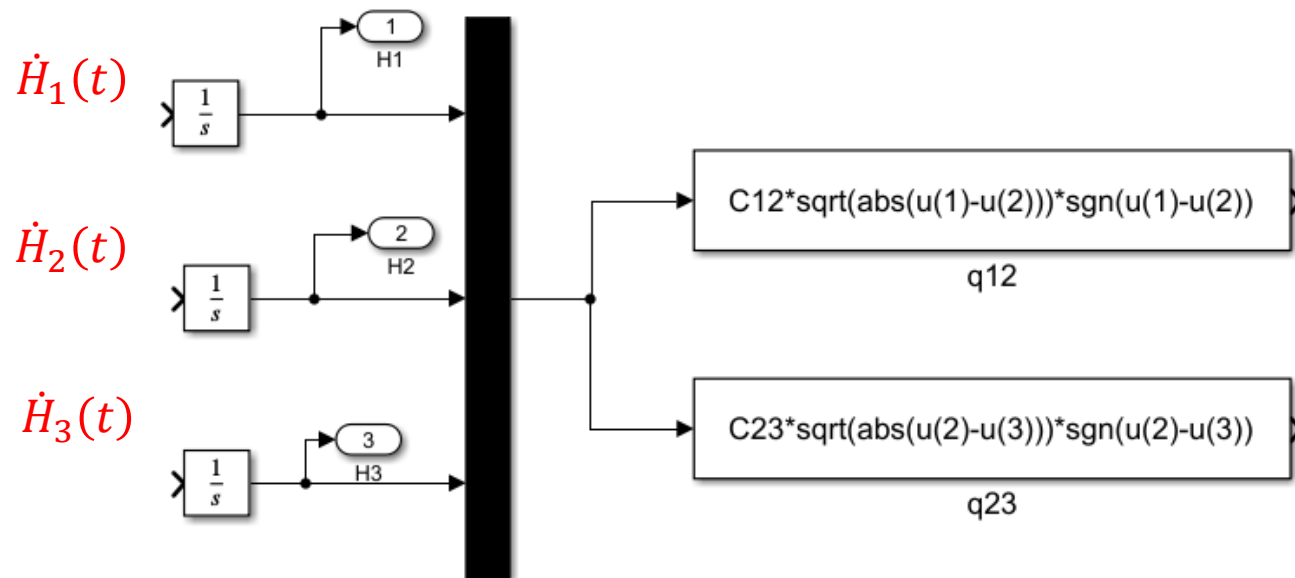
$$\begin{bmatrix} H_1(t) \\ H_2(t) \\ H_3(t) \end{bmatrix}$$

La creazione di un array di segnali si fa mediante il blocco Simulink «Mux» (libreria «Commonly Used Blocks»)



Il blocco «Mux» ha di default due terminali di ingresso. Lo si configuri in modo da poter accorpare fra loro tre segnali (Number of inputs: 3).

Ora possiamo applicare il segnale vettoriale in ingresso a due istanze del blocco Fcn, e scrivere le relative espressioni per q_{12} e q_{23} . **All'interno delle espressioni, le tre componenti del vettore in ingresso sono indicate come $u(1)$, $u(2)$, $u(3)$**



$$q_{12}(t) = C_{12} \sqrt{|H_1(t) - H_2(t)|} \cdot \text{sign}[H_1(t) - H_2(t)]$$

$$q_{23}(t) = C_{23} \sqrt{|H_2(t) - H_3(t)|} \cdot \text{sign}[H_2(t) - H_3(t)]$$

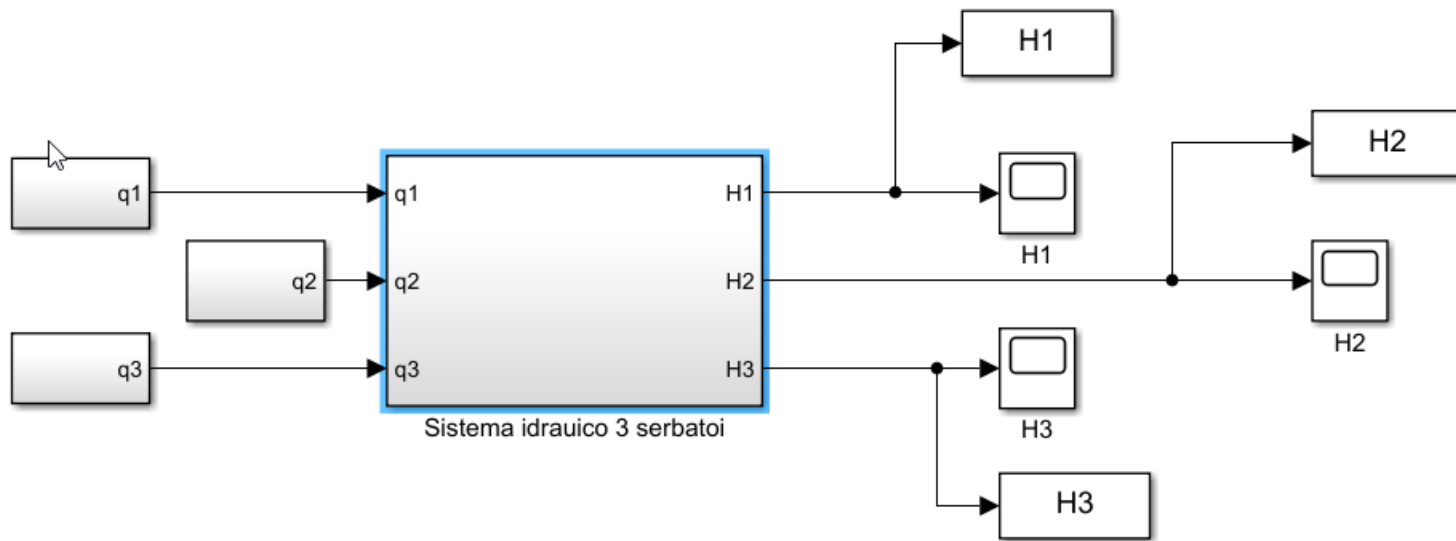
Files: Serbatoi_dati_gestione_Fcn.m
Serbatoi_v03.slx

Presentiamo una ulteriore variante in termini di modalità di realizzazione del modello Simulink che riduce ulteriormente, e **significativamente**, la complessità del modello di simulazione

Prendiamo come punto di partenza il file **Serbatoi_v02.slx**.

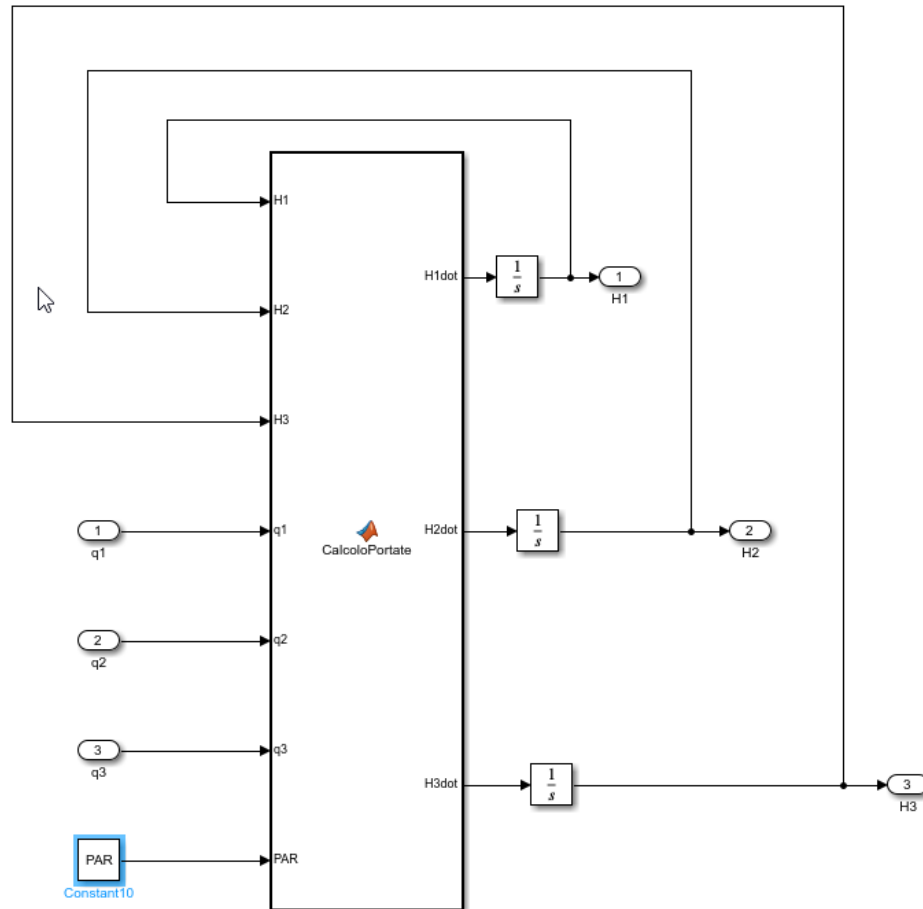
L'idea è quella di **utilizzare in maniera più corposa il blocco MATLAB function, facendo in modo che questo determini in uscita direttamente i segnali $\dot{H}_1(t)$, $\dot{H}_2(t)$ e $\dot{H}_3(t)$ con cui alimentare i tre integratori presenti nel modello**. In tal modo si vanno ad eliminare tutti quei blocchi elementari che sono stati impiegati allo scopo.

L'aspetto del modello nella pagina di lavoro principale resta invariato.



Cambia unicamente l'interno del Subsystem «Sistema idraulico 3 serbatoi»

IL blocco MATLAB function è configurato per ricevere in ingresso i tre livelli $H_1(t)$, $H_2(t)$ e $H_3(t)$, le tre portate $q_1(t)$, $q_2(t)$ e $q_3(t)$ in ingresso ai tre serbatoi, ed un vettore chiamato PAR che contiene tutti i parametri costanti del modello.



Il blocco MATLAB function produce in uscita i tre segnali $\dot{H}_1(t)$, $\dot{H}_2(t)$ e $\dot{H}_3(t)$ con cui vengono direttamente alimentati i tre integratori.

NB Il blocco constant genera (leggendolo dal workspace) il **vettore costante** PAR e lo applica in ingresso al blocco MATLAB function

File: Serbatoi_v04_MATLFunction.slx

Corpo del blocco MATLAB function

```
function [H1dot,H2dot,H3dot] = CalcoloPortate(H1,H2,H3,q1,q2,q3,PAR)
```

```
a1=PAR(1);
w1=PAR(2);
c2=PAR(3);
w2=PAR(4);
teta2=PAR(5);
a3=PAR(6);
w3=PAR(7);
C12=PAR(8);
C23=PAR(9);
```

Queste istruzioni estraggono dal vettore PAR i 9 parametri costanti. Ovviamente l'ordine dei parametri deve essere coerente con il modo con cui sia stato costruito il vettore PAR nello script di gestione.

```
q12=C12*sqrt(abs(H1-H2))*sign(H1-H2)
q23=C23*sqrt(abs(H2-H3))*sign(H2-H3)
```

Calcolo delle portate q12 e q23

```
H1dot=1/(a1*w1)*(q1-q12)
H2dot=1/(tan(teta2)*w2*H2+c2*w2)*(q2+q12-q23)
H3dot=1/(a3*w3)*(q3+q23)
```

Calcolo delle 3 uscite

Avendo impiegato per le variabili interne alla funzione il loro nome fisico, le istruzioni che definiscono le 3 uscite non fanno altro che riprodurre fedelmente i membri destri delle equazioni in forma esplicita.

Le modifiche da apportare allo script di gestione si limitano **alla definizione del vettore PAR nella parametrizzazione iniziale:**

```
%Parametri Test 1
% Parametri serbatoio #1
a1=5;
w1=1;
% Parametri serbatoio #2
c2=3;
w2=1;
teta2=pi/4;
% Parametri serbatoio #2
a3=4;
w3=2;
% Coefficienti di efflusso
C12=0.5;
C23=0.6;

PAR=[a1 w1 c2 w2 teta2 a3 w3 C12 C23]
```

ed alla sua **ridefinizione** ogni volta che viene modificato il valore di un parametro prima dei successivi run del modello

```
% Coefficiente di efflusso nel Test 2
C12=0.8;
PAR=[a1 w1 c2 w2 teta2 a3 w3 C12 C23]
sim('Serbatoi_v03_MATLFunction')
H1_test2=H1;
H2_test2=H2;
```

File: Serbatoi_dati_gestione_MATfunction.m

Appendice

Function files e anonymous functions

Function files

Sono dei particolari files matlab che, a differenza dagli script, restituiscono esplicitamente una o più variabili in uscita che dipendono da una o più variabili passate alla funzione come parametri di ingresso.

Se non specificato diversamente, un Function File lavora esclusivamente su variabili locali create e distrutte alla attivazione ed alla chiusura della funzione.

All'interno di un Function File NON sono accessibili le variabili del workspace.

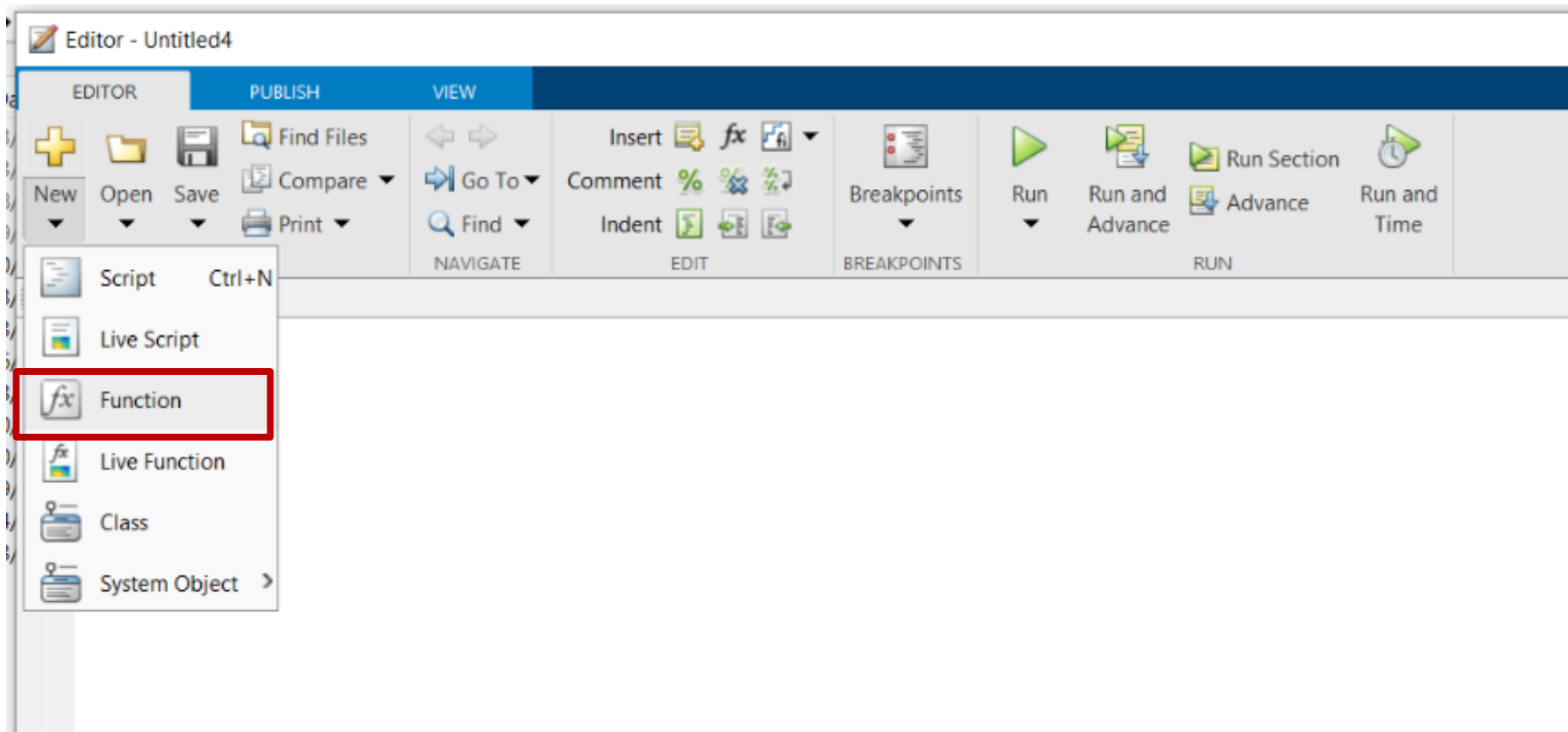
Si scrivono seguendo un formato standard, e **non** possono essere eseguite con il pulsante Save&Run dell'editor perche normalmente “attendono” argomenti in input.

Si avviano normalmente dal prompt di Matlab (o da uno script) con una sintassi del tipo:

```
[out1 out2]=nome_funzione(in1,in2,in3)
```

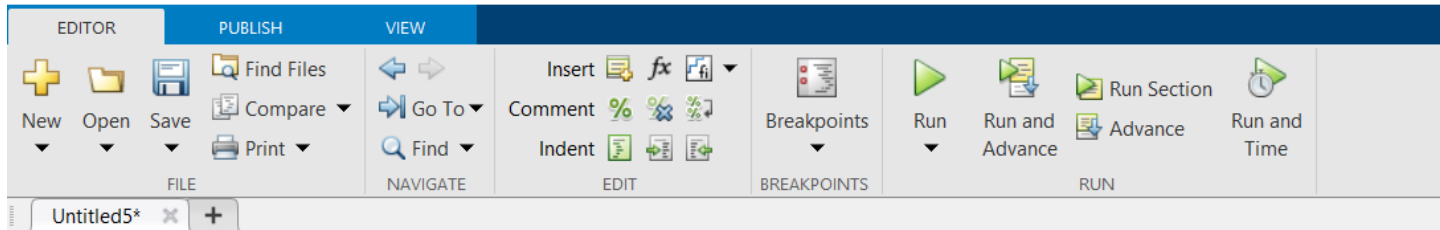
Questo è un esempio di chiamata ad una funzione che **accetta tre argomenti in ingresso** e restituisce **due argomenti in uscita**. Sia gli ingressi che le uscite possono essere variabili di qualunque tipo (array, stringhe di testo,...).

Selezionare dall'editor: New->Function



Template di un function file

Editor - Untitled5*



```

1  function [outputArg1,outputArg2] = untitled5(inputArg1,inputArg2) → Riga di intestazione
2  %UNTITLED5 Summary of this function goes here
3  % Detailed explanation goes here → Help
4  outputArg1 = inputArg1;
5  outputArg2 = inputArg2; → Corpo della funzione
6  end
7
8

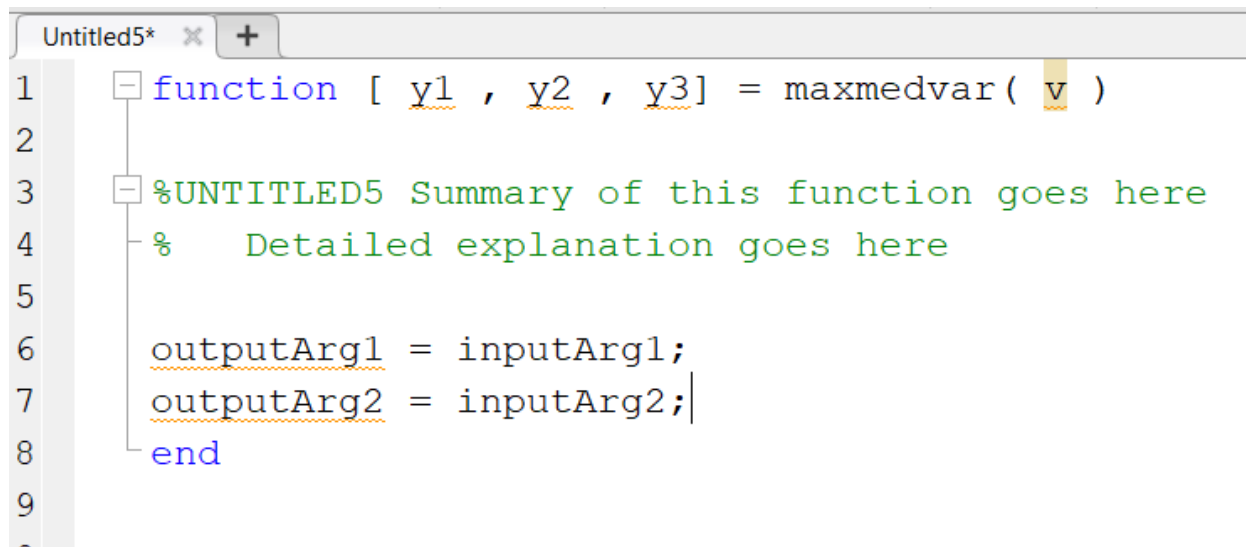
```

Il template si riferisce ad una funzione che accetta in ingresso due argomenti, e ne restituisce all'esterno altri due.

Esempio:

Scriviamo un function file che accetta in ingresso un vettore e ne restituisce il massimo, la media, e la varianza. Decidiamo che tale funzione si chiamerà «**maxmedvar**»

Come primo passo modifichiamo la prima riga della funzione, inserendo il numero corretto di ingressi e uscite, ed il nome della funzione



```

Untitled5*  x  +
1  function [ y1 , y2 , y3] = maxmedvar( v )
2
3  %UNTITLED5 Summary of this function goes here
4  %    Detailed explanation goes here
5
6  outputArg1 = inputArg1;
7  outputArg2 = inputArg2;
8  end
9

```

I nomi attribuiti a ingressi uscite sono completamente arbitrari. Quando, dal prompt di Matlab o dall'interno di uno script, si chiama la funzione è possibile usare, per gli ingressi e le uscite della stessa, dei nomi completamente differenti da quelli usati nella riga di intestazione e nel corpo della funzione

Ora modifichiamo il corpo della funzione. Esso deve contenere le istruzioni che **assegnano un valore a tutte le variabili di uscita**. Tali istruzioni possono utilizzare esplicitamente i parametri passati in ingresso alla funzione. All'interno di un Function File, come già detto, **NON** sono accessibili le variabili del workspace.

```
Untitled5*  x  +
1  function [ y1 , y2 , y3] = maxmedvar( v )
2
3  %UNTITLED5 Summary of this function goes here
4  %    Detailed explanation goes here
5
6  y1=max(v) ;
7  y2=mean(v) ;
8  y3=var(v) ;
9  |
10 end
1
```

Per assegnare un valore alle tre uscite di questa funzione, utilizziamo le funzioni matlab `max`, `mean` e `var`

Per concludere, anche se non è indispensabile, si può inserire un help che spieghi come opera la funzione e cosa faccia. L'help può opzionalmente includere degli esempi di uso della funzione stessa.

```
function [ y1 , y2 , y3 ] = maxmedvar( v )

% maxmedvar: questa funzione riceve un vettore v in ingresso e ne
% restituisce il massimo, la media e la varianza mediante tre variabili
% distinte di uscita.
% Esempio:      [mass med var]=maxmedvar(v)

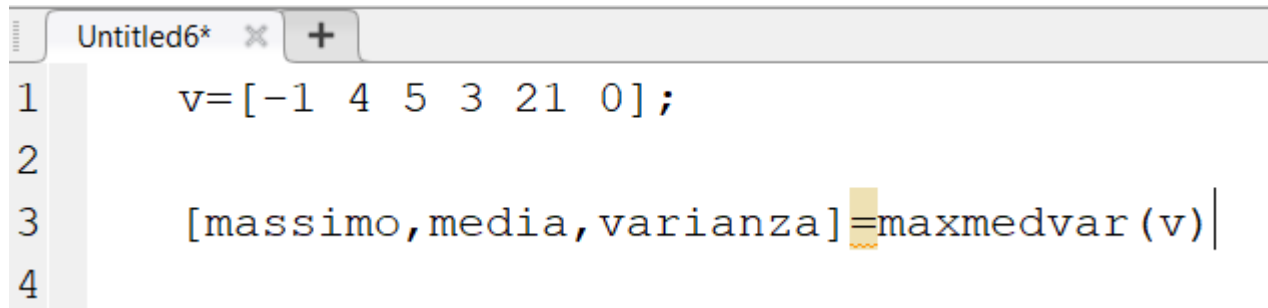
y1=max(v);
y2=mean(v);
y3=var(v);

end
```

Ora salviamo il function file attribuendogli **lo stesso nome** della funzione che abbiamo inserito nella riga di intestazione. Questo è, tra l'altro, il nome che viene proposto da Matlab al momento del salvataggio.

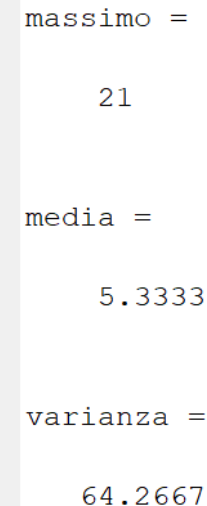
Scriviamo uno script che impiega la funzione appena creata.

Lo script non farà altro che creare un vettore di test e passarlo in ingresso alla funzione.



```
1 v = [-1 4 5 3 21 0];
2
3 [massimo, media, varianza] = maxmedvar(v);
4
```

Output restituito dallo script nel prompt dei comandi



```
massimo =
    21

media =
    5.3333

varianza =
    64.2667
```

Digitando nel prompt dei comandi `help maxmedvar` compare:

```
>> help maxmedvar  
maxmedvar: questa funzione riceve un vettore in ingresso e ne  
restituisce il massimo, la media e la varianza mediante tre variabili  
distinte di uscita.  
Esempio:    [mass med var]=maxmedvar(v)
```

Realizziamo la funzione in maniera diversa: ora facciamo in modo che restituisca all'esterno non più tre variabili distinte ma bensì un vettore di tre elementi le cui componenti siano, nell'ordine, il valore massimo, la media e la varianza del vettore in ingresso.

```
function [out] = maxmedvar2( v )

% maxmedvar: questa funzione riceve un vettore in ingresso e ne
% restituisce il massimo, la media e la varianza
% mediante un vettore di tre elementi.
% Esempio:
% out=maxmedvar(v)
% massimo=out(1);
% media=out(2);
% varianza=out(3);

out(1)=max(v);
out(2)=mean(v);
out(3)=var(v);

end
```


Esempio

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

```
function [determ autov] = mat1(A1,A2)
%mat1 funzione che riceve in input due matrici quadrate
% di pari dimensione e restituisce il determinante e gli
% autovalori della matrice prodotto
% Esempio di utilizzo:
% [D A]=mat1(M1,M2)

determ=det(A1*A2);
autov=eig(A1*A2);
end
```

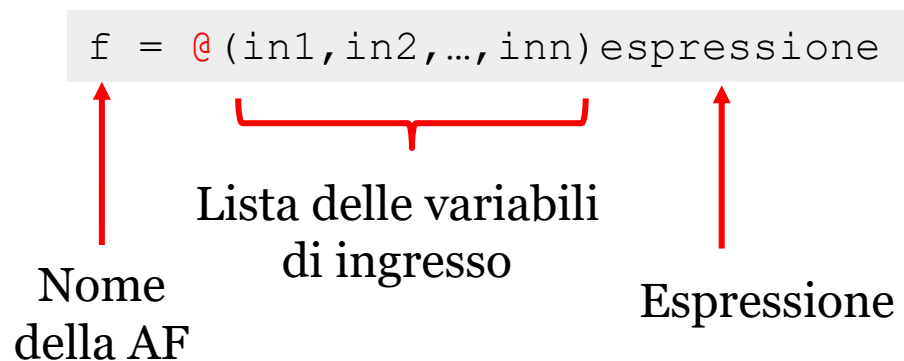
Anonymous functions (AF)

E' a tutti gli effetti un function file che non viene salvato in un m-file ma risiede in una variabile. E' quindi locale alla sessione di lavoro corrente e semplifica progetti complessi riducendo il numero complessivo di files del progetto.

Risulta essere una espressione che può avere un numero arbitrario di variabili di ingresso ma un'unica uscita. Tale uscita può però essere costituita da un vettore, o da una matrice, quindi è possibile fare in modo che la AF restituisca un numero arbitrario di quantità di interesse.

Una AF sarà definita normalmente all'interno di uno script o di un function file.

La **sintassi** per creare una AF è la seguente



Esempio Scrivere una AF che implementa la seguente funzione:

$$g(x) = x^2 + 3x + 4$$

La AF sarà definita dalla seguente istruzione:

```
g = @ (x) x^2+3*x+4;
```

E' possibile impiegare la AF, ad esempio in uno script, come segue:

```
a1=g (4)
a2=g (10)
```

ottenendo il seguente output

```
a1 =
    32

a2 =
   134
```

Una AF può avere più di un parametro in ingresso

Esempio Scrivere una AF che implementa la funzione:

$$h(x, n) = x^n$$



```
h = @ (x, n) x^n;
```

Esempio Scrivere una AF che implementa la funzione:

$$f(x, y) = x^2 + y^2 + 2xy + y + 5$$



```
f = @ (x, y) x^2+y^2+2*x*y+y+5;
```

Una AF può ricevere in ingresso vettori o matrici

Esempio Scrivere una AF che riceve in input due vettori e restituisce uno scalare che contiene la somma di tutti gli elementi del primo vettore più il primo elemento del secondo vettore:



```
g=@(v,w)(sum(v)+w(1));
```

Una AF può restituire in uscita vettori o matrici

Esempio Scrivere una AF che riceve in input uno scalare x e restituisce in uscita

la matrice: $\begin{bmatrix} 1 & x & x^2 \\ 0 & 1 & x \\ 0 & 0 & 1 \end{bmatrix}$

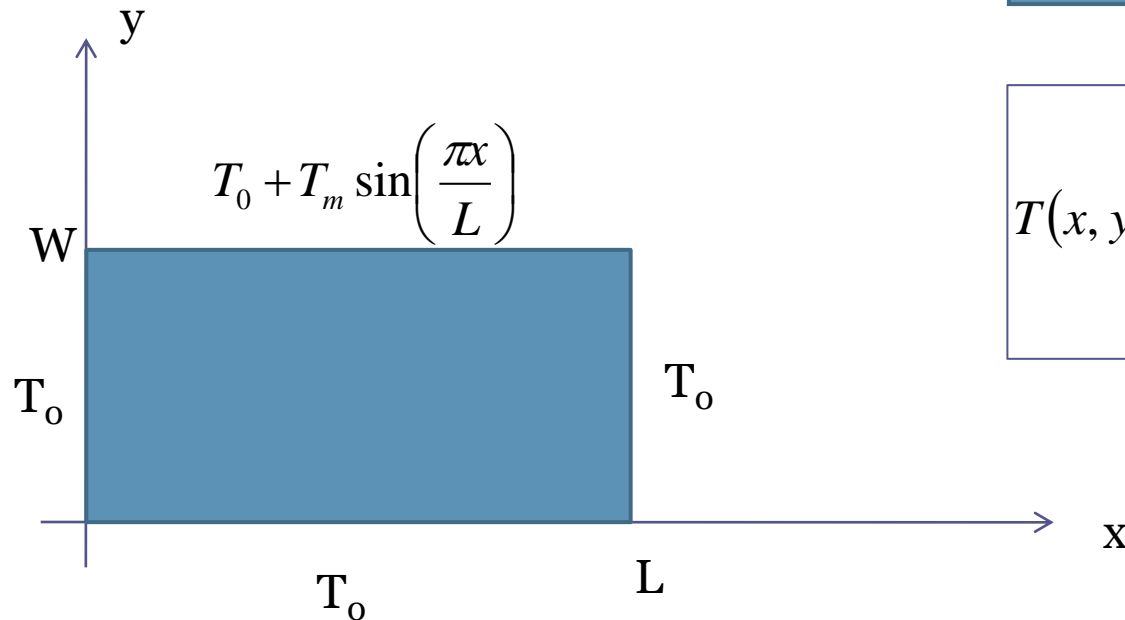


```
g=@(x)[1 x x^2;0 1 x;0 0 1];
```

Una AF può impiegare variabili presenti nel workspace

Esempio

Scrivere una AF che implementi la distribuzione di temperatura a regime in una sezione di una sbarra metallica a sezione rettangolare di lunghezza infinita con temperatura nel bordo imposta dall'esterno.



$$T(x, y) = T_0 + T_m \sin\left(\frac{\pi x}{L}\right) \frac{\sinh\left(\frac{\pi y}{L}\right)}{\sinh\left(\frac{\pi W}{L}\right)}$$

$$T(x, y) = T_0 + T_m \sin\left(\frac{\pi x}{L}\right) \frac{\sinh\left(\frac{\pi y}{L}\right)}{\sinh\left(\frac{\pi W}{L}\right)}$$

$$W = 0.6;$$

$$L = 0.6;$$

$$T_0 = 25;$$

$$T_m = 10;$$

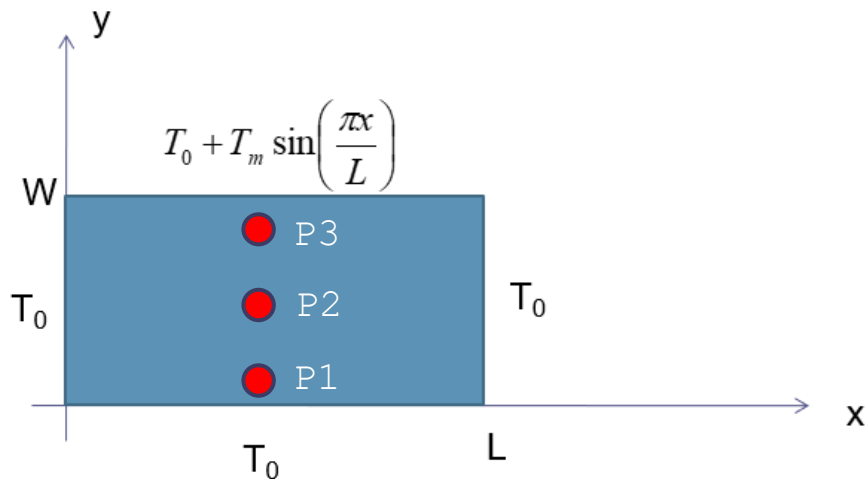
$$f = @(x, y) (T_0 + T_m * \sin(\pi * x / L) * \sinh(\pi * y / L) / \sinh(\pi * W / L));$$

$$P1 = f(0.3, 0.01)$$

$$P2 = f(0.3, 0.3)$$

$$P3 = f(0.3, 0.59)$$

Usiamo la AF per valutare il valore della temperatura in tre punti P1, P2 e P3 della sezione



```
P1 =
    25.0454

P2 =
    26.9927

P3 =
    34.4879
```

```
W=0.6;
L=0.6;
T0=25;
Tm=10;
```

```
f=@(x,y) (T0+Tm*sin(pi*x/L)*sinh(pi*y/L)/sinh(pi*W/L));
```

`x=linspace(0,L,50);` Utilizzo della AF con input vettoriale. Il vettore `x`
`P=f(x,W/2);` contiene le coordinate orizzontali di 50 punti equispaziati
lungo la linea B che divide la sezione a metà

```
plot(x,P),grid
title('Temperatura lungo B')
```

Grafico della temperatura lungo la linea B

