

# Controlli automatici

## Esercitazione n. 1

Introduzione alla simulazione di sistemi di controllo  
in ambiente Matlab-Simulink

## **Cosa vedremo:**

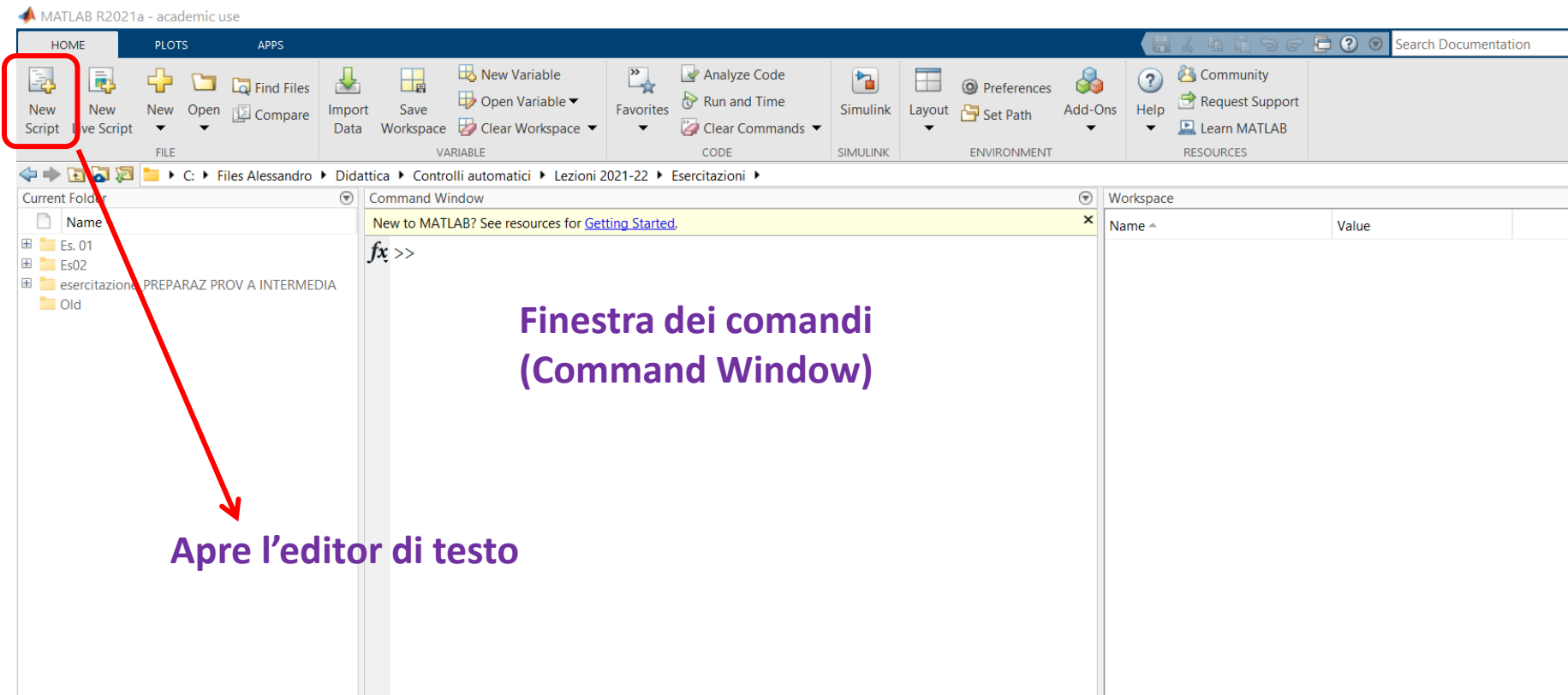
Cenni sulla rappresentazione e analisi di sistemi dinamici in ambiente Matlab (determinazione delle radici di un polinomio, tracciamento del luogo delle radici)

Simulazione dinamica di sistemi LTI a ciclo aperto

Simulazione dinamica di sistemi di controllo LTI in retroazione

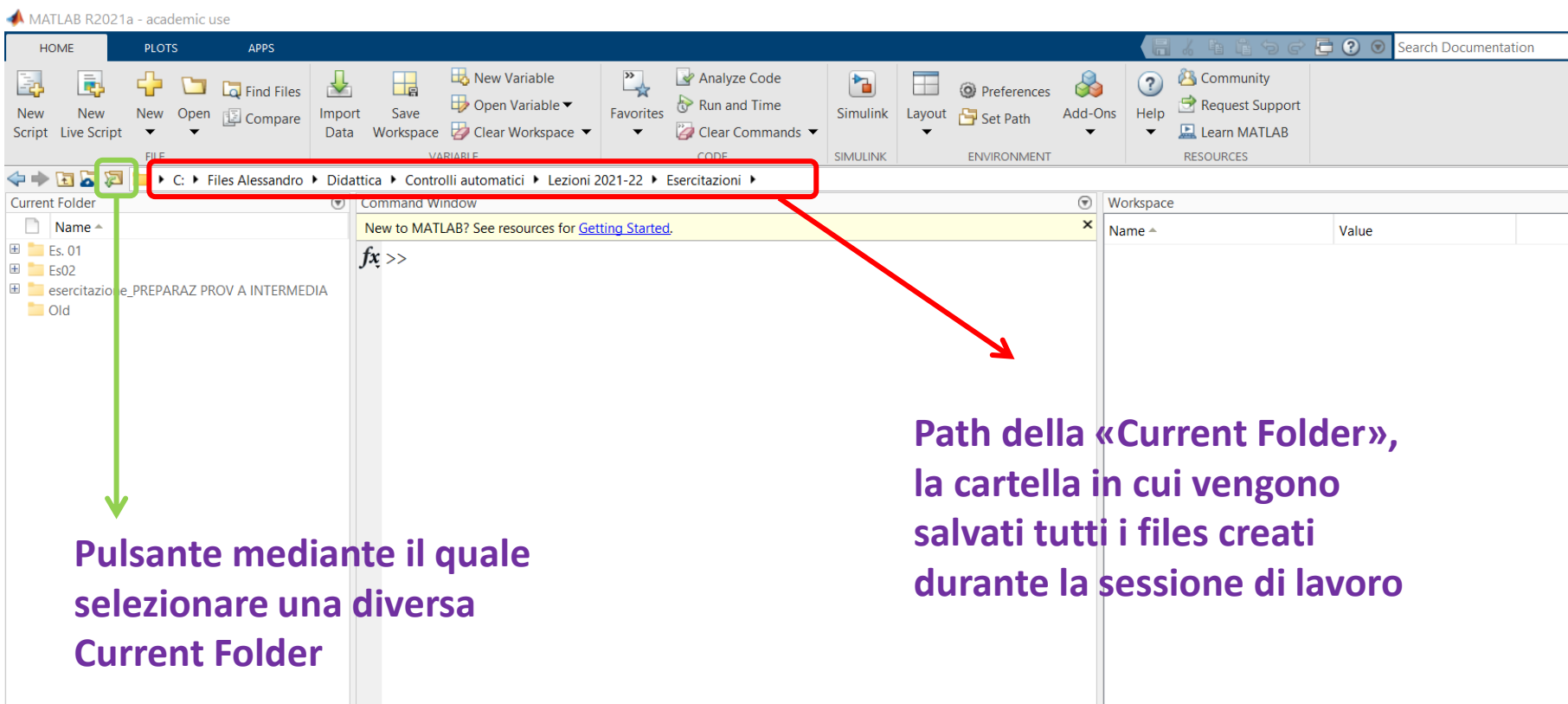
Cenni sulla rappresentazione e analisi di sistemi dinamici in ambiente Matlab  
(determinazione delle radici di un polinomio, tracciamento del luogo delle radici)

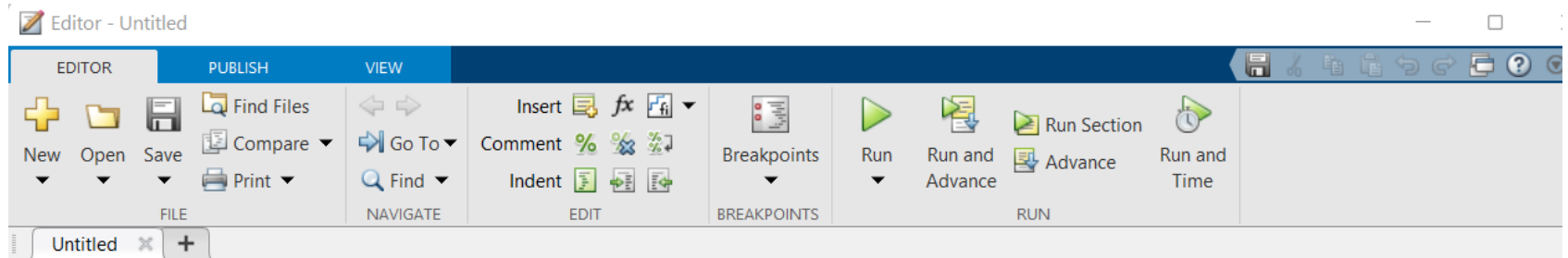
## Si apra Matlab (versione R2021a)



Cenni sulla rappresentazione e analisi di sistemi dinamici in ambiente Matlab  
(determinazione delle radici di un polinomio, tracciamento del luogo delle radici)

## Si apra Matlab (versione R2021a)





**Editor di testo.**

**Utilizzeremo questa interfaccia per scrivere i nostri programmi Matlab (estensione .m, «script») mediante i quali definire polinomi e sistemi dinamici ed effettuare determinate tipologie di analisi**

## Rappresentazione e analisi di polinomi

Rappresentazione di polinomi per mezzo di **vettori**

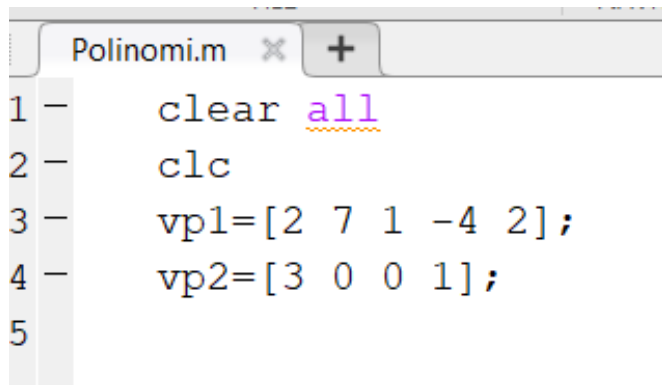
$$p1(x) = 2x^4 + 7x^3 + x^2 - 4x + 2 \longrightarrow vp1 = [2 \quad 7 \quad 1 \quad -4 \quad 2]$$

$$p2(x) = 3x^3 + 1 \longrightarrow vp2 = [3 \quad 0 \quad 0 \quad 1]$$

In ambiente Matlab, un polinomio di grado  $n$  è rappresentato mediante un vettore di dimensione  $n+1$  che contiene tutti i coefficienti del polinomio partendo da quello del termine di grado più elevato fino a giungere al termine noto.

## Rappresentazione e analisi di polinomi

Nell'editor di testo creiamo lo Script «polinomi.m» all'interno del quale, come operazione preliminare, definiamo i due vettori `vp1` e `vp2` associati ai polinomi  $p1(x)$  e  $p2(x)$  riportati nella slide precedente.



```
1 - clear all
2 - clc
3 - vp1=[2 7 1 -4 2];
4 - vp2=[3 0 0 1];
5
```

Un vettore in ambiente Matlab può essere definito come mostrato nell'esempio a sinistra, con gli elementi racchiusi fra parentesi quadre e separati da spazi (o indifferentemente da virgole).

L'istruzione `clear all` cancella dalla memoria del programma tutte le variabili pre-esistenti. L'istruzione `clc` «ripulisce» la Command Window.

## Calcolo delle radici di un polinomio

Funzione “**roots**”

Utile per determinare poli e zeri delle funzioni di trasferimento

Deve essere passato, come argomento di ingresso alla funzione `roots`, il vettore associato al polinomio. Completiamo lo Script con le istruzioni per il calcolo delle radici dei polinomi  $p_1(x)$  e  $p_2(x)$

```
Polinomi.m  x  +
1 - clear all
2 - clc
3 - vp1=[2 7 1 -4 2];
4 - vp2=[3 0 0 1];
5
6 - radicip1=roots(vp1)
7 - radicip2=roots(vp2)
8
```



```
Command Window

radicip1 =

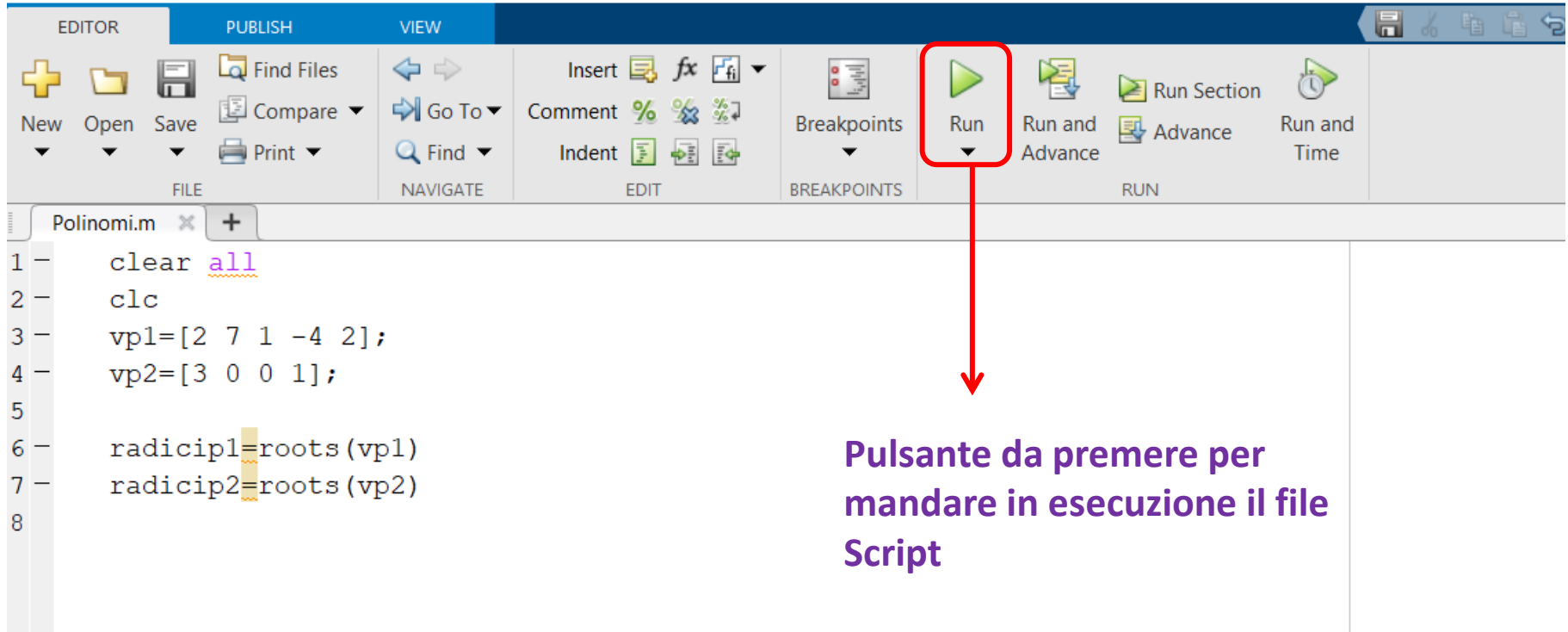
    -3.0962 + 0.0000i
    -1.2285 + 0.0000i
     0.4124 + 0.3047i
     0.4124 - 0.3047i

radicip2 =

    -0.6934 + 0.0000i
     0.3467 + 0.6005i
     0.3467 - 0.6005i
```

Se si omette di inserire il punto e virgola al termine della istruzione (come fatto nelle righe 6 e 7), dopo avere mandato in esecuzione il file script (v. prossima slide) il risultato viene stampato nella Command Window (v. finestra a destra)





The image shows the MATLAB Editor interface. The top menu bar includes EDITOR, PUBLISH, and VIEW. Below the menu bar are several toolbars: FILE (New, Open, Save, Compare, Print), NAVIGATE (Go To, Find), EDIT (Insert, Comment, Indent), BREAKPOINTS (Breakpoints), and RUN (Run, Run and Advance, Run Section, Advance, Run and Time). The Run button, represented by a green play icon, is highlighted with a red box. A red arrow points from the Run button to the text below. The script file Polinomi.m is open in the editor, showing the following code:

```
1 - clear all
2 - clc
3 - vp1=[2 7 1 -4 2];
4 - vp2=[3 0 0 1];
5
6 - radicip1=roots(vp1)
7 - radicip2=roots(vp2)
8
```

**Pulsante da premere per  
mandare in esecuzione il file  
Script**

## Prodotto tra 2 polinomi

### Funzione “conv”

Devono essere passati, come argomenti di ingresso alla funzione conv, i due vettori associati ai polinomi da moltiplicare. La funzione conv restituisce il vettore associato al polinomio prodotto.

Es. Determinare il polinomio  $p_3(x) = (x + 2)(2x^2 + x + 3)$

```
clear all, clc  
p1=[1 2];  
p2=[2 1 3];  
p3=conv(p1,p2)
```



Command Window

```
p3 =  
  
     2     5     5     6
```

Si verifica con semplici calcoli algebrici come il polinomio  $p_3(x)$  abbia la seguente forma

$$p_3(x) = 2x^3 + 5x^2 + 5x + 6$$

## Polinomio di radici assegnate $z_1, z_2, \dots, z_n$

Funzione “**Poly**”

$$p3 = \text{poly}([z_1 \quad z_2 \quad \dots \quad z_n]) \quad \Rightarrow \quad p3(x) = (x - z_1)(x - z_2) \dots (x - z_n)$$

Deve essere passato, come argomento di ingresso alla funzione `conv`, il vettore che contiene le radici assegnate  $z_1, z_2, \dots, z_n$ .

La funzione `poly` restituisce il vettore associato al polinomio  $p_3$ .

Es. Determinare il polinomio  $P$  le cui radici sono  $z_1 = -1, z_2 = -2, z_3 = -3$ .

$$P(x) = (x + 1)(x + 2)(x + 3)$$

```
clear all, clc  
P=poly([-1 -2 -3])
```



```
P =  
1    6   11    6
```

Soluzione:  $P(x) = x^3 + 6x^2 + 11x + 6$

## Risposta al gradino unitario

Funzione “step”

Creiamo un grafico della risposta al gradino unitario del sistema dinamico avente come FdT

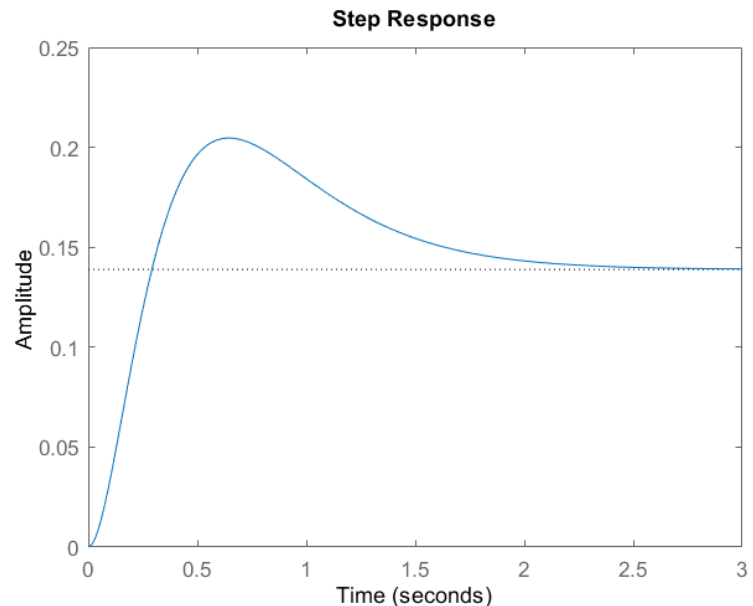
$$F(s) = \frac{100(s + 1)}{(s + 3)(s + 4)(s + 6)}$$

Debbono essere costruiti, e passati come argomenti di ingresso alla funzione step, i vettori associati ai polinomi a numeratore e denominatore della FdT:

```
clear all, clc
```

```
num=10*[1 1];  
den=poly([-3 -4 -6]);
```

```
step(num,den)
```



Per scegliere la **durata temporale** della simulazione:

```
step(num,den,Tfinal)
```

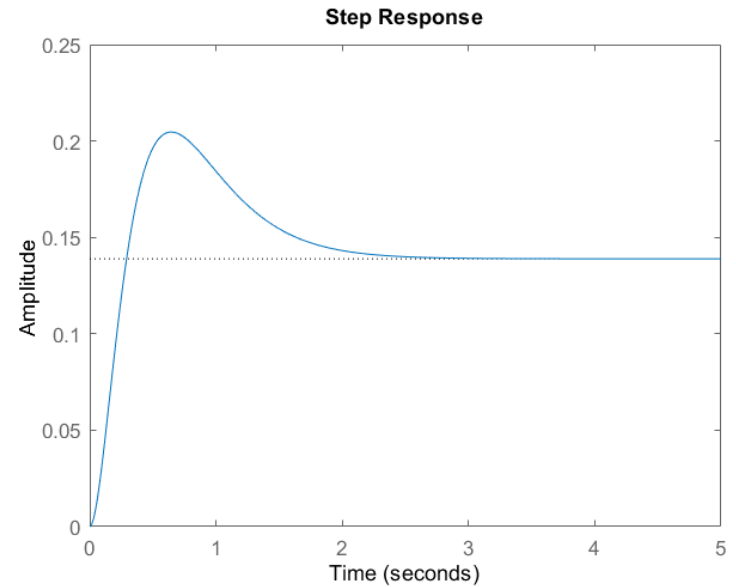
Esempio:

```
clear all, clc
```

```
num=10*[1 1];
```

```
den=poly([-3 -4 -6]);
```

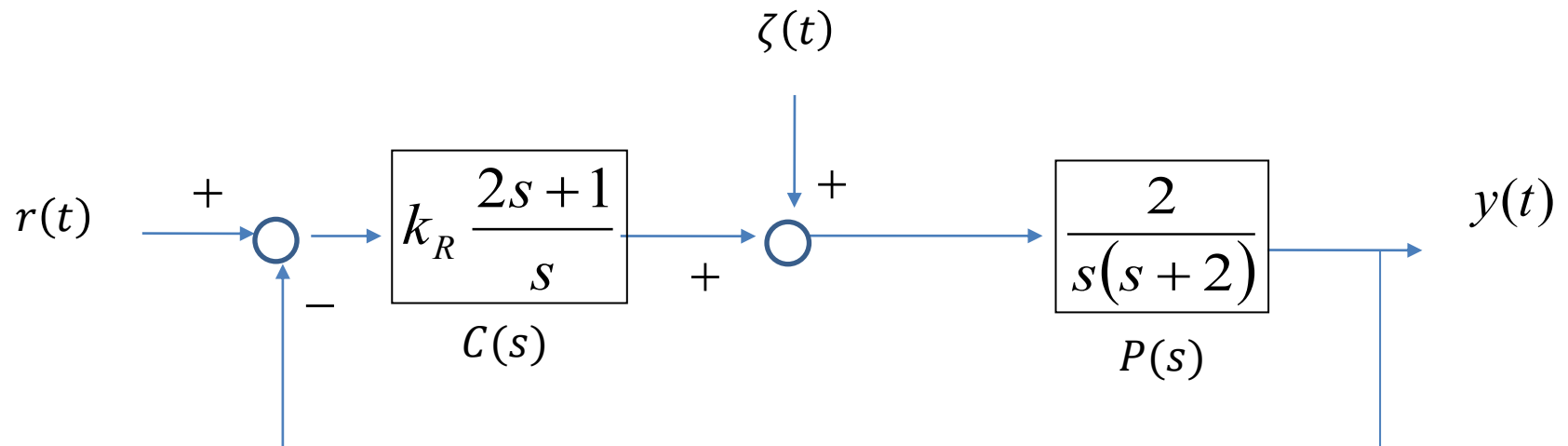
```
step(num,den,5)
```



## Tracciamento del luogo delle radici

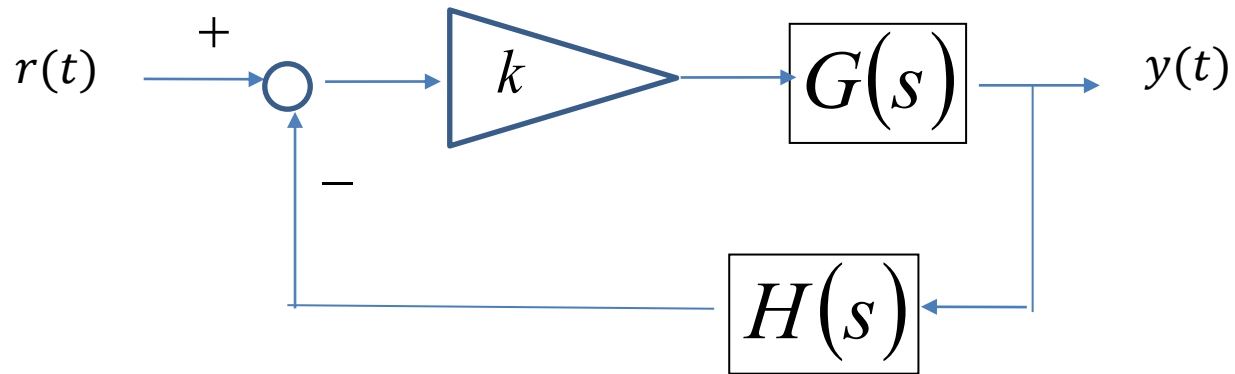
Funzione “rlocus”

Esempio

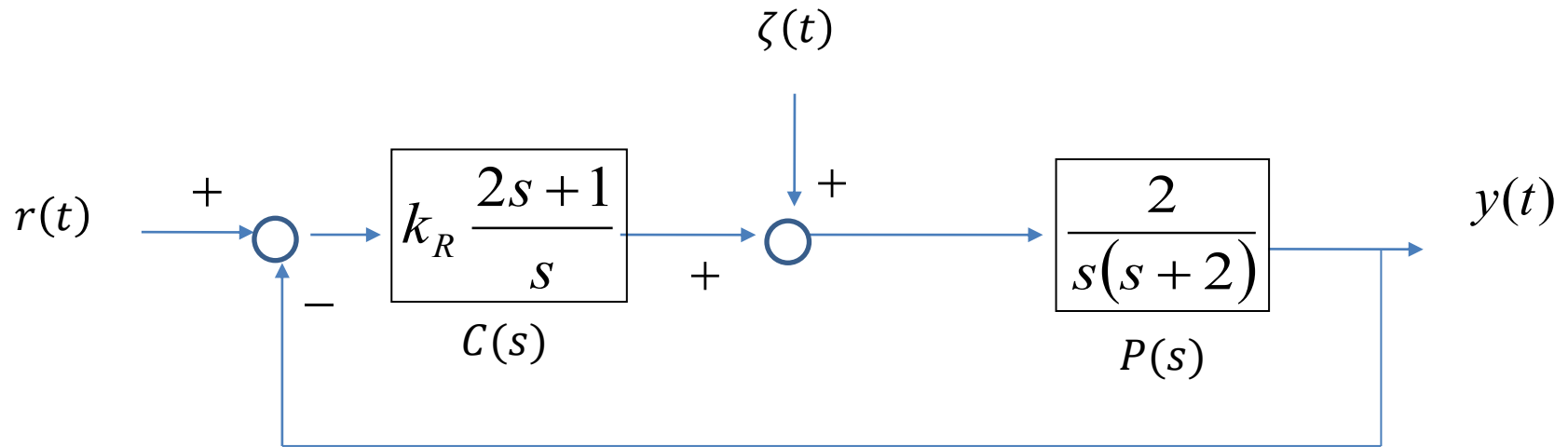


Desideriamo analizzare numericamente la stabilità a ciclo chiuso al variare del guadagno  $k_R$ . Tracciamo il LdR e interpretiamolo in tal senso.

Riconduciamo lo schema alla forma «standard» per il tracciamento del LdR

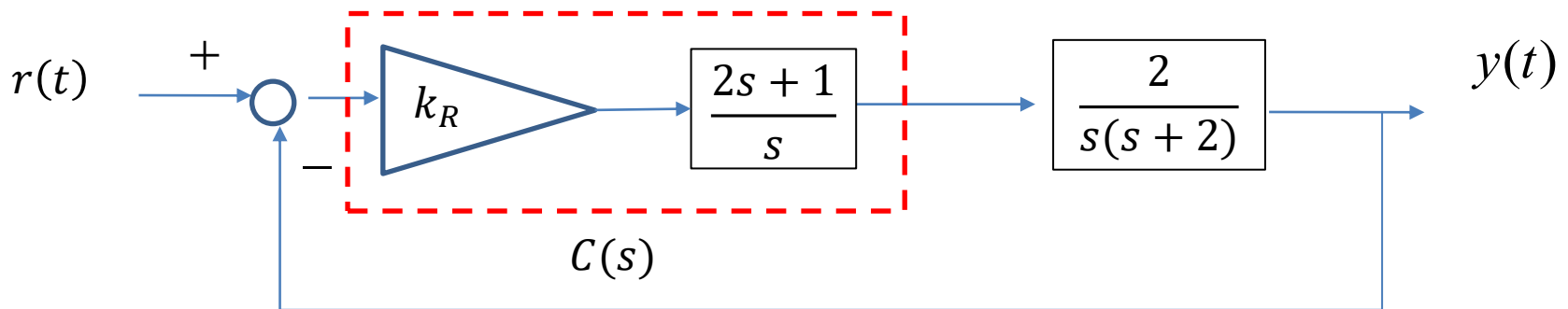


$$L(s) = G(s)H(s)$$

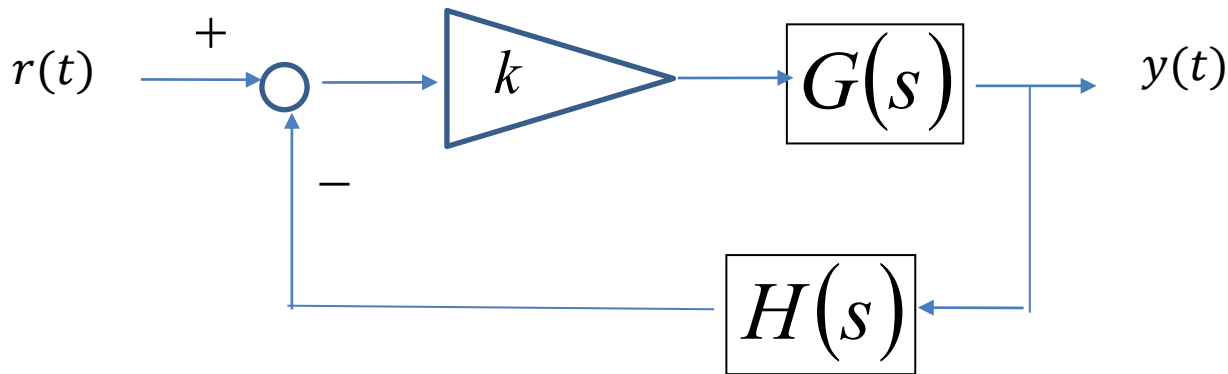
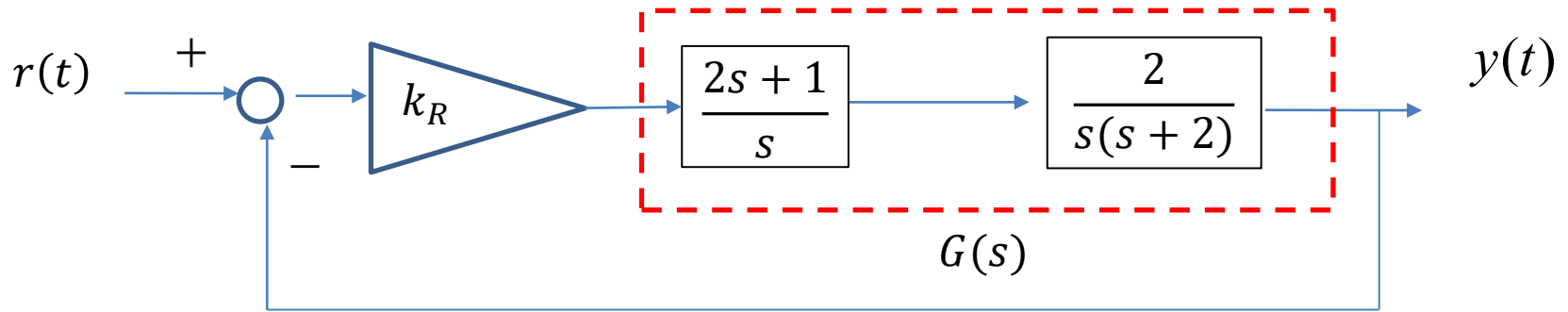


Possiamo rimuovere dallo schema il disturbo  $\zeta(t)$  in quanto la sua presenza o meno non interferisce con gli scopi della analisi.

Scorporiamo il guadagno  $k_R$  dalla parte dinamica del controllore  $\frac{2s+1}{s}$







Confrontando lo schema in alto con lo schema standard si ricava:

$$G(s) = \frac{2s+1}{s} \cdot \frac{2}{s(s+2)} = \frac{4s+2}{s^2(s+2)}$$

$$H(s) = 1$$

$$L(s) = G(s)H(s) = \frac{4s+2}{s^2(s+2)}$$

L'espressione **analitica** della  $L(s)$ , che abbiamo appena ricavato, è l'unica cosa che serve per il tracciamento del LdR.

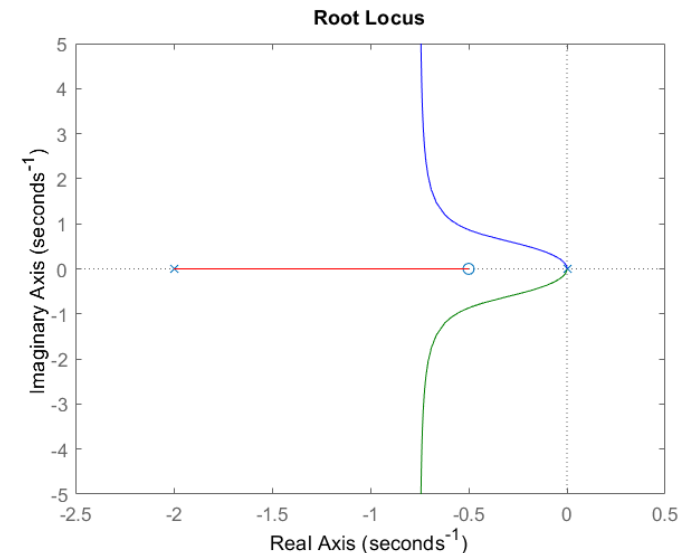
Per il tracciamento mediante Matlab del luogo delle radici si impiega la funzione **rlocus**, alla quale devono essere passati come argomenti di ingresso i vettori associati ai polinomi a numeratore e denominatore della  $L(s)$

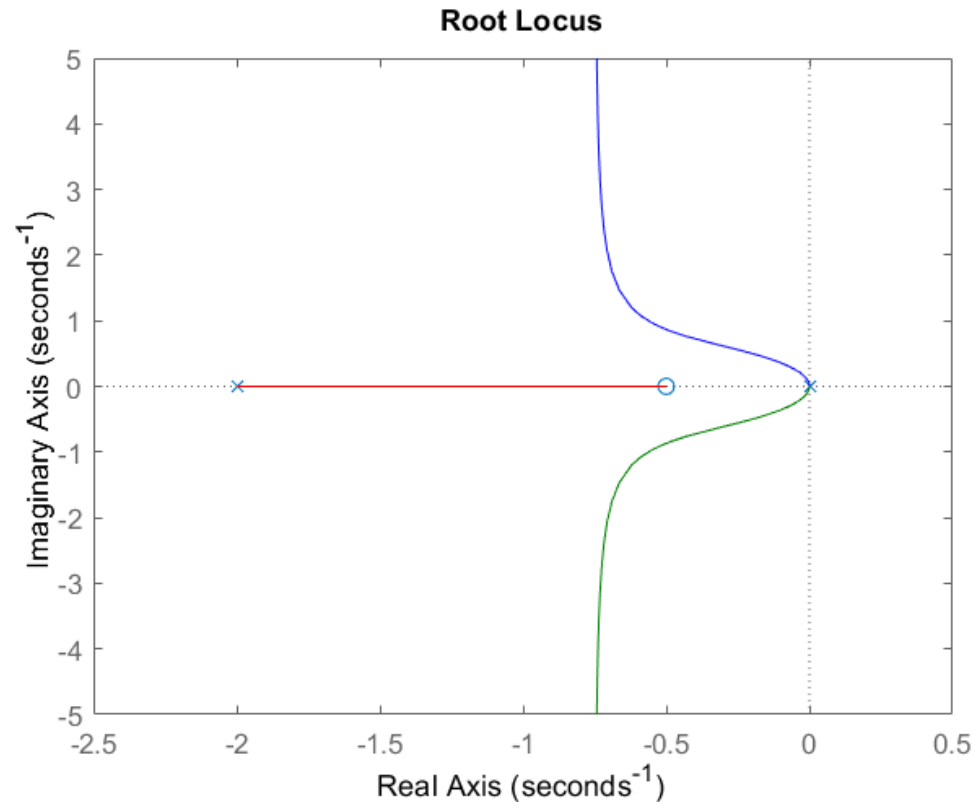
Sintassi generica:

**rlocus(numL,denL)**

$$L(s) = G(s)H(s) = \frac{4s + 2}{s^2(s + 2)}$$

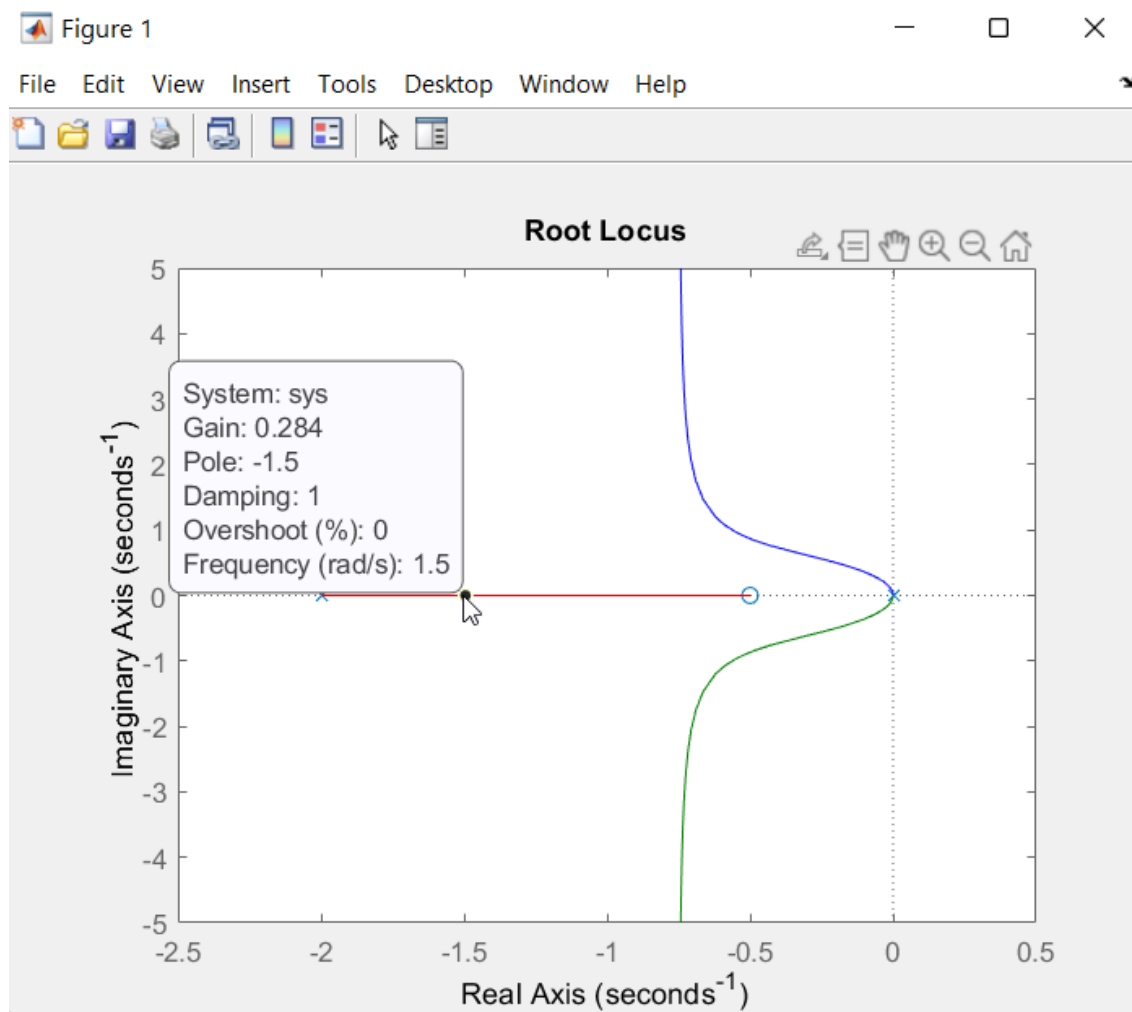
```
LdR01.m x +
1 - numL=[4 2];
2 - denL=poly([0 0 -2]);
3 - %sintassi alternativa: denL=[1 2 0 0];
4 - rlocus(numL,denL)
```





I 3 rami sono tracciati in colore diverso. I poli e gli zeri della  $L(s)$  sono marcati con delle crocette e con dei pallini. Non viene evidenziato il verso di percorrenza, dei rami che è peraltro banalmente deducibile.

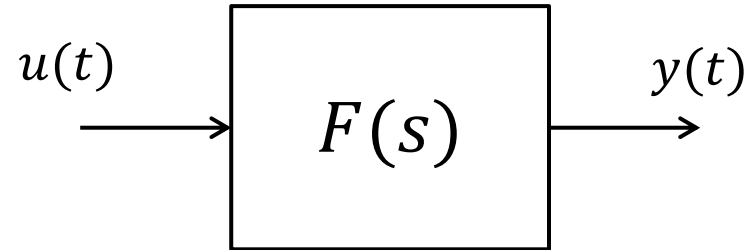
Si nota come tutti e tre i rami siano interamente contenuti nel semipiano sinistro, da cui deduciamo come il sistema di controllo sia asintoticamente stabile a ciclo chiuso per qualunque valore di  $K_R$



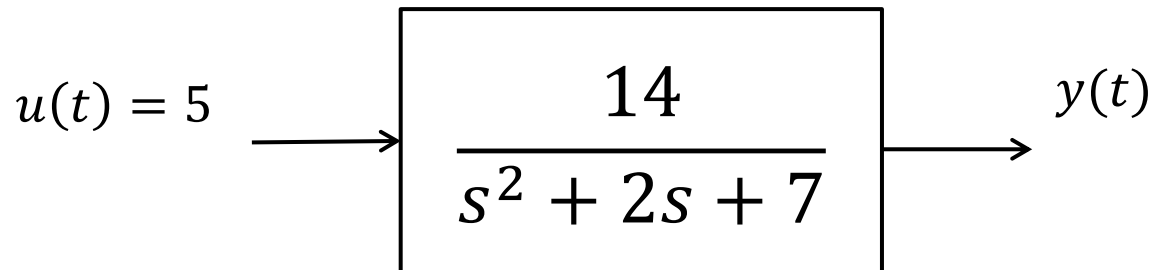
**Scorrendo con la freccia del mouse sopra i rami** è possibile valutare il valore del guadagno associato a ciascun punto del luogo delle radici (funzionalità utile, ad esempio, per determinare il valore di guadagno associato ad un punto doppio, o al punto in corrispondenza del quale uno dei rami attraversa l'asse immaginario)

# Simulazione dinamica di sistemi LTI a ciclo aperto mediante Simulink

Impariamo ad eseguire la simulazione dinamica a **ciclo aperto** in cui un sistema dinamico LTI descritto dalla FdT  $F(s)$  viene sottoposto ad un certo segnale di ingresso  $u(t)$ , e si desidera visualizzarne la risposta  $y(t)$



Piu in dettaglio, ci proponiamo di visualizzare la risposta al gradino di ampiezza 5 ( $u(t) = 5$ ) di un processo del secondo ordine



# Realizziamo il modello Simulink

La realizzazione di un modello Simulink per la simulazione dinamica avviene per via grafica, realizzando uno **schema a blocchi** in tutto e per tutto simile a quelli che normalmente disegniamo a lezione per rappresentare i sistemi di controllo.

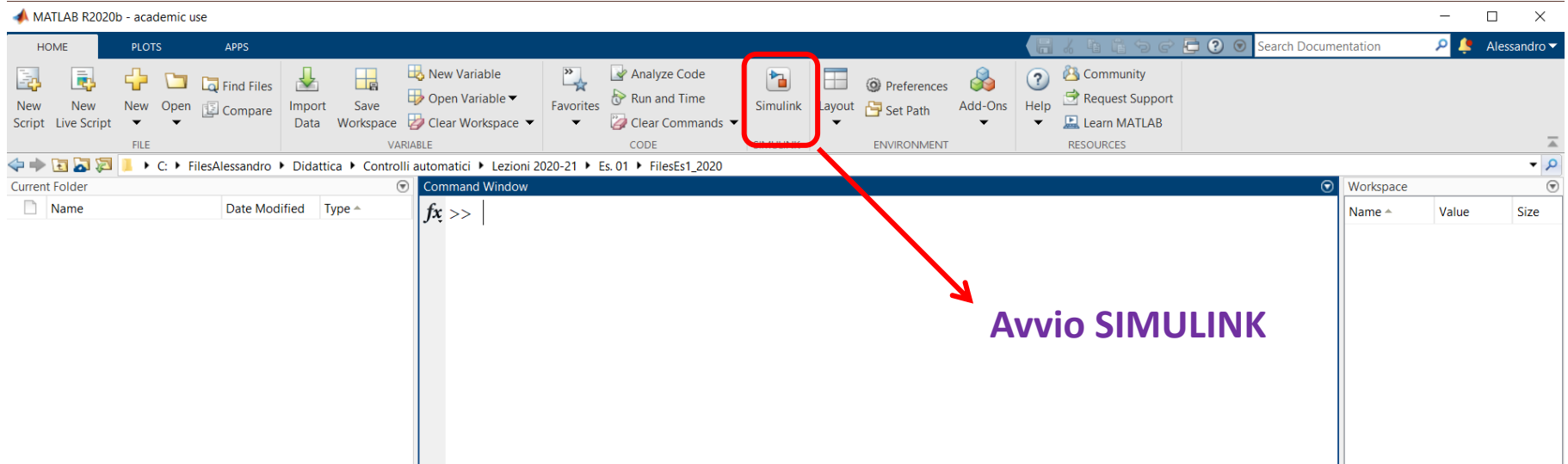
La realizzazione di un modello Simulink avviene attraverso tre fasi:

- si importano in una pagina bianca di lavoro i **blocchi elementari** necessari per la realizzazione dello schema di simulazione
- si **parametrizzano i blocchi** in modo che implementino le funzionalità desiderate
- si **interconnettono tra loro i blocchi** per realizzare lo schema desiderato

Fatto ciò, si può avviare la simulazione e visualizzarne i risultati

**Si apra Matlab**

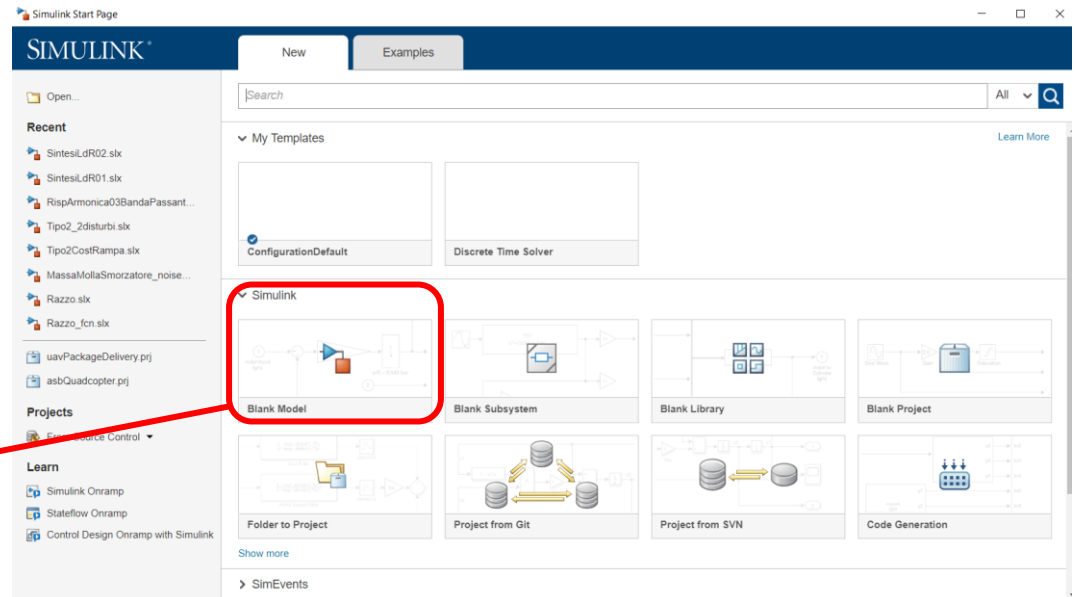
# Finestra di avvio di Matlab (rel. R2020b)



Avvio SIMULINK

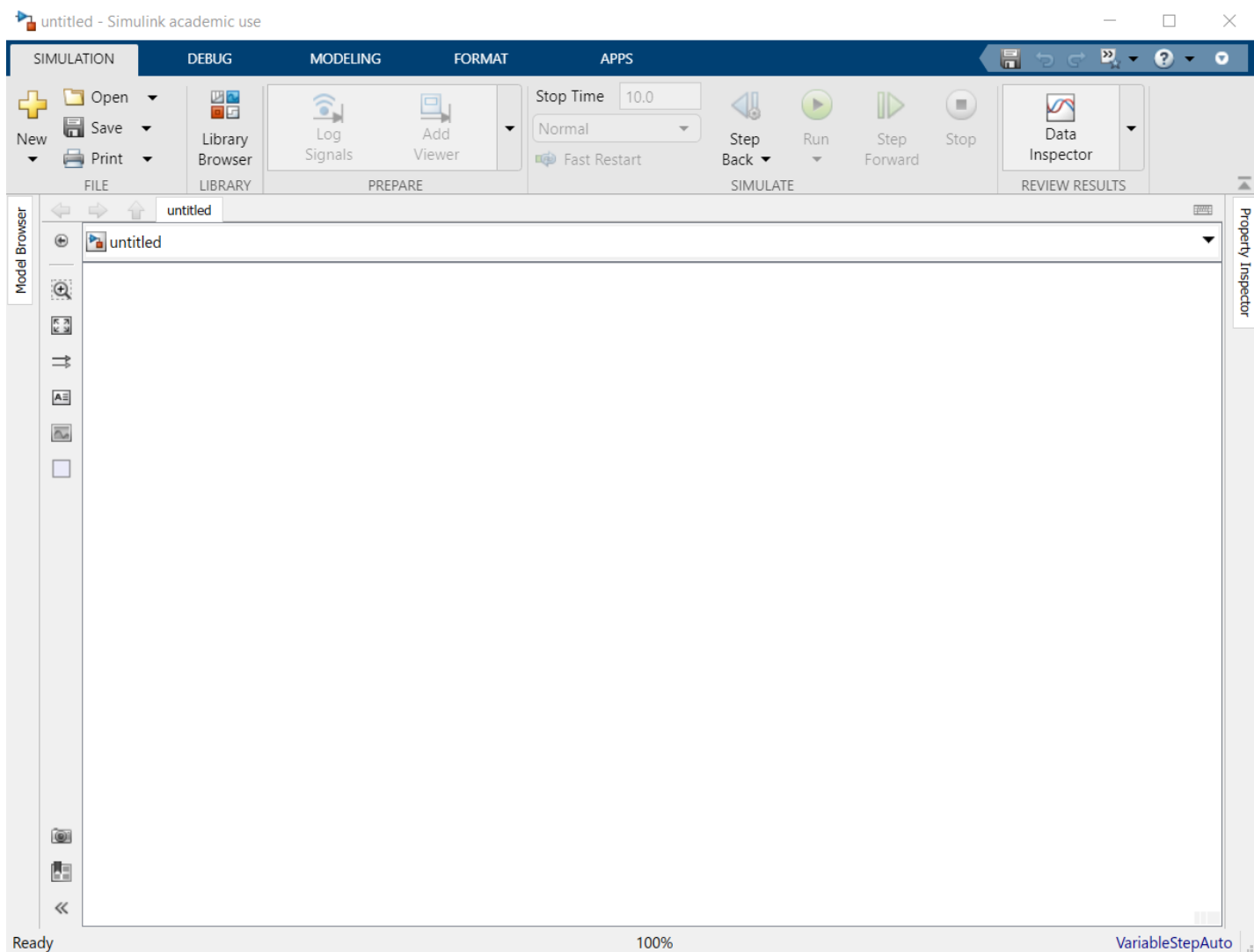
## Finestra di avvio SIMULINK

Dalla finestra di avvio di Matlab,  
apriamo Simulink.  
Si apre la relativa finestra di avvio:



Apertura di un Blank Model

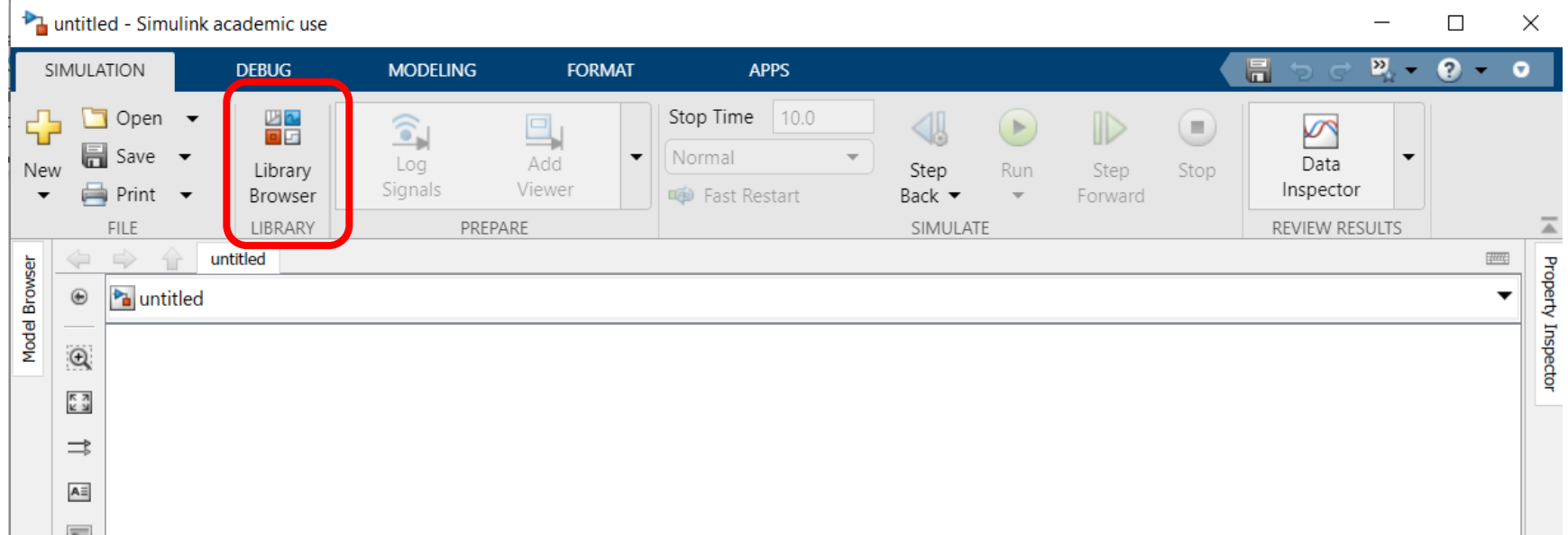
# Modello Simulink in bianco



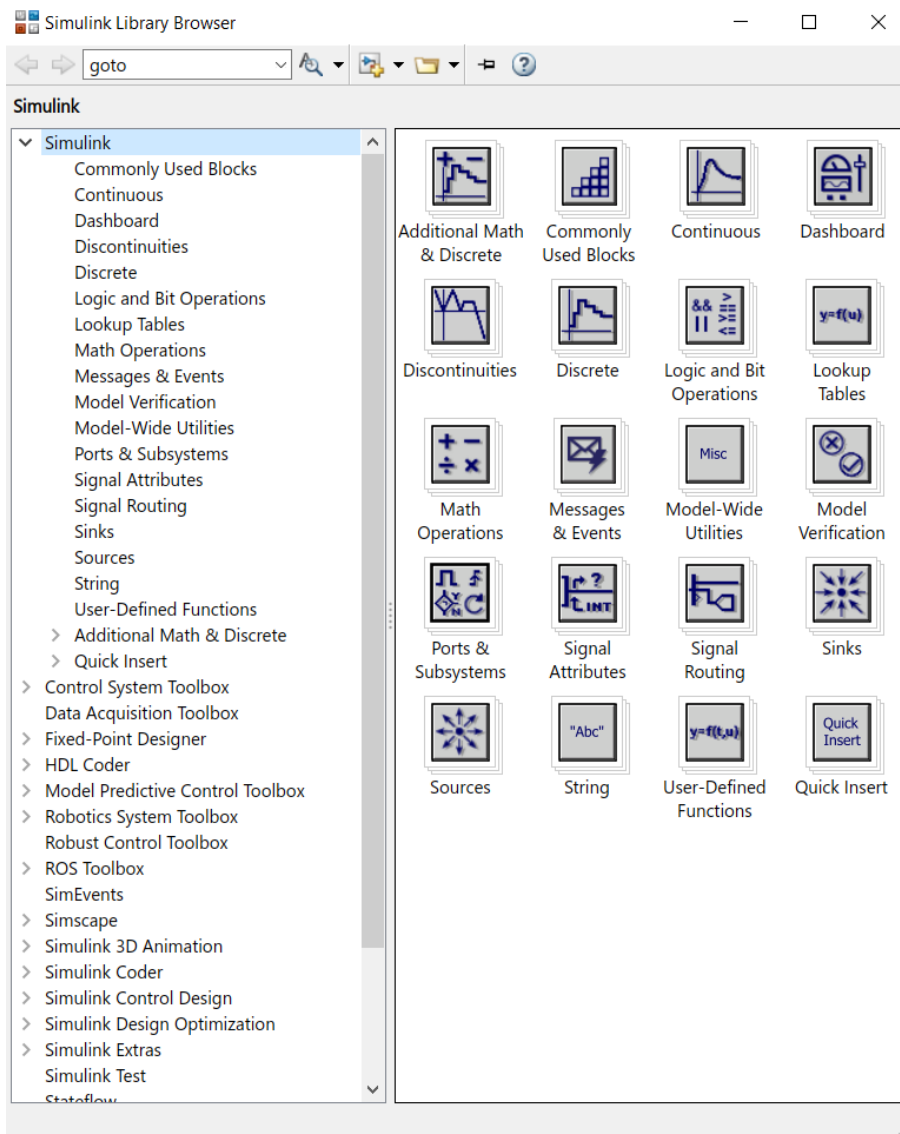


La realizzazione di modelli di simulazione dinamica avviene per via grafica **assemblando fra loro all'interno della pagina di lavoro un certo numero di blocchi Simulink**, in modo da implementare le funzionalità desiderate.

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»)



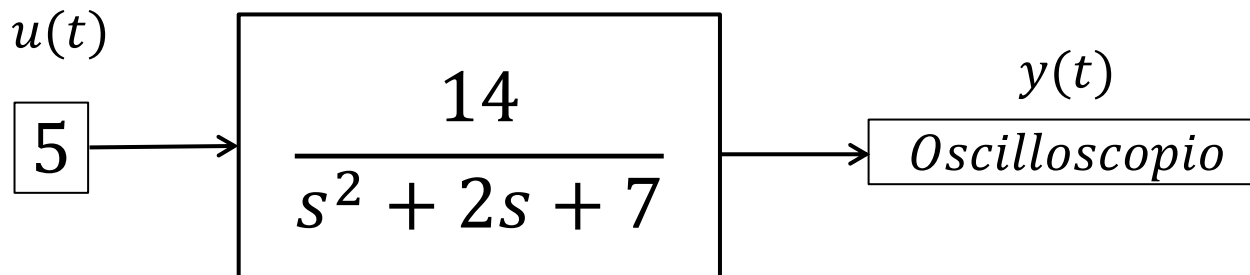
# Simulink Library Browser



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

Per realizzare lo schema a blocchi associato ad una simulazione dinamica a ciclo aperto sono sufficienti **tre blocchi elementari** Simulink:

- 1 un blocco elementare che implementi una FdT
- 2 un blocco elementare che generi il segnale di ingresso
- 3 un blocco elementare «oscilloscopio» che consenta la visualizzazione del segnale di uscita

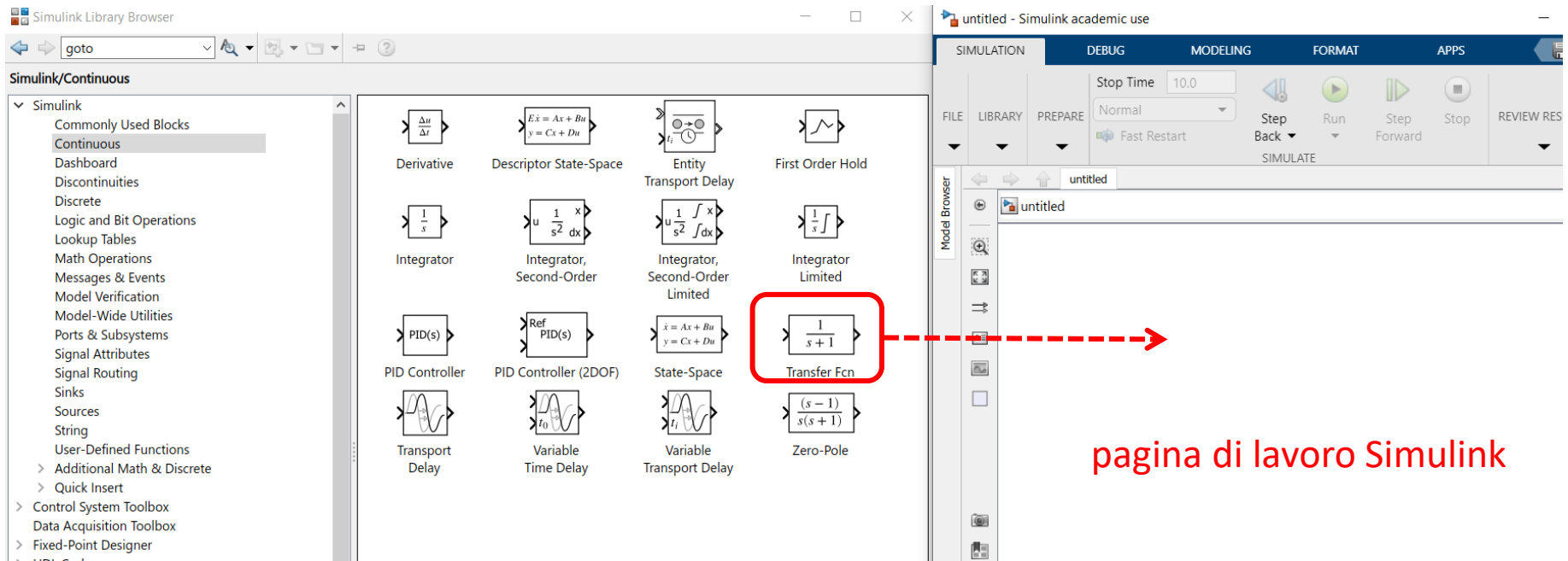


Nella pagina di lavoro Simulink, andremo a “disegnare” una rappresentazione analoga a quella riportata qui sopra

Importiamo nella pagina di lavoro il blocco “Transfer Fcn” (funzione di trasferimento), che si trova nella sotto-libreria “Continuous” della libreria principale “Simulink”

E’ uno dei blocchi (non l’unico) per mezzo dei quali si possono realizzare in ambiente Simulink delle funzioni di trasferimento

L’importazione dei blocchi nella pagina di lavoro si effettua con il mouse mediante drag-and-drop

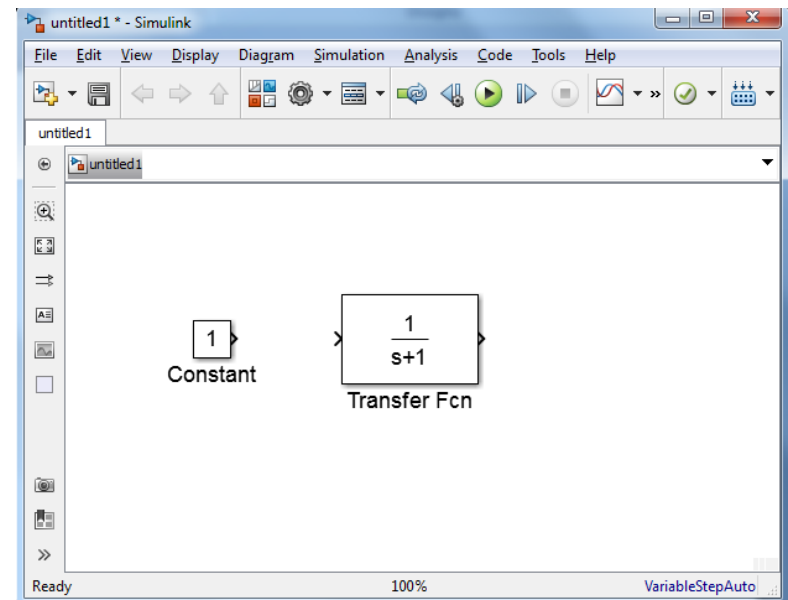


Ora importiamo il blocco necessario per generare il segnale di ingresso costante.

Tale blocco si chiama `Constant`, e si trova nella sotto-libreria “`Sources`” della libreria principale “`Simulink`”, che contiene una ampia varietà di blocchi elementari preposti alla **generazione di segnali**

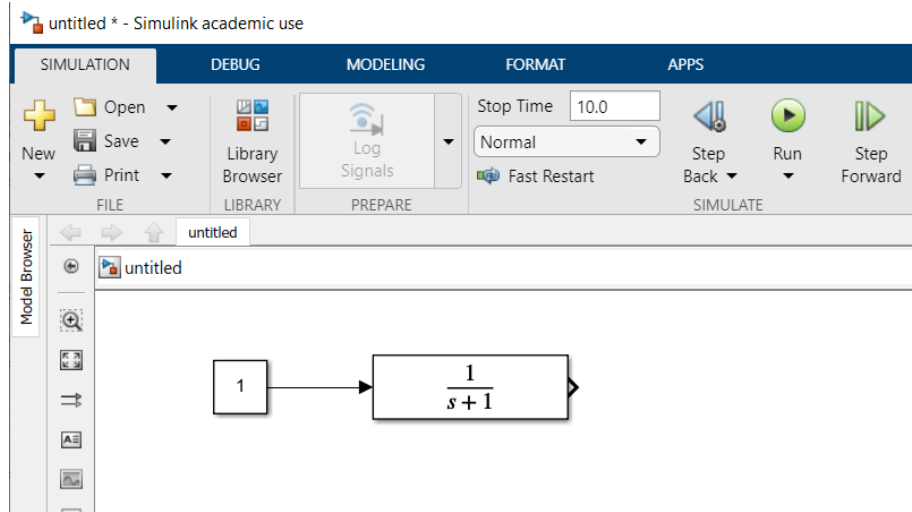
**Disponiamo i blocchi come nella figura a lato**

Si noti che il blocco `Constant` possiede unicamente un terminale di uscita, mentre invece il blocco `Transfer Fcn` possiede un terminale di ingresso ed un terminale di uscita, come è lecito attendersi.



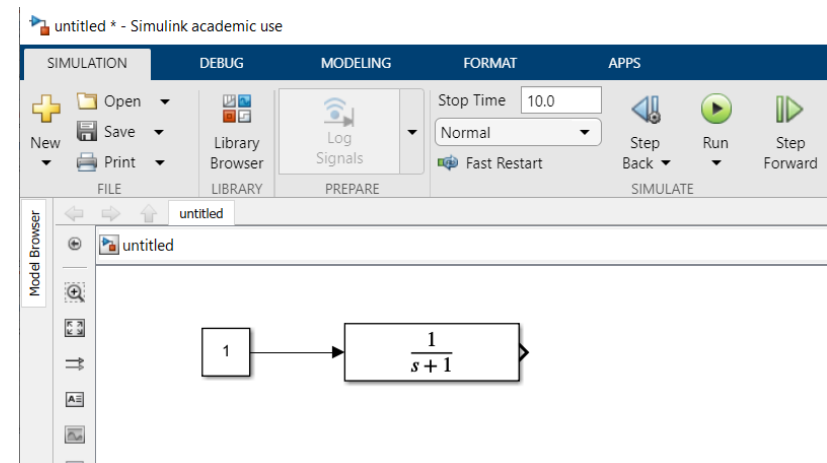
## Interconnettiamo i due blocchi.

Il tracciamento di una connessione si effettua portandosi con il mouse nel punto di inizio della linea di connessione (quindi nel terminale di uscita del blocco Constant), premendo il tasto sinistro, e successivamente portandosi con il mouse - mantenendo premuto il tasto - verso il punto di destinazione, in questo caso il terminale di ingresso del blocco Transfer Fcn.



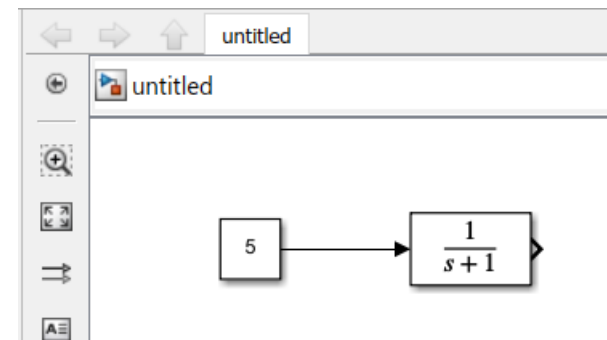
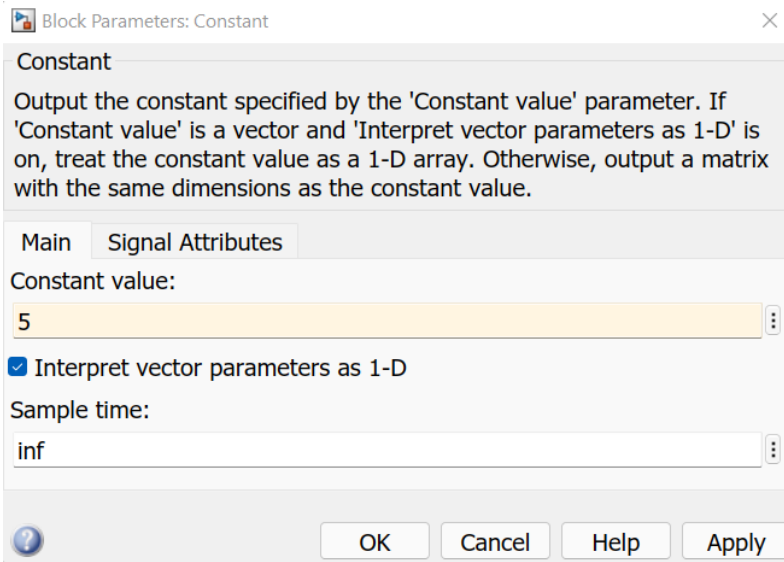
Per effettuare un collegamento tra due blocchi vi è anche una **procedura rapida**. Si deve selezionare il blocco di origine (cliccandovi sopra), e si deve successivamente selezionare il blocco di destinazione con il **tasto ctrl premuto**.

Di default, il blocco Constant è parametrizzato per generare un segnale costante di ampiezza unitaria, mentre il blocco Transfer Fcn è parametrizzato di default con la FdT  $\frac{1}{s+1}$

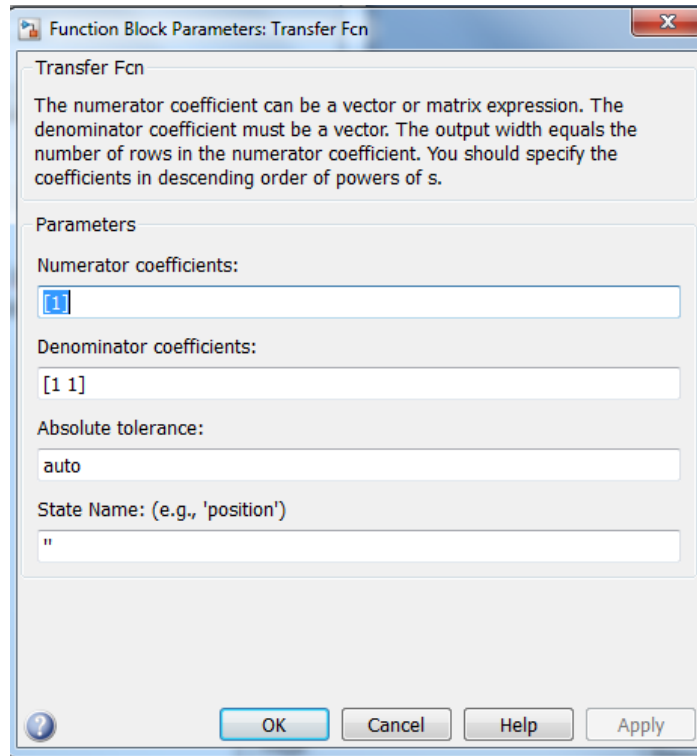


Facendo doppio click su un blocco, si accede alla sua finestra di parametrizzazione.

Il blocco Constant va riconfigurato modificando l'ampiezza del gradino da 1 a 5.



Dobbiamo inoltre configurare il blocco Transfer Fcn.  
Apriamone la finestra di parametrizzazione



The image shows a MATLAB/Simulink dialog box titled "Function Block Parameters: Transfer Fcn". It contains a description of the block's coefficients and a section for parameter entry.

**Transfer Fcn**

The numerator coefficient can be a vector or matrix expression. The denominator coefficient must be a vector. The output width equals the number of rows in the numerator coefficient. You should specify the coefficients in descending order of powers of  $s$ .

**Parameters**

Numerator coefficients:  
[1]

Denominator coefficients:  
[1 1]

Absolute tolerance:  
auto

State Name: (e.g., 'position')  
"

Buttons: ? OK Cancel Help Apply



## Parametrizzazione del blocco Transfer Function

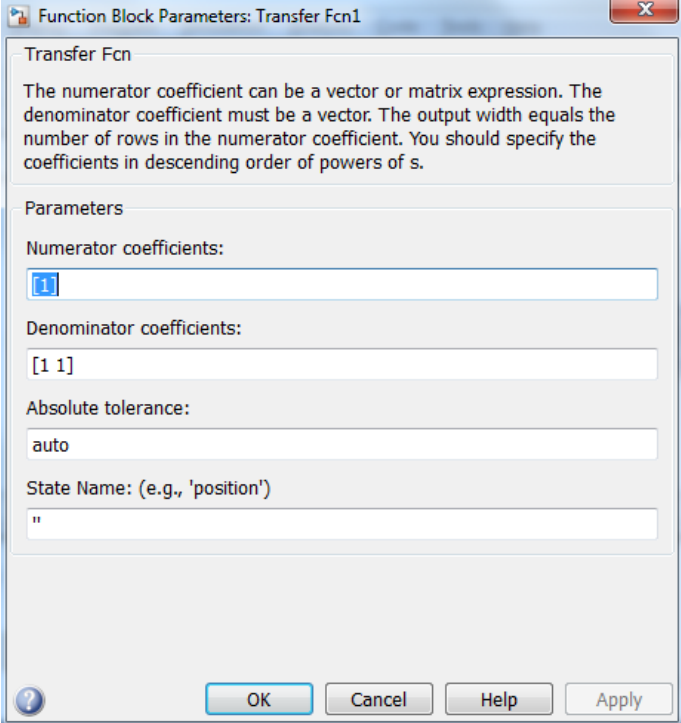
Il blocco deve rappresentare la FdT

$$F(s) = \frac{14}{s^2 + 2s + 7}$$

Si devono specificare i **coefficienti dei polinomi a numeratore e denominatore** della FdT utilizzando la notazione Matlab per la rappresentazione dei polinomi (un vettore che contiene i coefficienti del polinomio **in ordine decrescente** rispetto alle potenze di s)

$$14 \Rightarrow [14]$$

$$s^2 + 2s + 7 \Rightarrow [1 \ 2 \ 7]$$



Altri esempi:

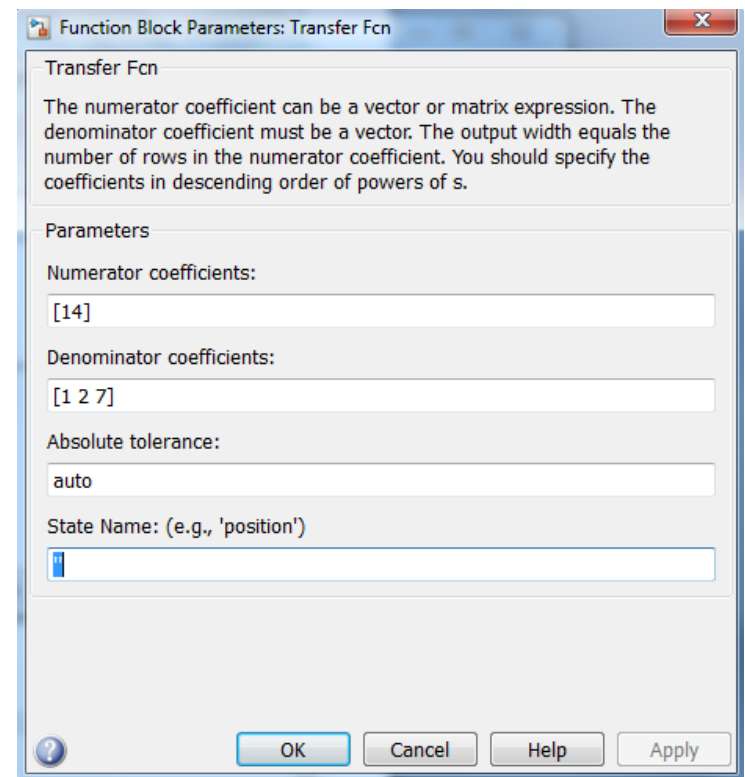
$$s \Rightarrow [1 \ 0]$$

$$s^4 - 2s^3 + 3s + 2 \Rightarrow [1 \ -2 \ 0 \ 3 \ 2]$$

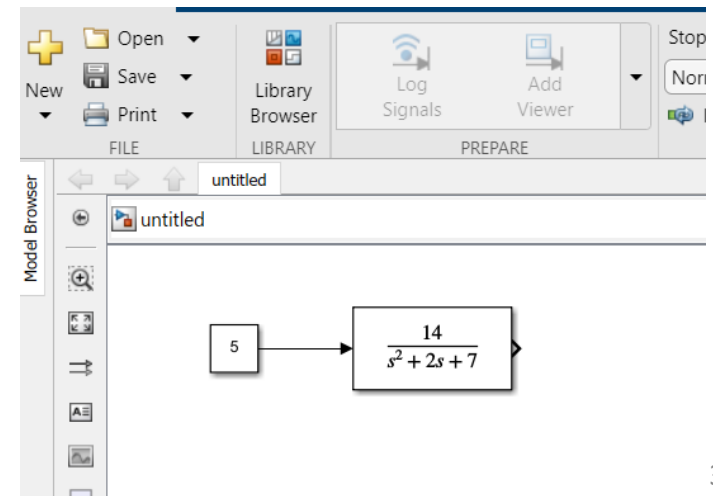
$$s(s + 1) = s^2 + s \Rightarrow [1 \ 1 \ 0]$$

$$s^3 + 4 \Rightarrow [1 \ 0 \ 0 \ 4]$$

Il blocco deve essere pertanto parametrizzato nella seguente maniera.



Dopo aver premuto il tasto OK, l'aspetto del blocco Transfer Fcn nella pagina di lavoro cambia, e visualizza al proprio interno la FdT avente i parametri scelti (eventualmente il blocco deve essere ingrandito affinché la visualizzazione sia corretta)

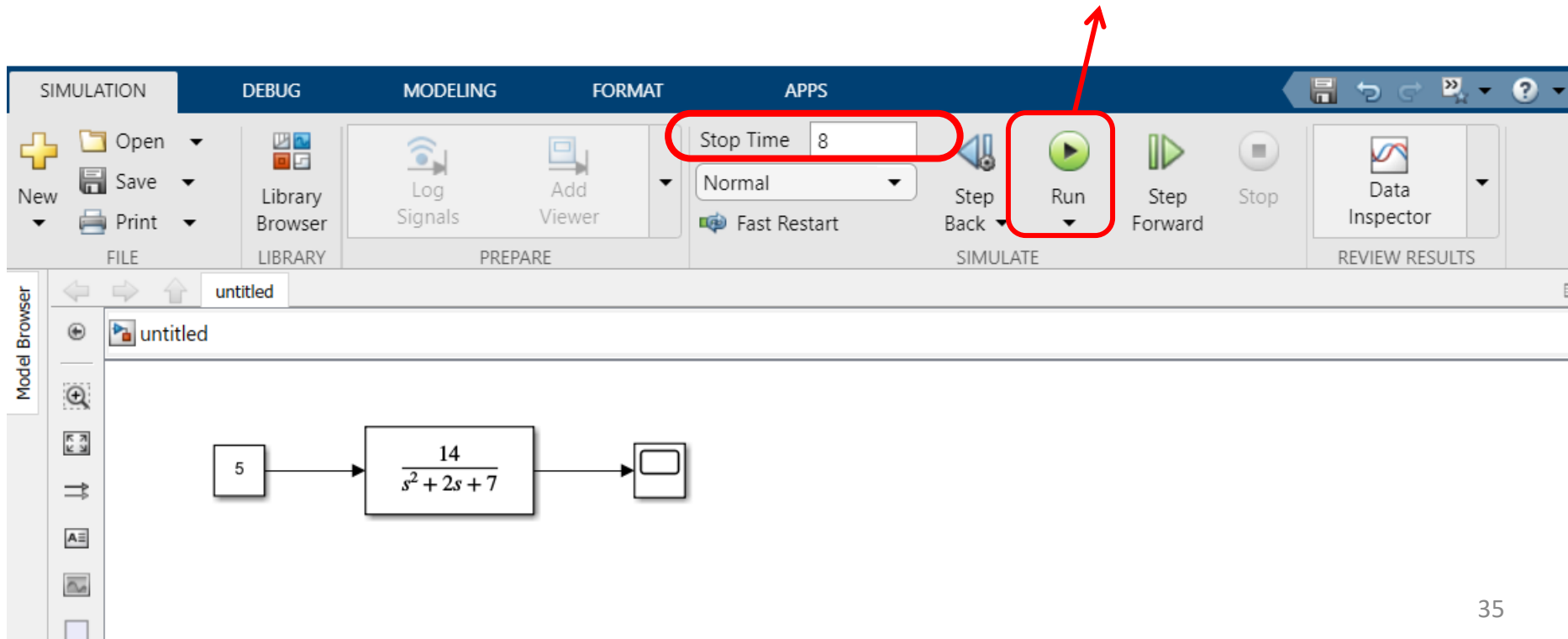


Ora importiamo il blocco “**Scope**” (oscilloscopio), che consente la visualizzazione di un segnale.

Il blocco Scope sta nella sotto-libreria “**Sinks**” della libreria principale “**Simulink**”. Dopo averlo importato nella pagina di lavoro, colleghiamone il terminale di ingresso al terminale di uscita del blocco Transfer Fcn.

Impostare nella casella **Stop Time** la durata della simulazione (il valore di default è 10 secondi, scegliamo di eseguire la simulazione dinamica per **8 secondi**), e cliccare sul **pulsante Run** per avviare la simulazione.

Pulsante «Run»



Prima di fare doppio click sul blocco Scope e visualizzare la risposta, ne valutiamo «su carta» alcuni parametri caratteristici mediante le formule viste a lezione

$$F(s) = \frac{14}{s^2 + 2s + 7}$$

$F(s)$  è asintoticamente stabile in quanto il polinomio caratteristico è di secondo grado ed ha tutti i coefficienti di segno concorde.

Guadagno statico  $\mu = F(0) = 2$

Valore di regime della risposta all'ingresso  $u(t) = 5$   $y_{\infty} = 5 \cdot \mu = 10$

Poli: `poli=roots([1 2 7])`

$$p_{1,2} = -1 \pm 2.449i$$

Command Window

```
poli =  
  
-1.0000 + 2.4495i  
-1.0000 - 2.4495i
```

$$p_{1,2} = -1 \pm 2.449i = a + bj$$

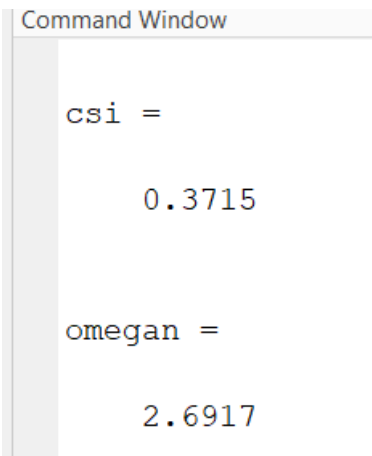
Parte reale:  $a = -1$

Parte immaginaria:  $b = \pm 2.449$

Smorzamento  $\xi = -\frac{a}{\sqrt{a^2+b^2}} = 0.37$

Pulsazione naturale  $\omega_n = \sqrt{a^2 + b^2} = 2.69$

```
clc
a=-1;
b=2.499;
csi=-a/sqrt(a^2+b^2)
omegan=sqrt(a^2+b^2)
```



Command Window

```
csi =
    0.3715

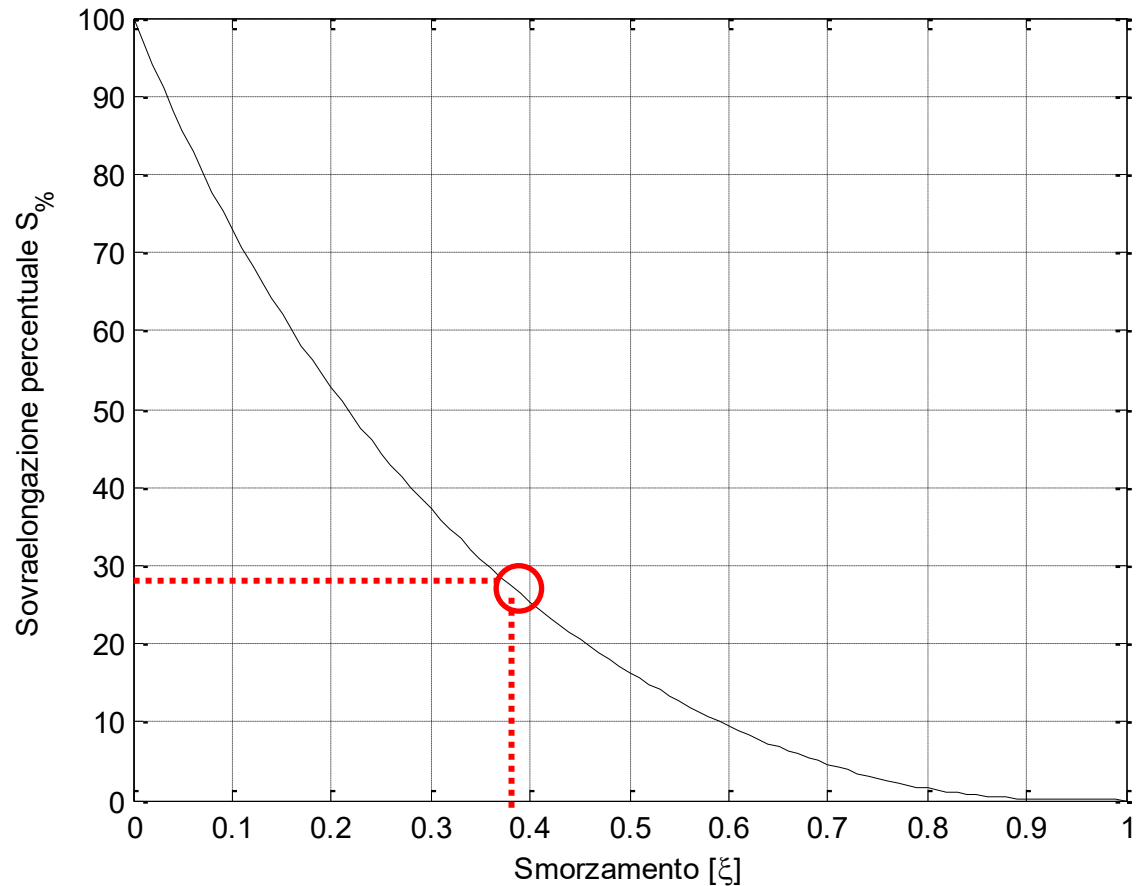
omegan =
    2.6917
```

## Sovraelongazione percentuale

$$S_{\%} = 100e^{-\frac{\xi\pi}{\sqrt{1-\xi^2}}}$$

$$\xi \approx 0.37$$

$$S_{\%} \approx 28$$



Il valore massimo dell'uscita durante il transitorio sarà pertanto

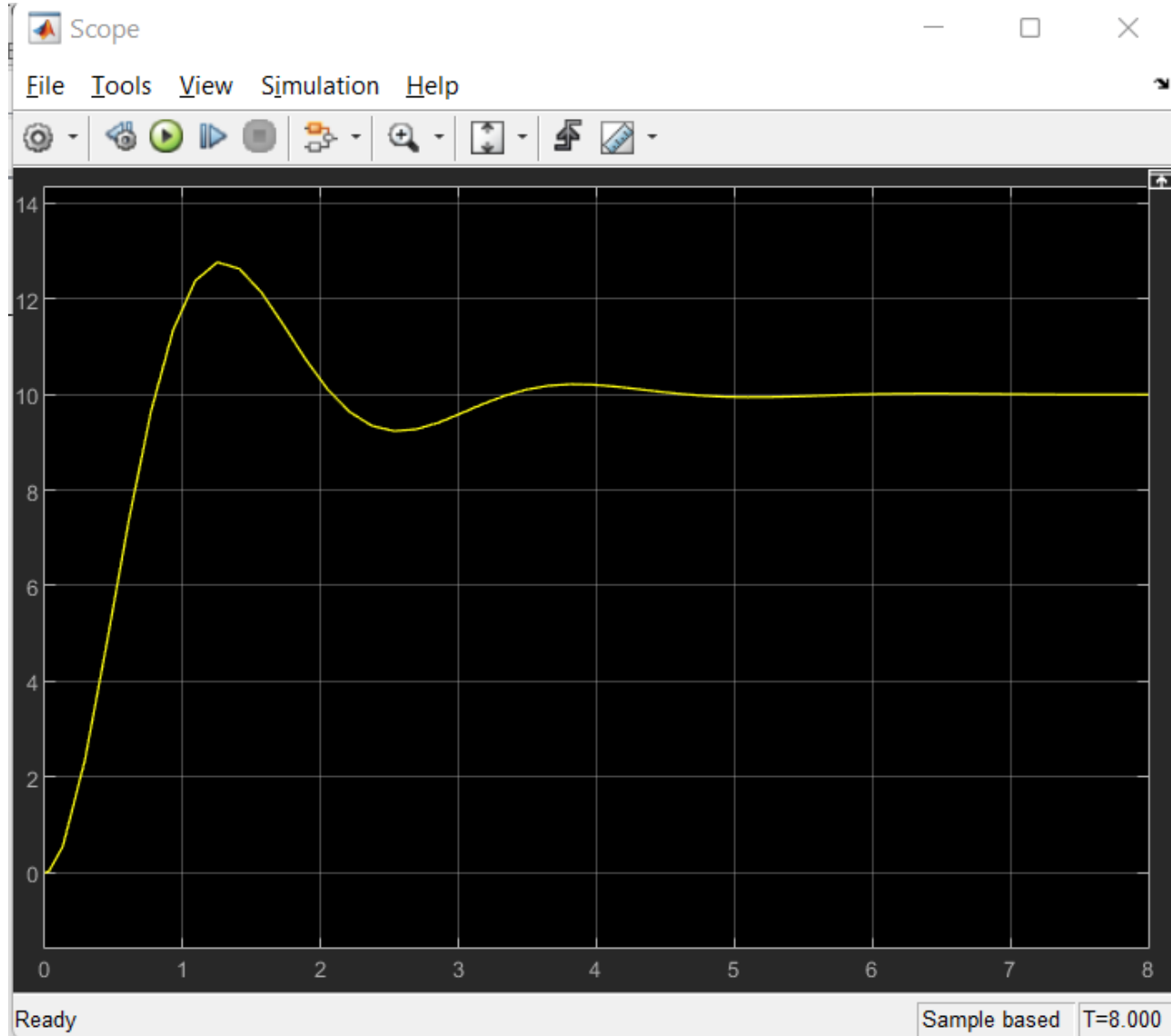
$$y_{max} = y_{\infty} + 0.28y_{\infty} = 1.28 y_{\infty} = 12.8$$

$$\tau_{eq} = \frac{1}{\xi \omega_n} = 1 \text{ s}$$

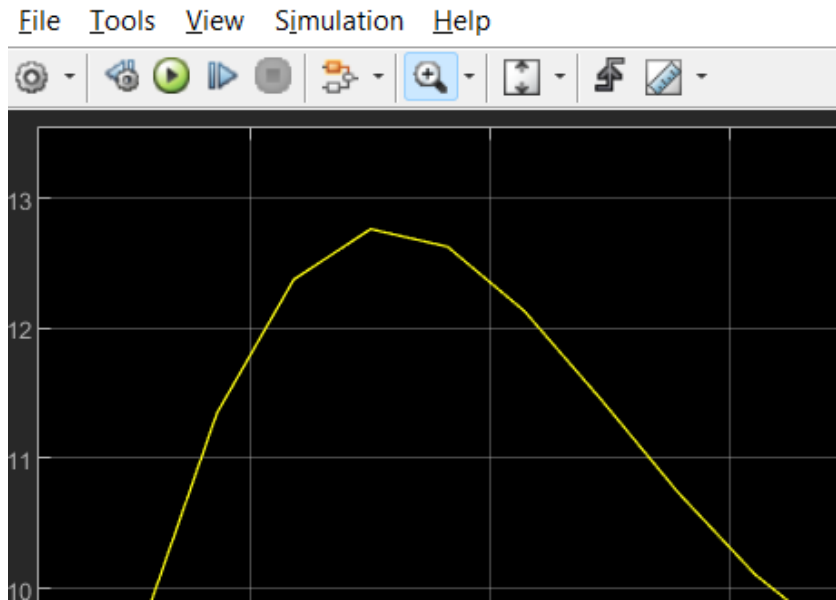
Costante di tempo equivalente e tempi di assestamento

	$T_{a5\%}$	$T_{a2\%}$	$T_{a1\%}$
$F(s) = \frac{\mu \omega_n^2}{(s^2 + 2\xi \omega_n s + \omega_n^2)}$	$3\tau_{eq}$	$3.9 \tau_{eq}$	$4.6 \tau_{eq}$
$F(s) = \frac{14}{s^2 + 2s + 7}$	$3 \text{ s}$	$3.9 \text{ s}$	$4.6 \text{ s}$

Ora dopo avere eseguito la simulazione, fare doppio click sul blocco “Scope”. La risposta possiede le caratteristiche attese







## Grafico “spigoloso”

Il grafico è stato realizzato interpolando un numero di punti insufficiente (l’uscita del sistema è stata calcolata in corrispondenza di un numero insufficiente di istanti temporali tali da non consentire la creazione di un grafico sufficientemente regolare)

Dobbiamo modificare i parametri di configurazione del “**solver**”, che definisce (fra le altre cose) il passo di discretizzazione temporale che viene impiegato nella esecuzione del modello (cioè nella **risoluzione numerica della equazione differenziale** associata al modello).

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

Search

**Solver**

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

Simulation time

Start time: 0.0 Stop time: 8

Solver selection

Type: Fixed-step Solver: auto (Automatic solver selection)

▼ Solver details

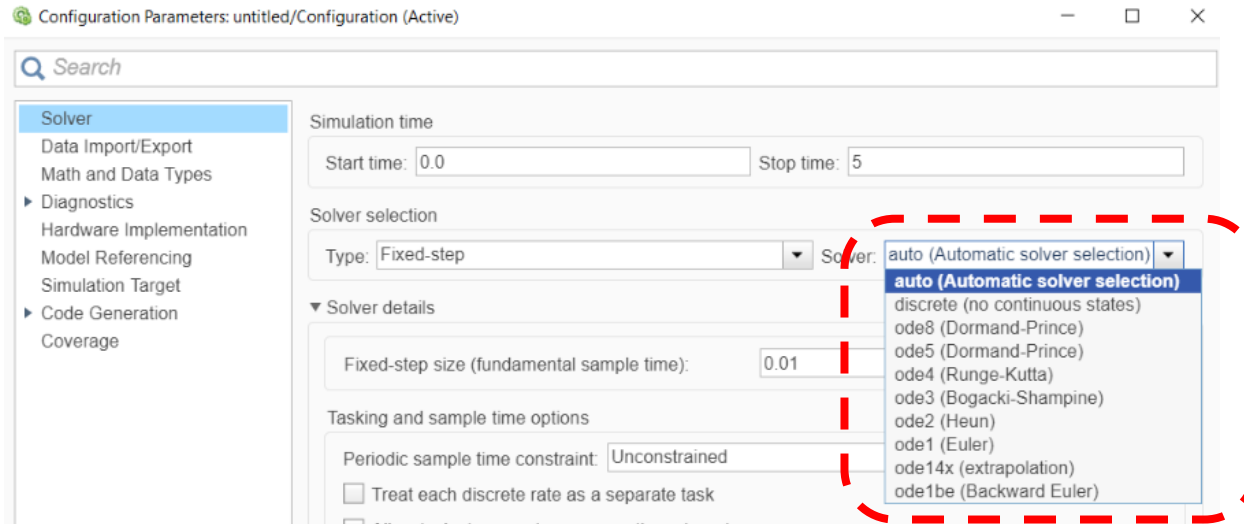
Fixed-step size (fundamental sample time): 0.01

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 (è possibile creare un template di configurazione che consente di creare direttamente un nuovo modello con le impostazioni desiderate)

Naturalmente se si avesse necessita di simulare un sistema dinamico in cui i segnali variano molto rapidamente 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.



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:

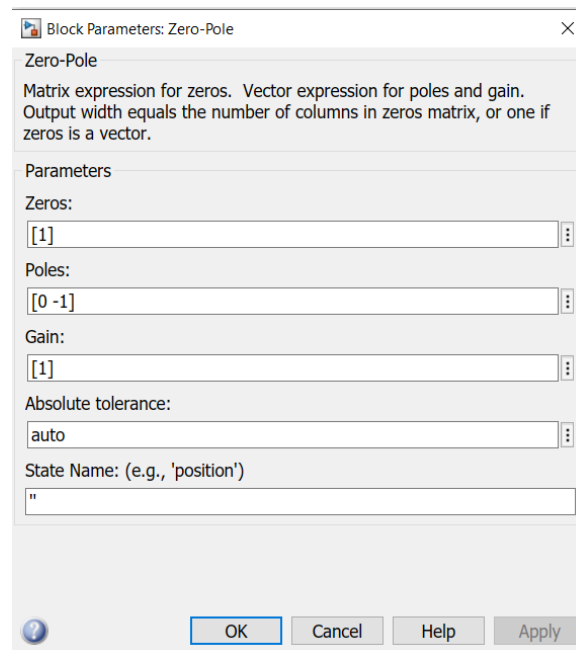
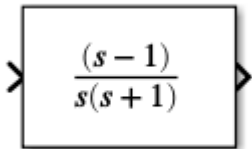
<https://www.mathworks.com/help/simulink/ug/choose-a-solver.html>

Vediamo **un altro blocco elementare per la modellazione di FdT**

Per modellare la seguente FdT il blocco Transfer Fcn risulta poco conveniente

$$F(s) = \frac{2(s + 0.5)}{(s + 1)(s + 2)(s + 5)}$$

Si può utilizzare in alternativa il blocco «Zero-Pole», che si trova sempre nella medesima libreria Continuous dalla quale prelevammo il blocco Transfer Fcn



A screenshot of the 'Block Parameters: Zero-Pole' dialog box. The dialog has a title bar with a close button. The main area contains the following fields:

- Zero-Pole**: Matrix expression for zeros. Vector expression for poles and gain. Output width equals the number of columns in zeros matrix, or one if zeros is a vector.
- Parameters**:
  - Zeros:** [1]
  - Poles:** [0 -1]
  - Gain:** [1]
  - Absolute tolerance:** auto
  - State Name:** (e.g., 'position')

At the bottom, there are buttons for '?', 'OK', 'Cancel', 'Help', and 'Apply'.

Vettore che contiene tutti gli zeri

Vettore che contiene tutti i poli

Guadagno in alta frequenza

$$F(s) = \frac{2(s + 0.5)}{(s + 1)(s + 2)(s + 5)}$$

Block Parameters: Zero-Pole

Zero-Pole

Matrix expression for zeros. Vector expression for poles and gain.  
Output width equals the number of columns in zeros matrix, or one if zeros is a vector.

Parameters

Zeros:

[-0.5]

Poles:

[-1 -2 -5]

Gain:

[2]

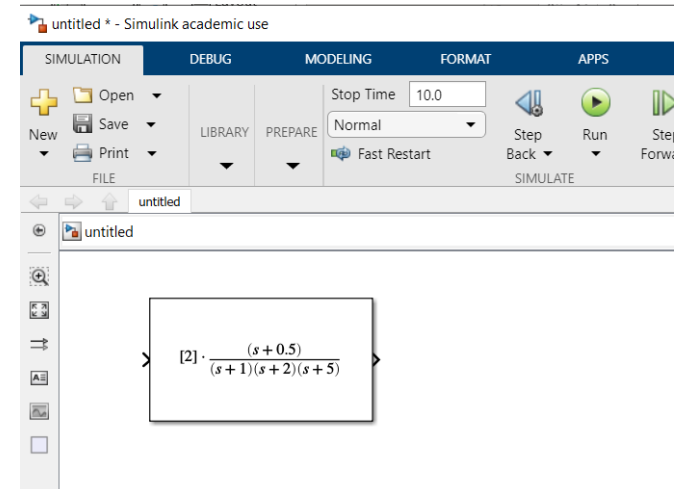
Absolute tolerance:

auto

State Name: (e.g., 'position')

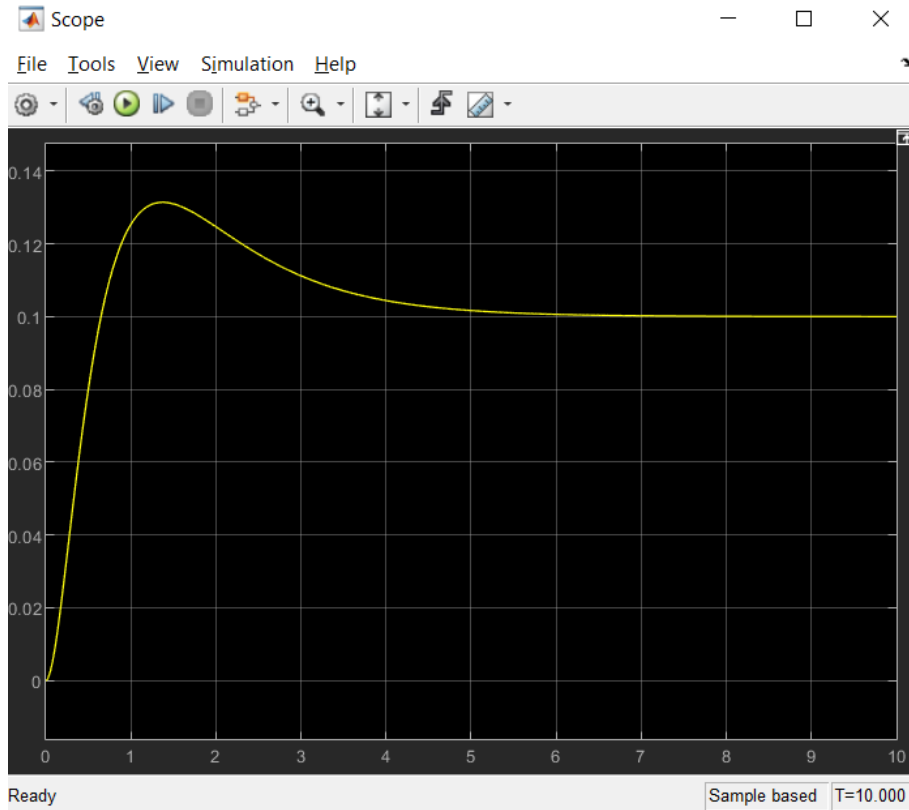
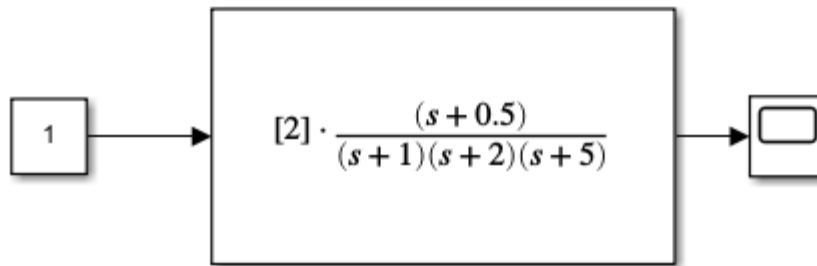
"

OK Cancel Help Apply



Visualizzare la risposta al gradino unitario.

Che tipo di curva ci aspettiamo ?

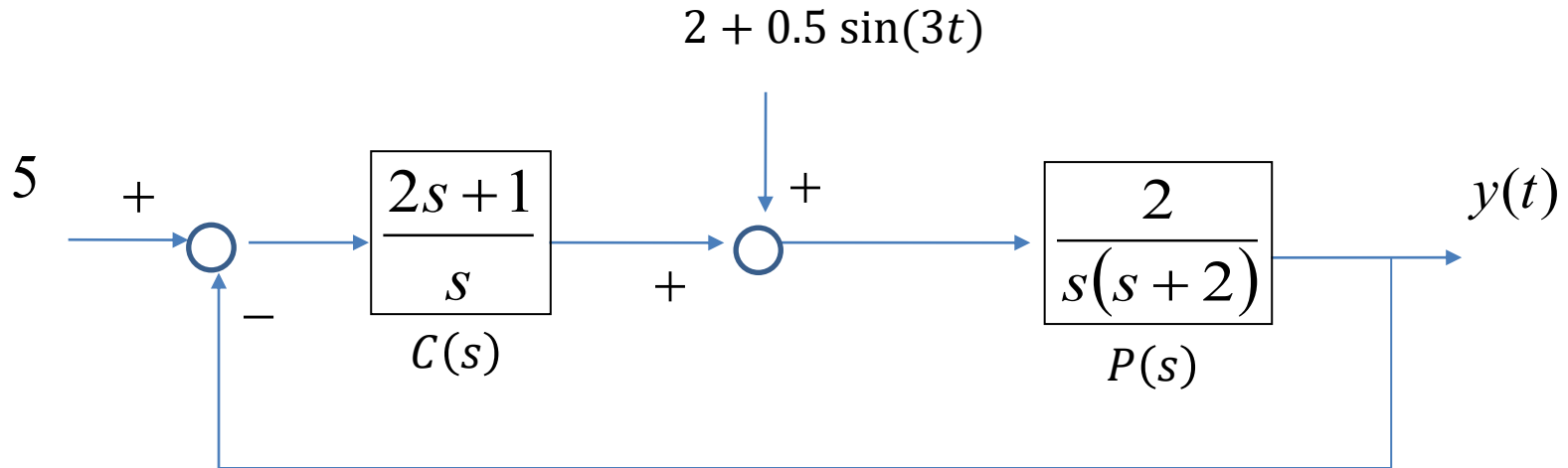


Sovraelongazione indotta dallo zero in bassa frequenza

Dopo il punto di massimo, convergenza monotona verso il regime.

Assenza di comportamenti oscillatori

## Simulazione dinamica di sistemi di controllo LTI in retroazione

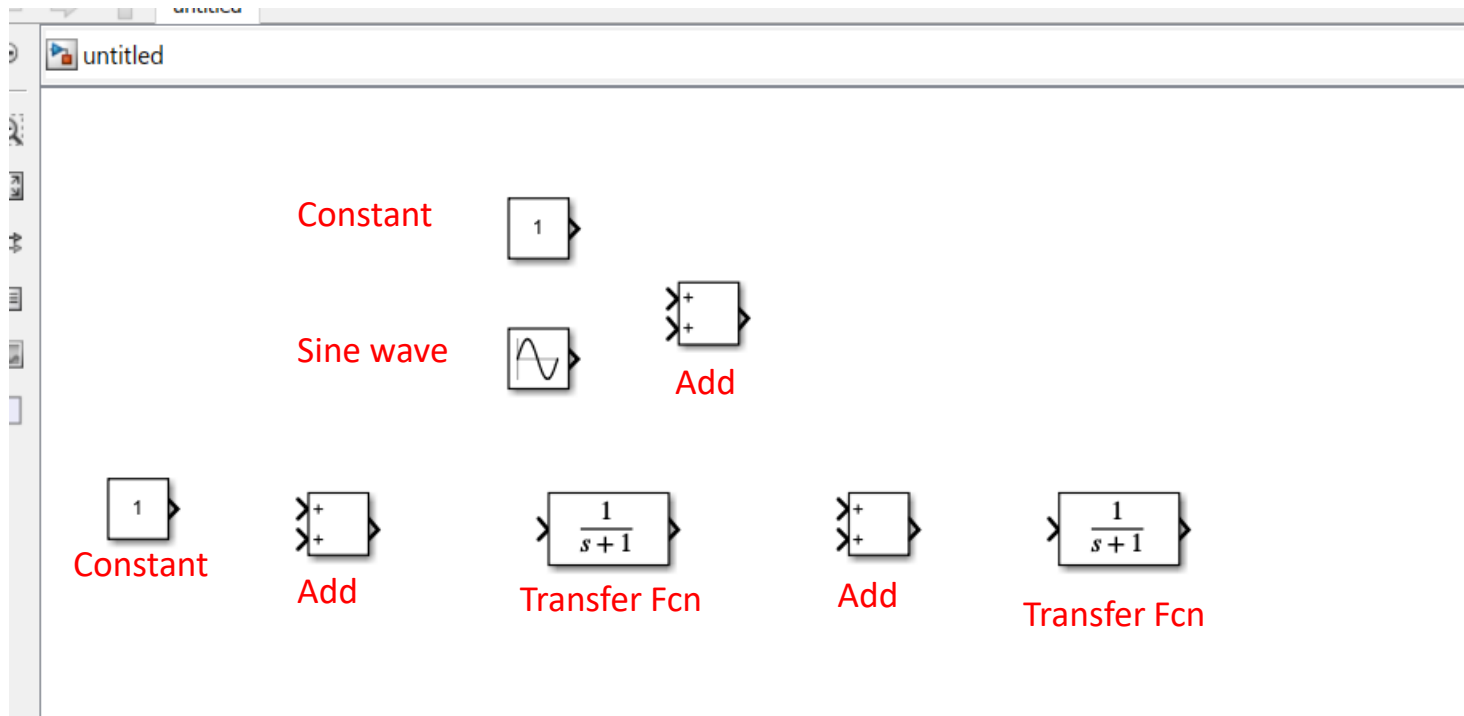


Dato il sistema di controllo in figura, valutare mediante simulazione dinamica il comportamento transitorio e di regime dell'uscita.

Dobbiamo realizzare nella pagina di lavoro Simulink uno schema in tutto e per tutto equivalente a quello riportato in alto. Rispetto all'esempio precedente serviranno dei blocchi elementari aggiuntivi: due blocchi «Transfer Function», vari nodi sommatori (blocco «Add»), due blocchi «constant» per generare le componenti costanti del set point e del disturbo ed un blocco «Sine wave» per generare la componente sinusoidale del disturbo.

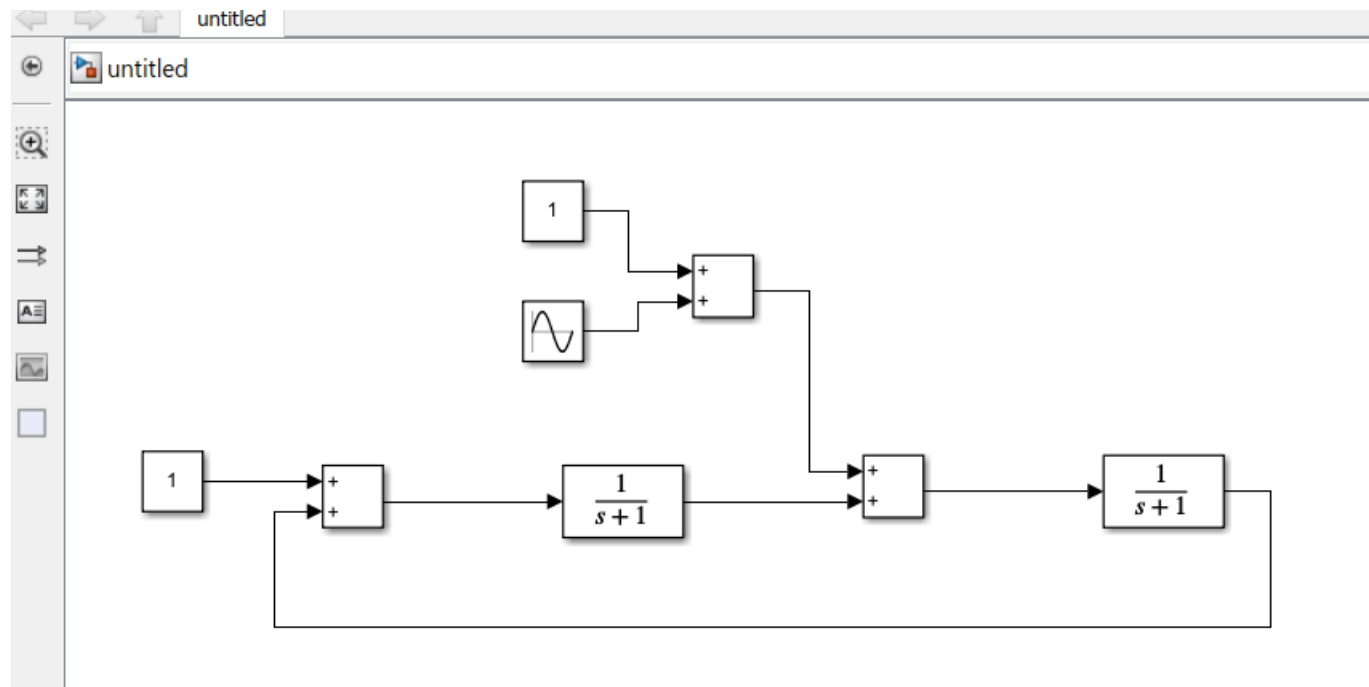
Dopo avere aperto, con le medesime modalità impiegate in precedenza, un modello Simulink «in bianco», vediamo un modo alternativo per importare nello schema i blocchi elementari che ci servono. E' sufficiente fare doppio click in un punto qualsiasi dello schema, e scrivere le prime lettere del nome del blocco affinché venga generata una lista che include il blocco cercato, che può quindi essere selezionato ed importato nel modello.

Importare con la modalità suggerita i blocchi elementari menzionati nella slide precedente, e disporli come nella figura sottostante



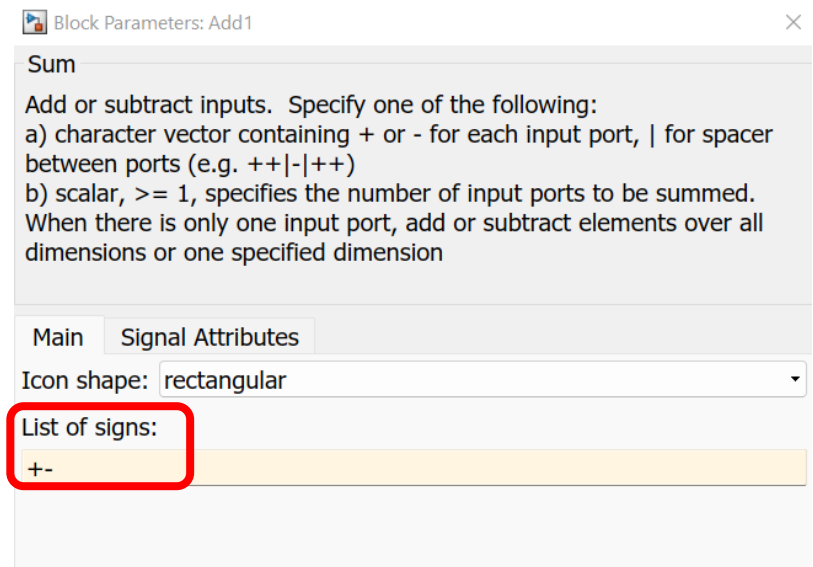
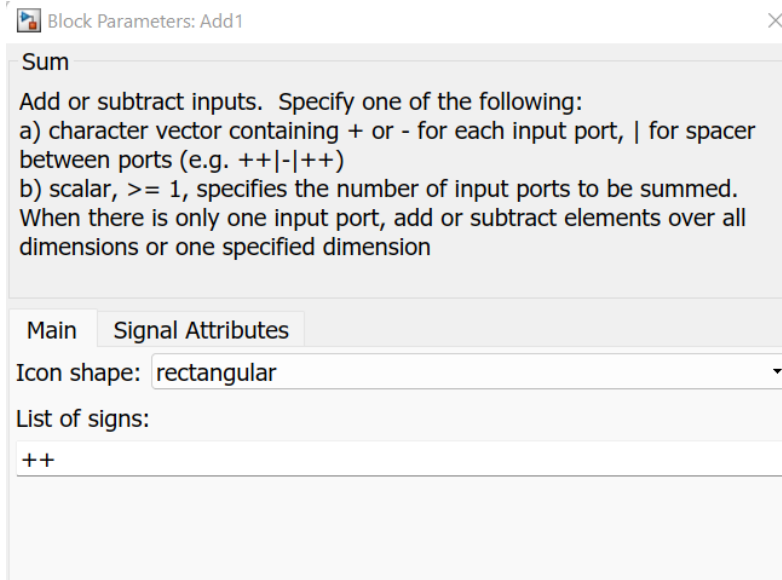


Realizziamo le connessioni come in figura

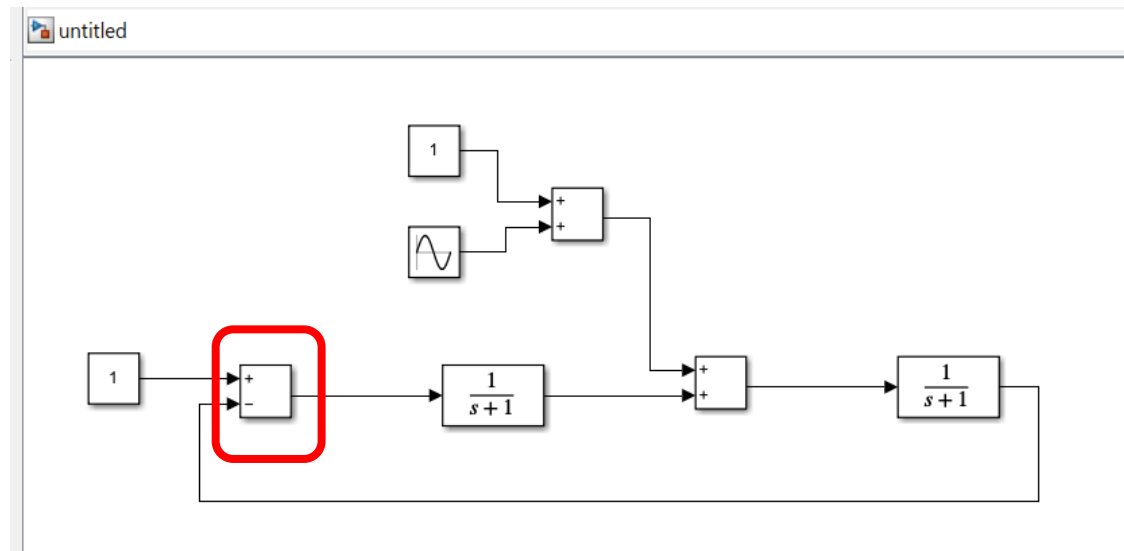


Ora vanno parametrizzati i vari blocchetti. Ciascun blocco, come visto in precedenza, si parametrizza facendo doppio click su di esso ed accedendo alla relativa maschera di parametrizzazione.

Iniziamo con il modificare il nodo sommatore in cui deve essere eseguita la differenza fra il set-point e l'uscita.



Cambia l'aspetto del nodo sommatore, e compare il terminale negativo



Ora parametrizziamo i due blocchi «Constant» ed i due blocchi «Transfer Fcn»

$$C(s) = \frac{2s + 1}{s}$$

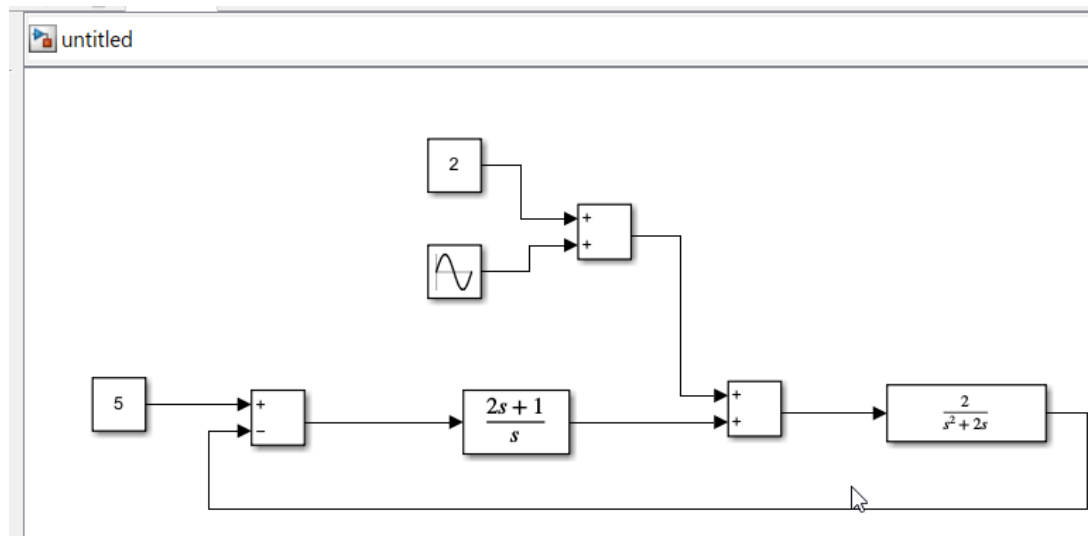


Parameters	
Numerator coefficients:	
[2	1]
Denominator coefficients:	
[1	0]

$$P(s) = \frac{2}{s(s + 2)} = \frac{2}{s^2 + 2s}$$



Parameters	
Numerator coefficients:	
[2]	
Denominator coefficients:	
[1	2 0]



Ora dobbiamo parametrizzare il blocco **Sine Wave** affinché generi il segnale  $0.5 \sin(3t)$   
Mediante il blocco Sine Wave è possibile generare un segnale avente la forma seguente:

$$A \sin(\omega t + \phi) + B$$

Amplitude      Frequency      Phase      Bias

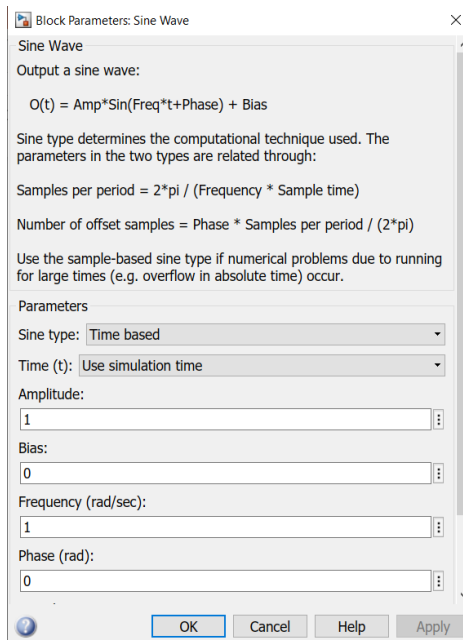
$A = \text{Amplitude} = 0.5$

$\omega = \text{Frequency} = 3$

$\phi = \text{Phase} = 0$

$B = \text{Bias} = 0$

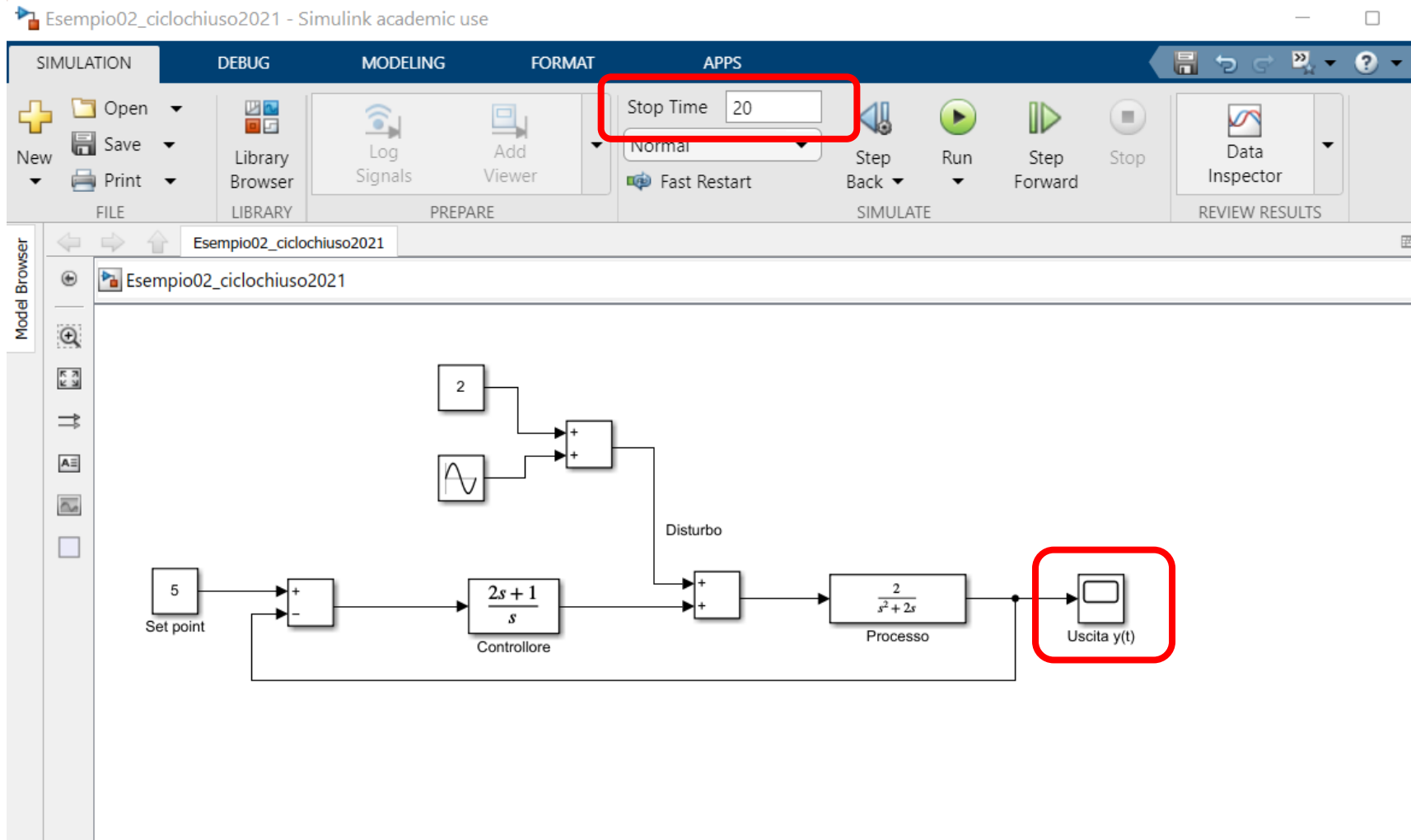
Finestra di parametrizzazione (i valori di default producono in uscita il segnale  $\sin(t)$ )



Per generare il segnale  $0.5 \sin(3t)$  si parametrizzi il blocco inserendo i valori soprariportati (v. sotto)

Parameters	
Sine type:	Time based
Time (t):	Use simulation time
Amplitude:	0.5
Bias:	0
Frequency (rad/sec):	3
Phase (rad):	0

Inseriamo il blocco Scope e colleghiamoci in ingresso l'uscita del sistema, impostiamo la durata della simulazione a **20 secondi**, modifichiamo i parametri del solutore numerico come fatto nell'esempio precedente. Siamo ora pronti a mandare in run la simulazione



# Evoluzione temporale dell'uscita

