a place of mind

ELEC 341: Systems and Control

Lecture 21

Frequency response shaping with Matlab (Simulink simulation)

a place of mind UBC

Course roadmap

Modeling

- ✓ Laplace transform
- ✓ Transfer function
- Models for systems
  - ✓ Electrical
  - ✓ Electromechanical
  - ✓ Mechanical
- ✓ Linearization, delay

Analysis

- ✓ Stability
  - ✓ Routh-Hurwitz
  - ✓ Nyquist
- ✓ Time response
  - ✓ Transient
  - ✓ Steady state
- ✓ Frequency response
  - ✓ Bode plot

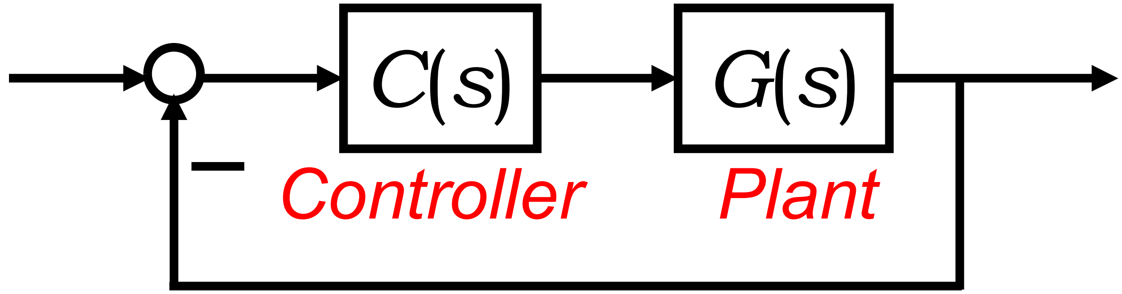
Design

- ✓ Design specs
- ✓ Root locus
- ✓ Frequency domain
- ✓ PID & Lead-lag
- ✓ Design examples

Matlab simulations

Example 1
(SISO Design Tool in Matlab)

- Consider a system:



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

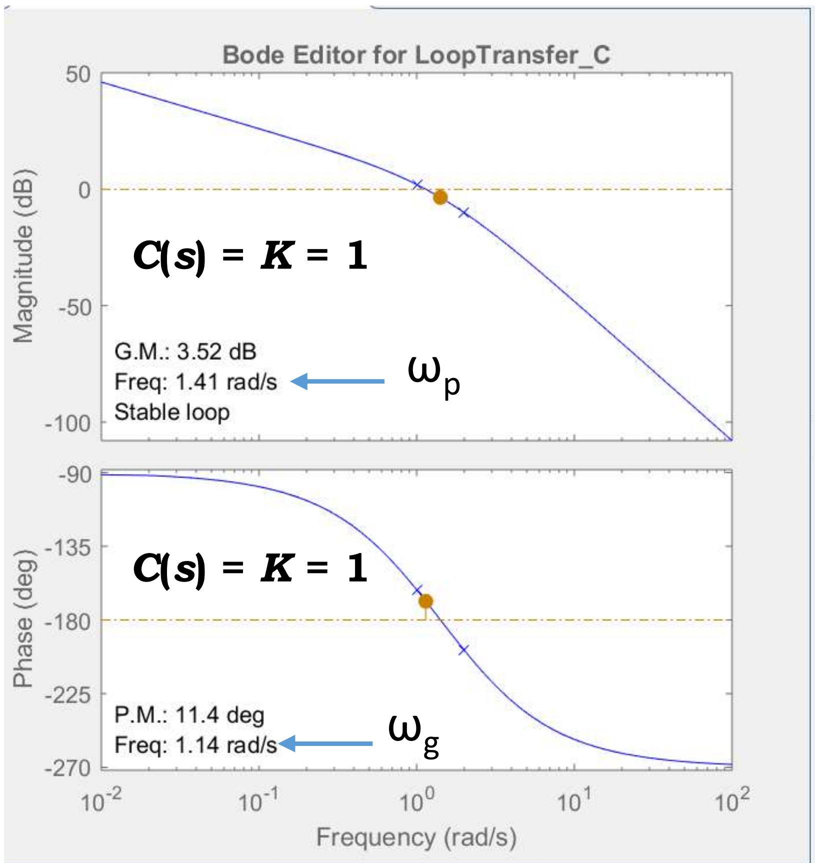
- Design specs:
  - Closed-loop system is stable
  - PM at least 50 deg
  - 2% Settling time < 4 s
  - Steady-state error
    - For unit step input: $e_{ss} = 0$

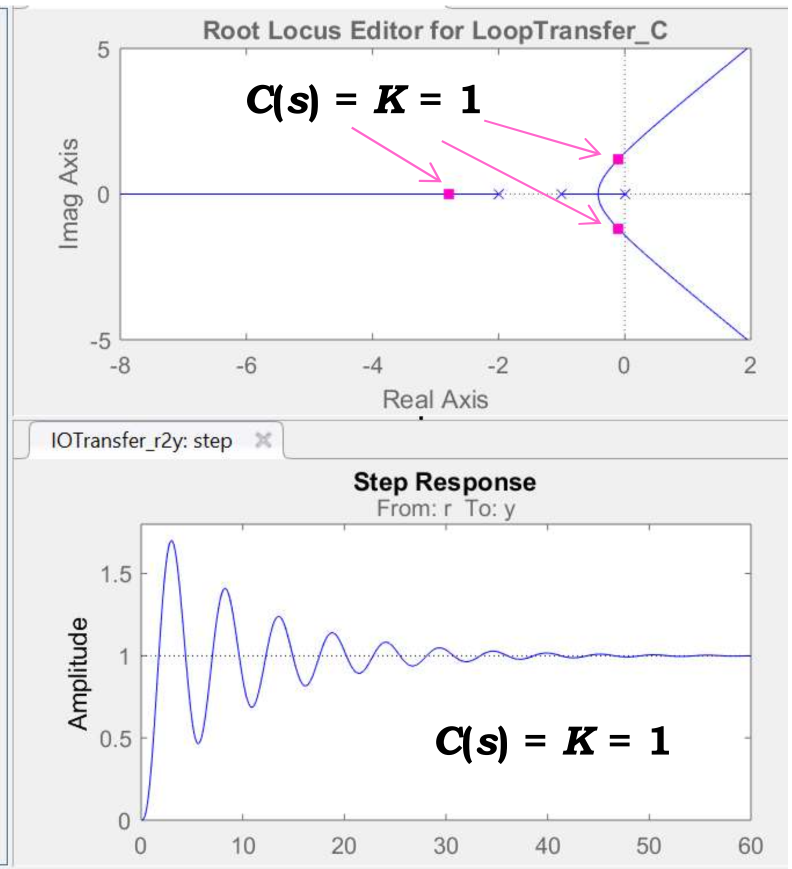
Example 1 (cont'd)

OL Bode plot

Root locus

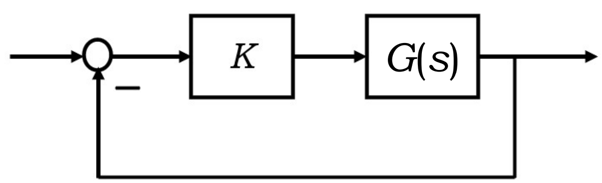
```
>> s = tf('s')
Transfer function:
s
>> sysG = 4/s/(s+1)/(s+2)
Transfer function:
        4
-----------------
s^3 + 3 s^2 + 2 s
>> sisotool(sysG)
```





Default setting:
$C(s) = K = 1$

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

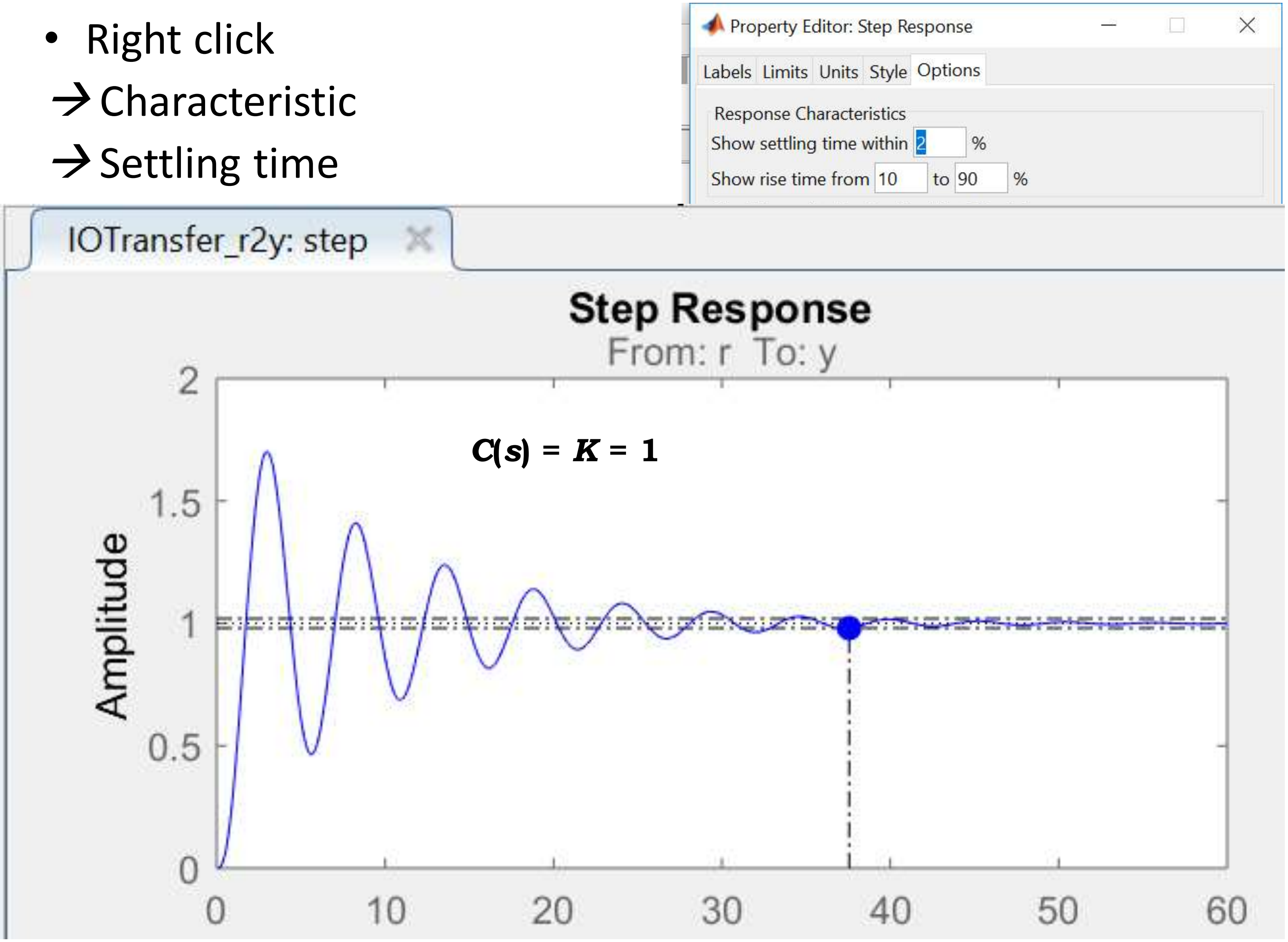


Example 1 (cont'd)

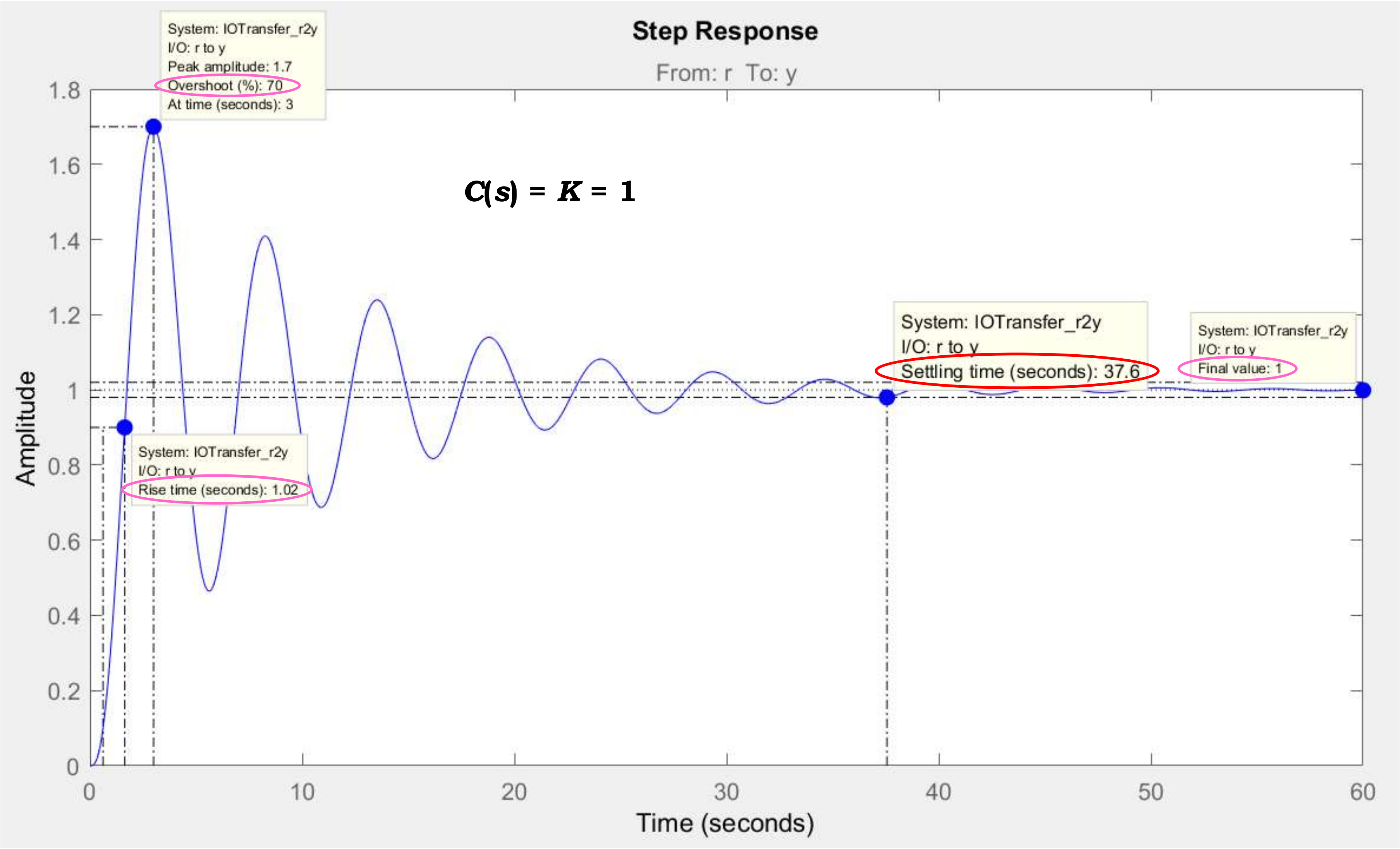
- Show settling time
  - Right click

  → Characteristic

  → Settling time



Example 1 (cont'd)



a place of mind UBC

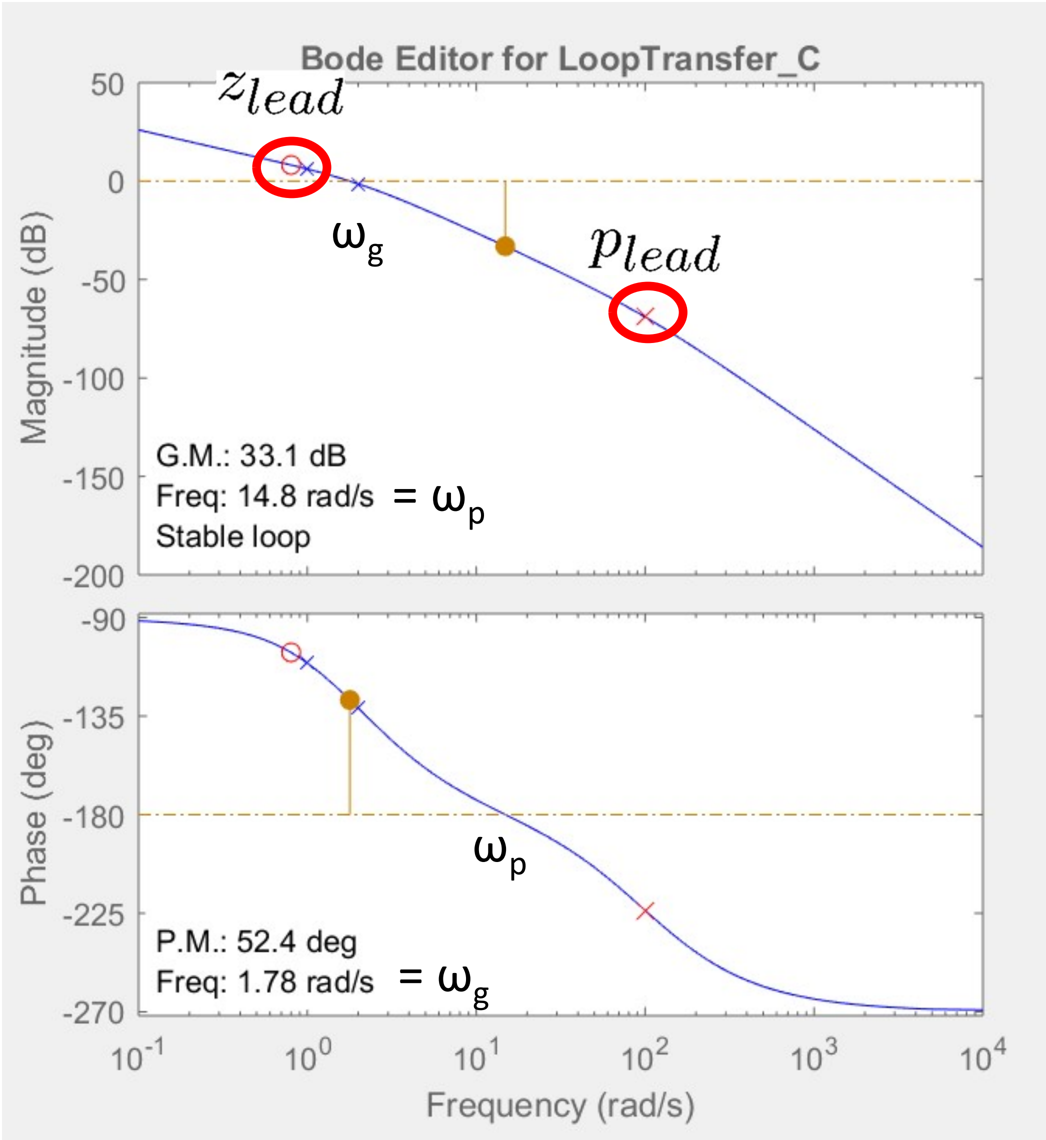
Example 1 (cont'd)

- Add a pole & a zero of a compensator:

$$C_{Lead}(s) = K\frac{s + z_{Lead}}{s + p_{Lead}}$$

- If necessary, move the pole/zero/gain
  - by click-and-drag, or
  - *Design → Edit Compensator…*

```
Tunable Block
Name: C
Sample Time: 0
Value:
  125 (s+0.8)
  -----------
    (s+100)
```



PM (= 52.4) > 50 degree OK!

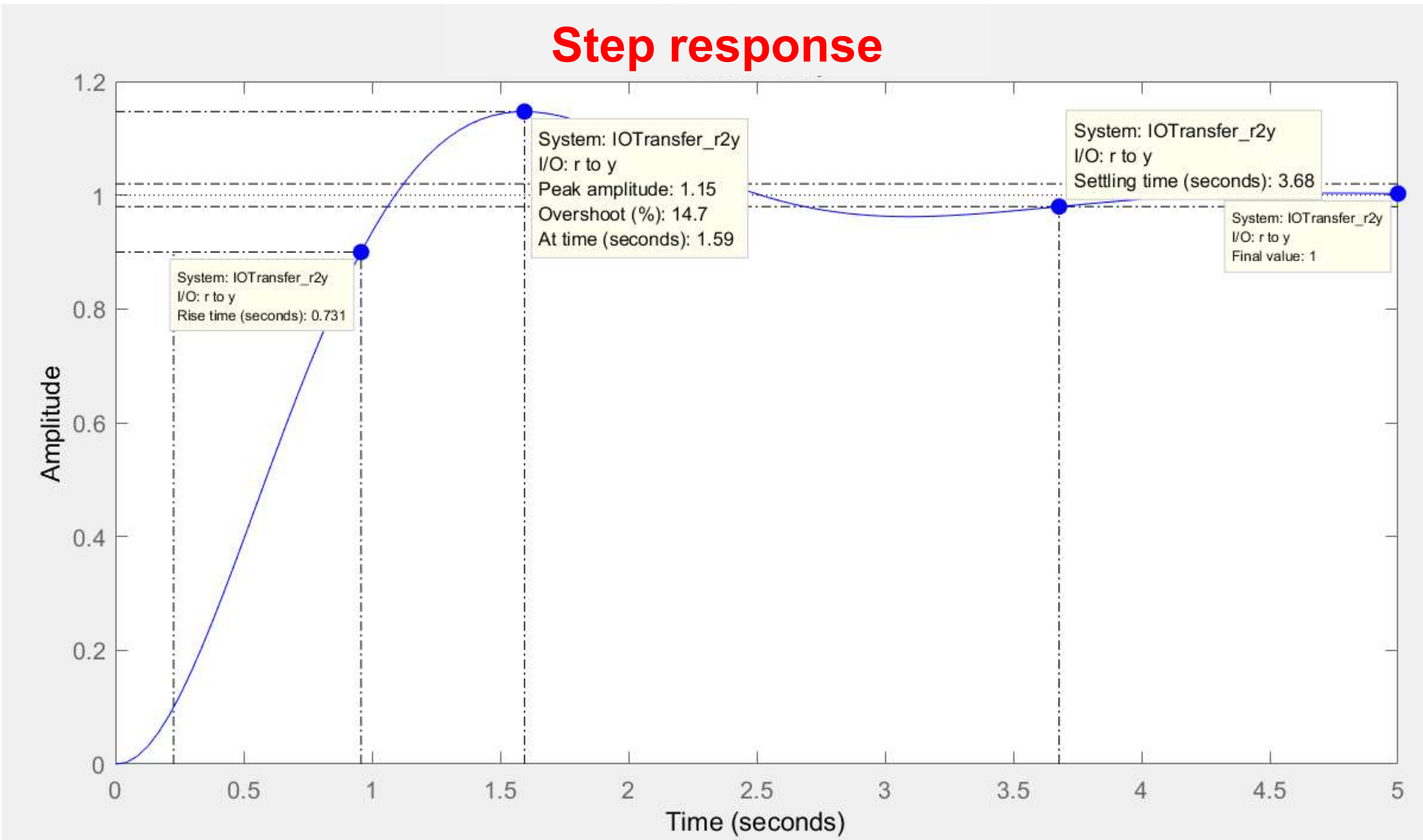
Example 1 (cont'd)

The following response satisfies all the design specifications:

- **(a)** Closed-loop system is stable
- **(b)** PM at least 50°. It is 52.4°.
- **(c)** 2% Settling time < 4 s. It is 3.68 s.
- **(d)** Steady-state error is zero for a unit step input

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

$$C_{Lead}(s) = K\frac{s + z_{Lead}}{s + p_{Lead}} = 125\,\frac{s + 0.8}{s + 100}$$



$$C_{Lead}(s).G(s) = 125\,\frac{s+0.8}{s+100}\cdot\frac{4}{s(s+1)(s+2)}$$

Simulink

- **Simulink**, developed by MathWorks, is a graphical programming environment for modeling, simulating, and analyzing dynamic systems.
  - Its primary interface is a graphical block diagramming tool and a customizable set of block libraries.
  - It offers tight integration with the rest of the MATLAB environment.
- Simulink is basically a piece of software for modeling and simulating a system, as well as programing and designing controllers.
- Engineers use Simulink to solve engineering problems in many industries, such as:
  - Automotive
  - Biomedical
  - Aerospace
  - Chemical processes
  - Communications
  - Industrial automation
  - Electronics
  - etc.

Simulink

- Simulink in MATLAB can be directly used with **Arduino** to design, simulate, and deploy control systems and embedded applications without writing traditional code.
- With Simulink support packages for Arduino, you can build a block diagram (***instead of writing code***), simulate how your system behaves, and then download the model directly onto the Arduino hardware.
- This allows you to test real-time control algorithms, read sensors, and drive actuators using a **visual programming** approach, making it ideal for rapid prototyping and education.

Simulink

What is Visual Programming?

- A **visual programming approach** means you create programs by **connecting blocks or components in a graphical interface**, rather than writing lines of text-based code. You "program" by dragging, dropping, and linking blocks that represent operations (like reading a sensor, doing math, or turning on an LED).

How Simulink Uses It:

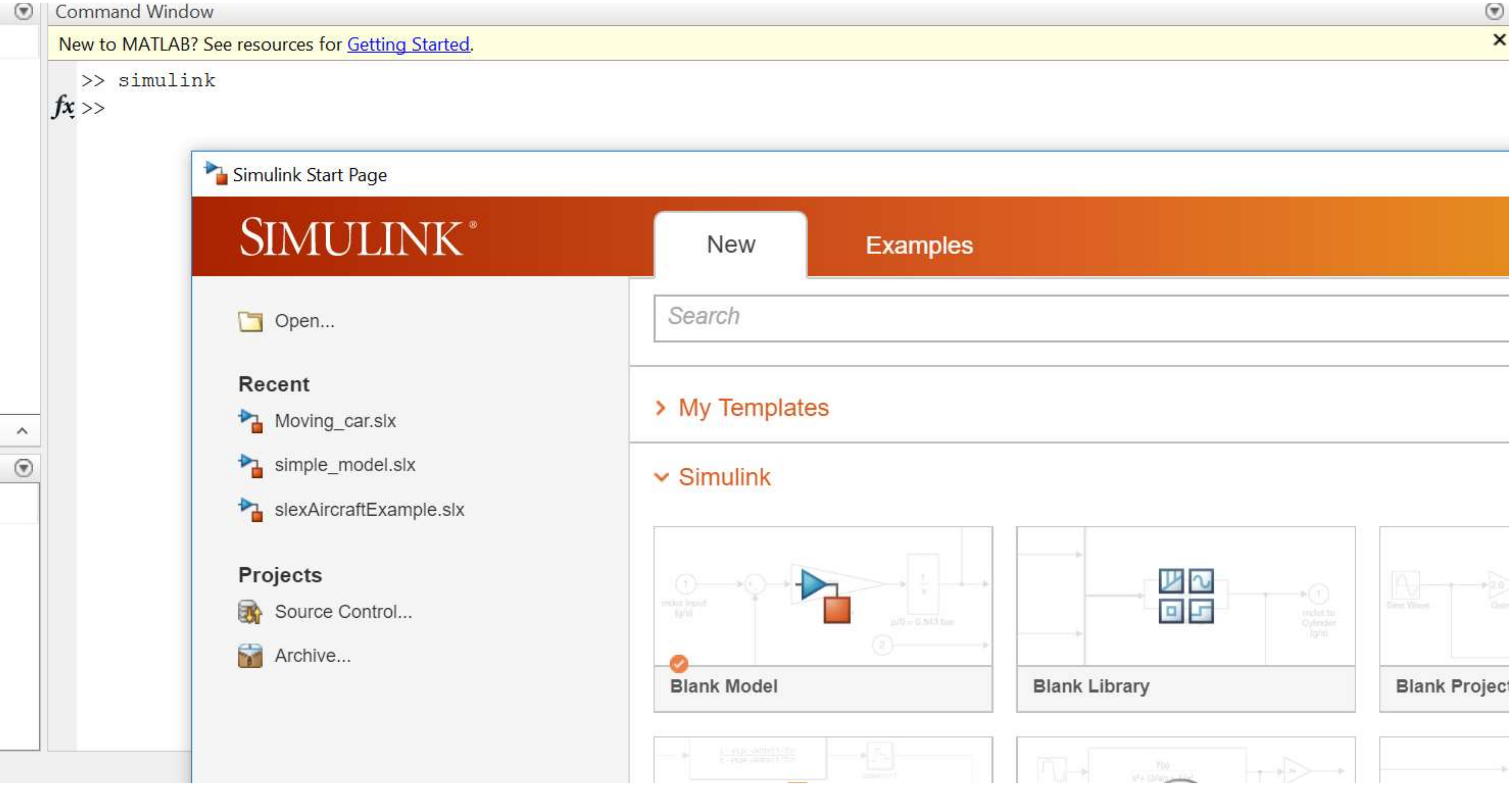
- In **Simulink**, you build your system using block diagrams — each block performs a specific function. For example:
  - A block for reading analog input
  - A block for multiplying a signal
  - A block for outputting a value to a motor
- Once your model is complete, Simulink converts it into C++ code behind the scenes and uploads it to the Arduino.

Summary:

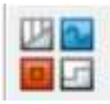
- **C++** is what Arduino typically runs.
- **Simulink** lets you skip writing C++ by **visually designing** your system.
- This is helpful for students, engineers, and educators who want to focus on logic and control, not on low-level code.

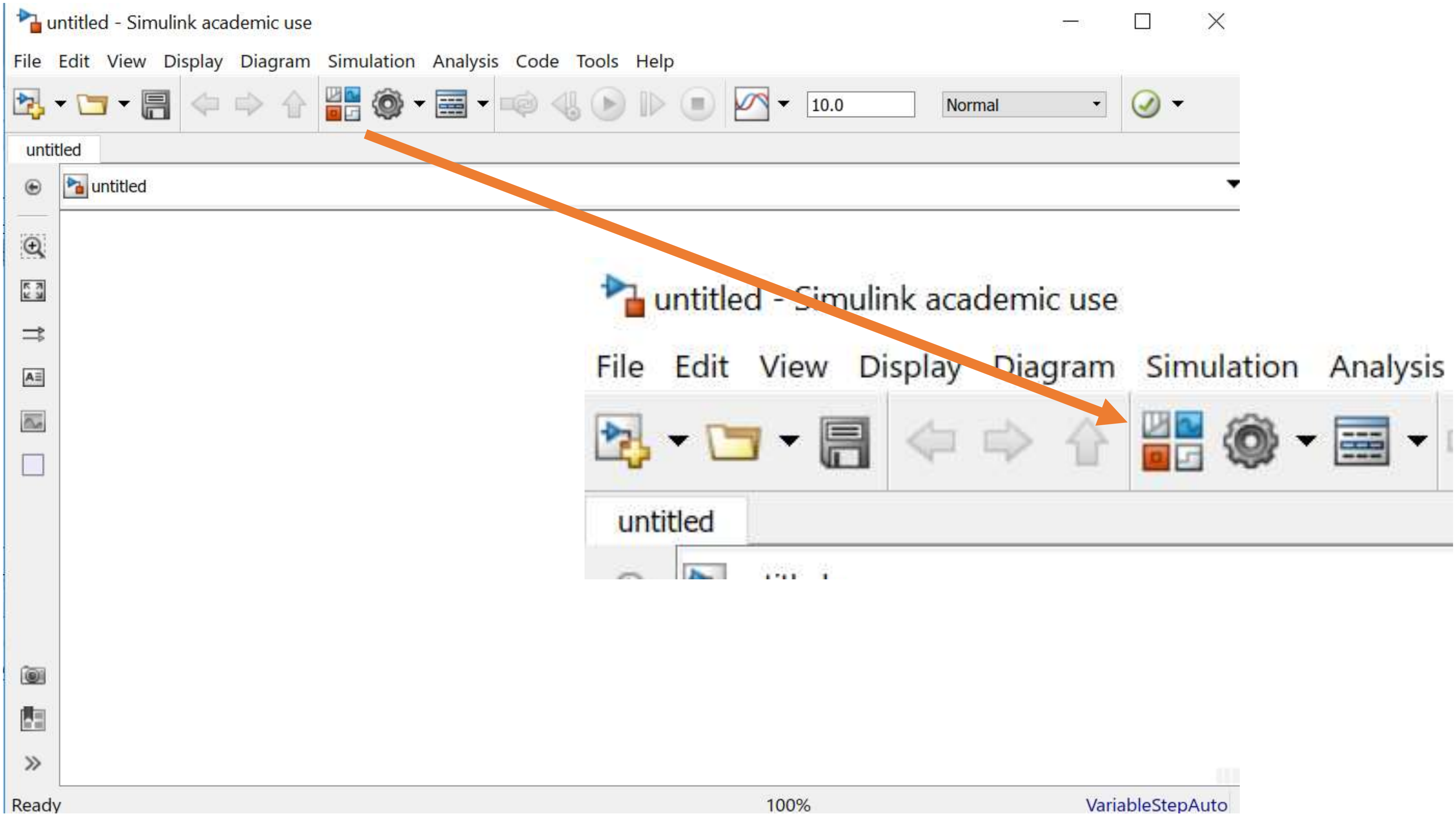
Example 2 (cont'd)

- In MATLAB prompt, type "simulink".
- Click on Blank Model.



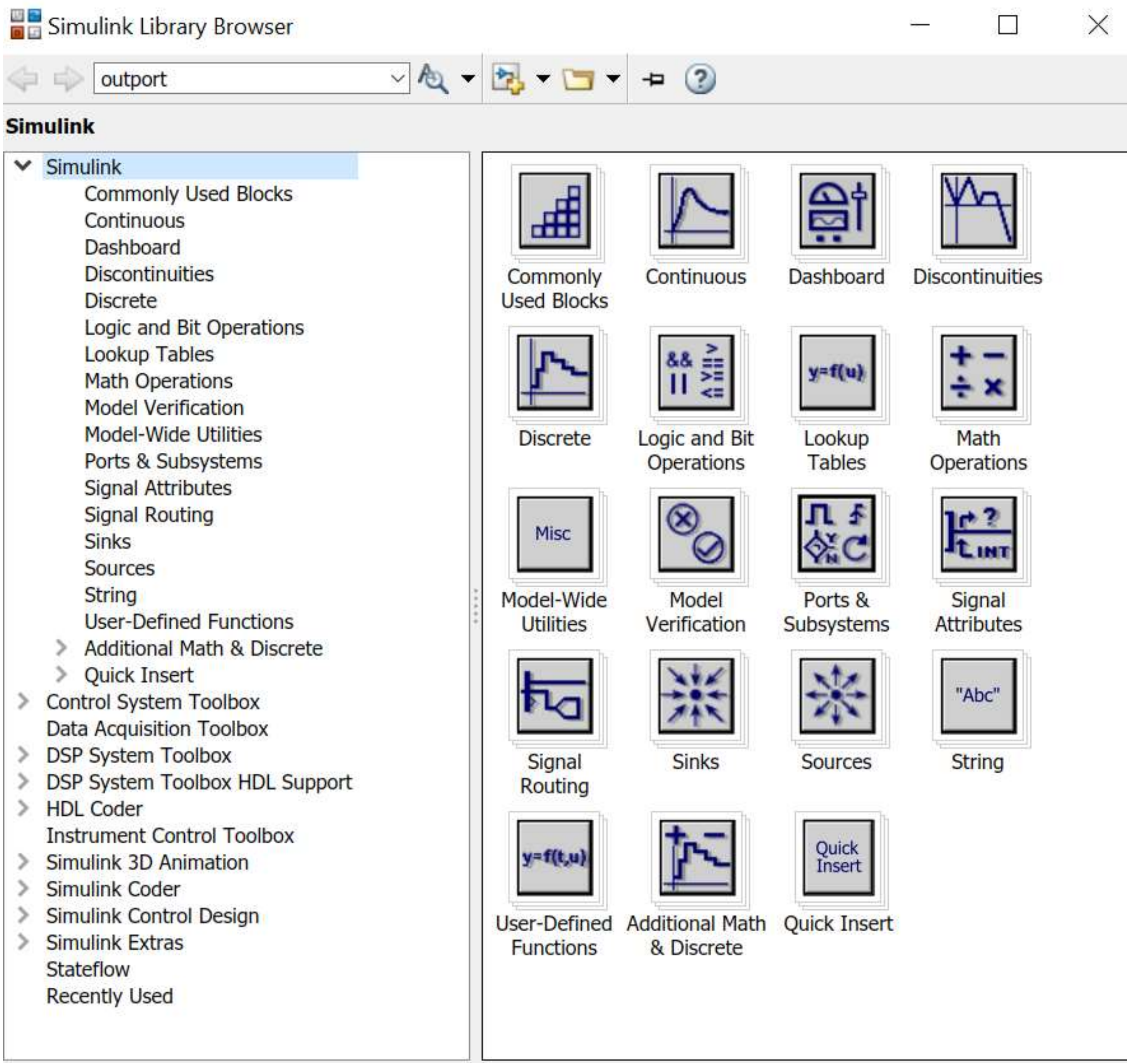
Example 2 (cont'd)

- Click on 

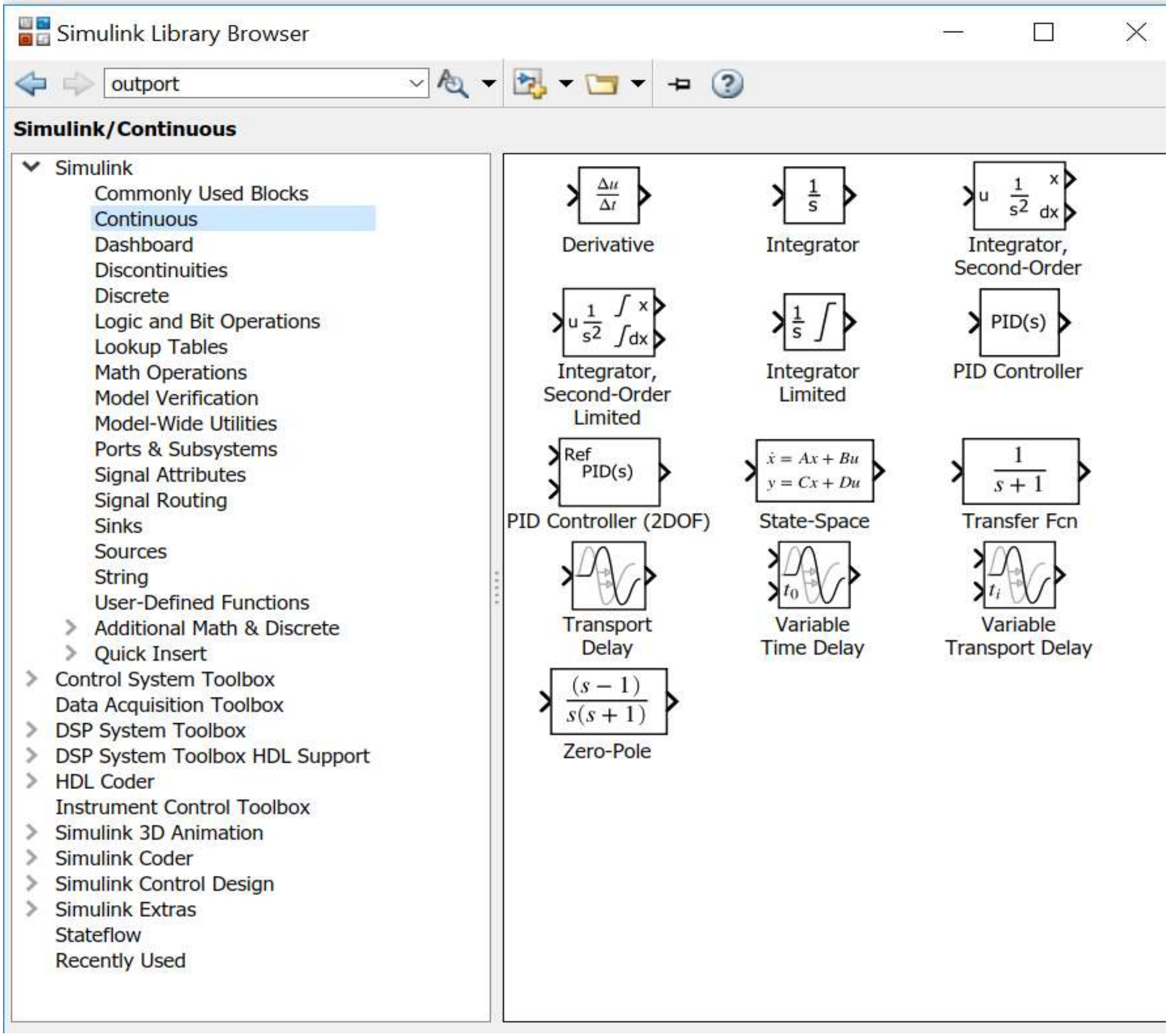


Example 2 (cont'd)

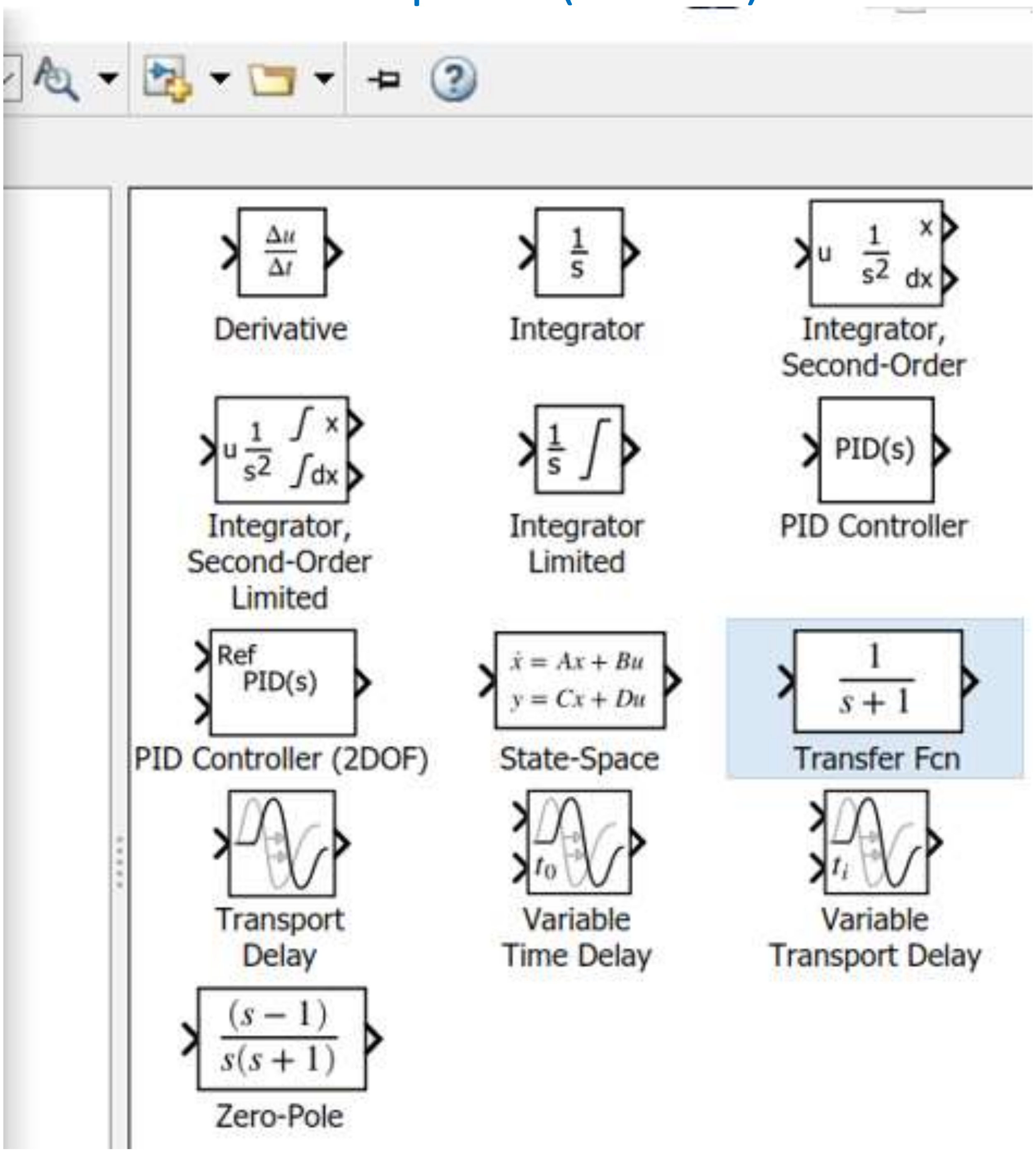
Then, Simulink Library Browser pops up:



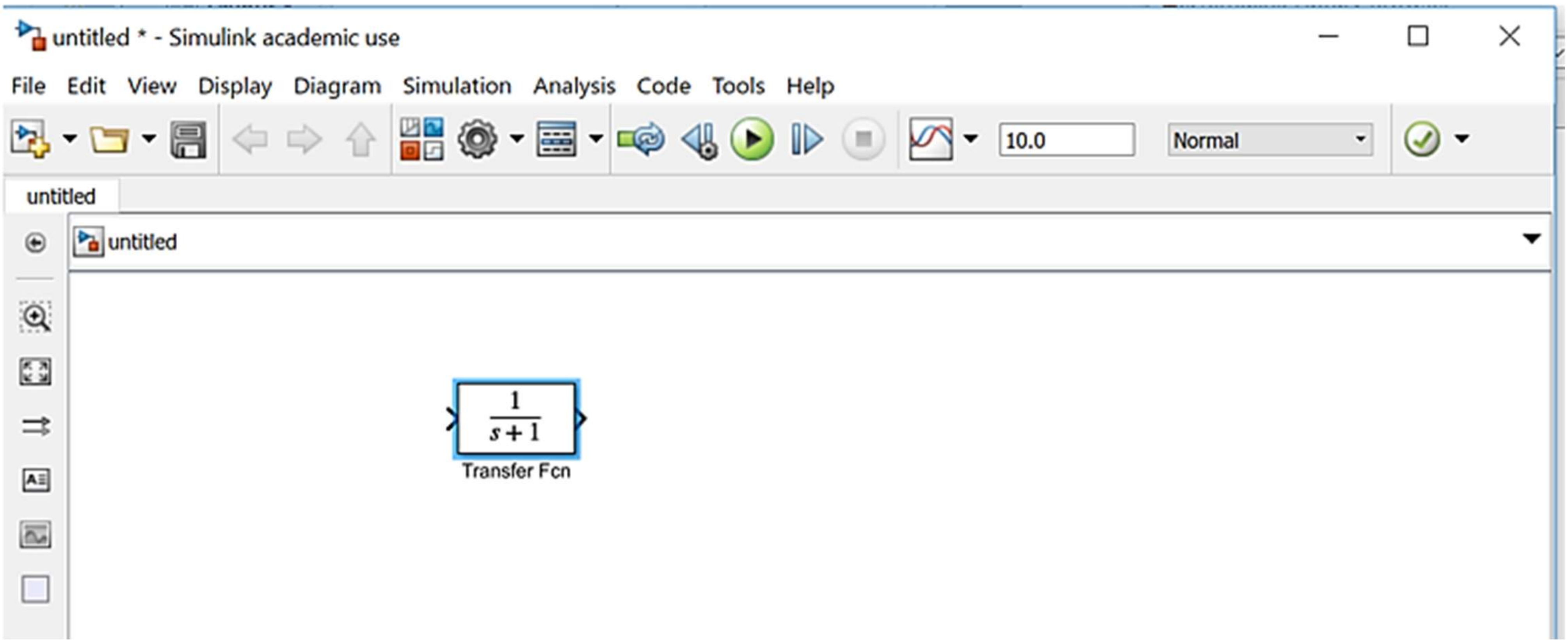
Example 2 (cont'd)



Example 2 (cont'd)

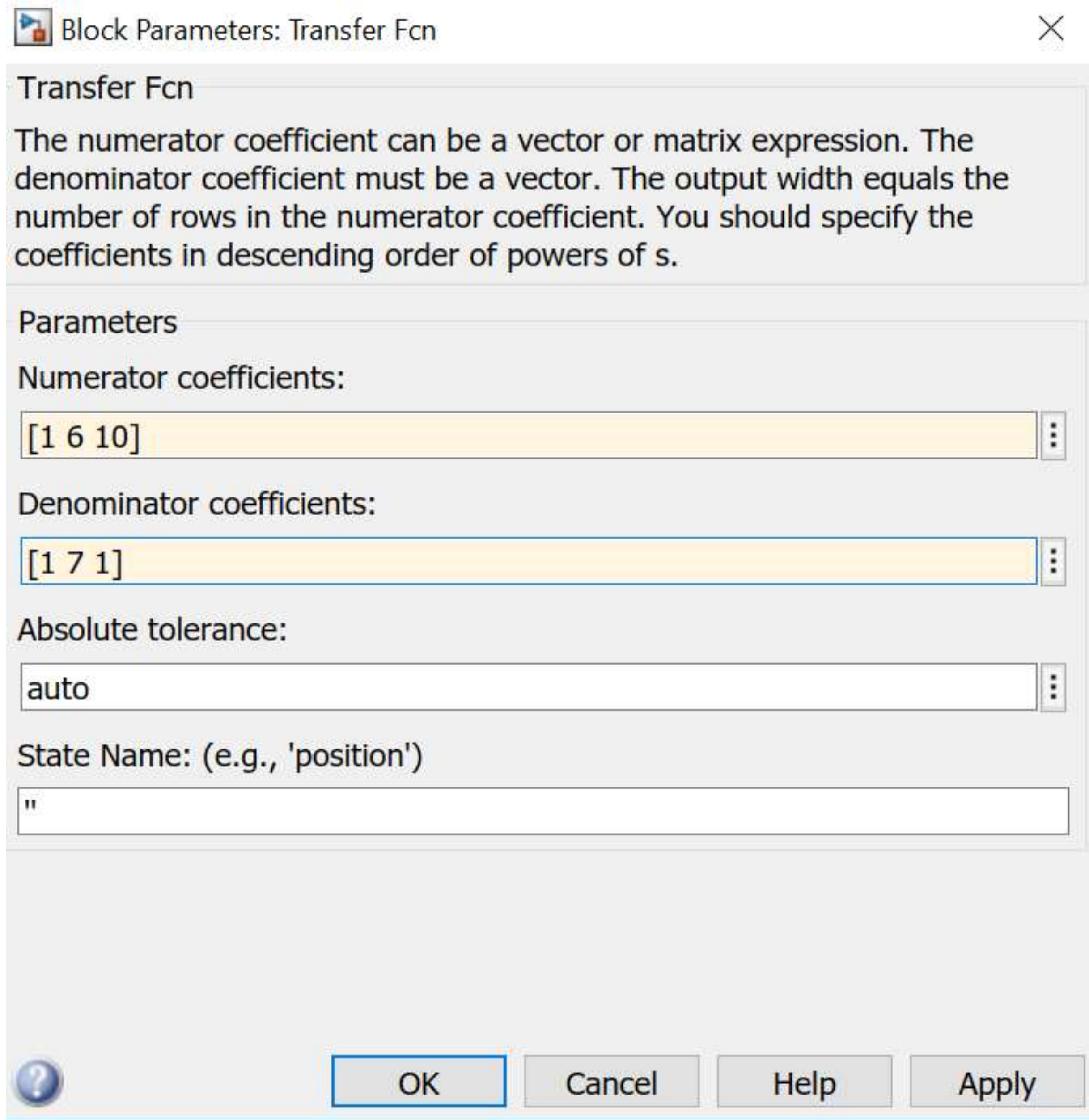


Example 2 (cont'd)

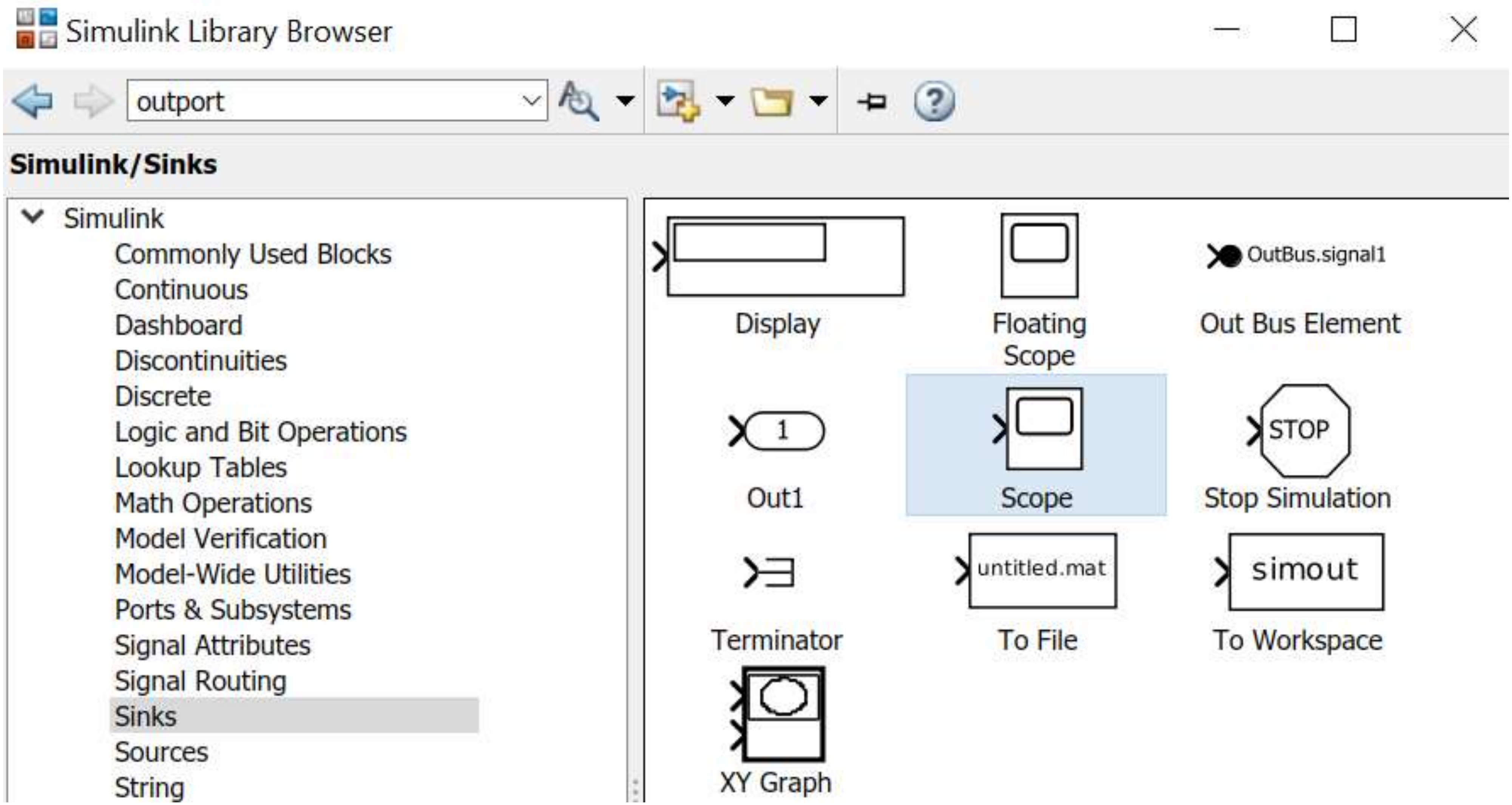


Example 2 (cont'd)

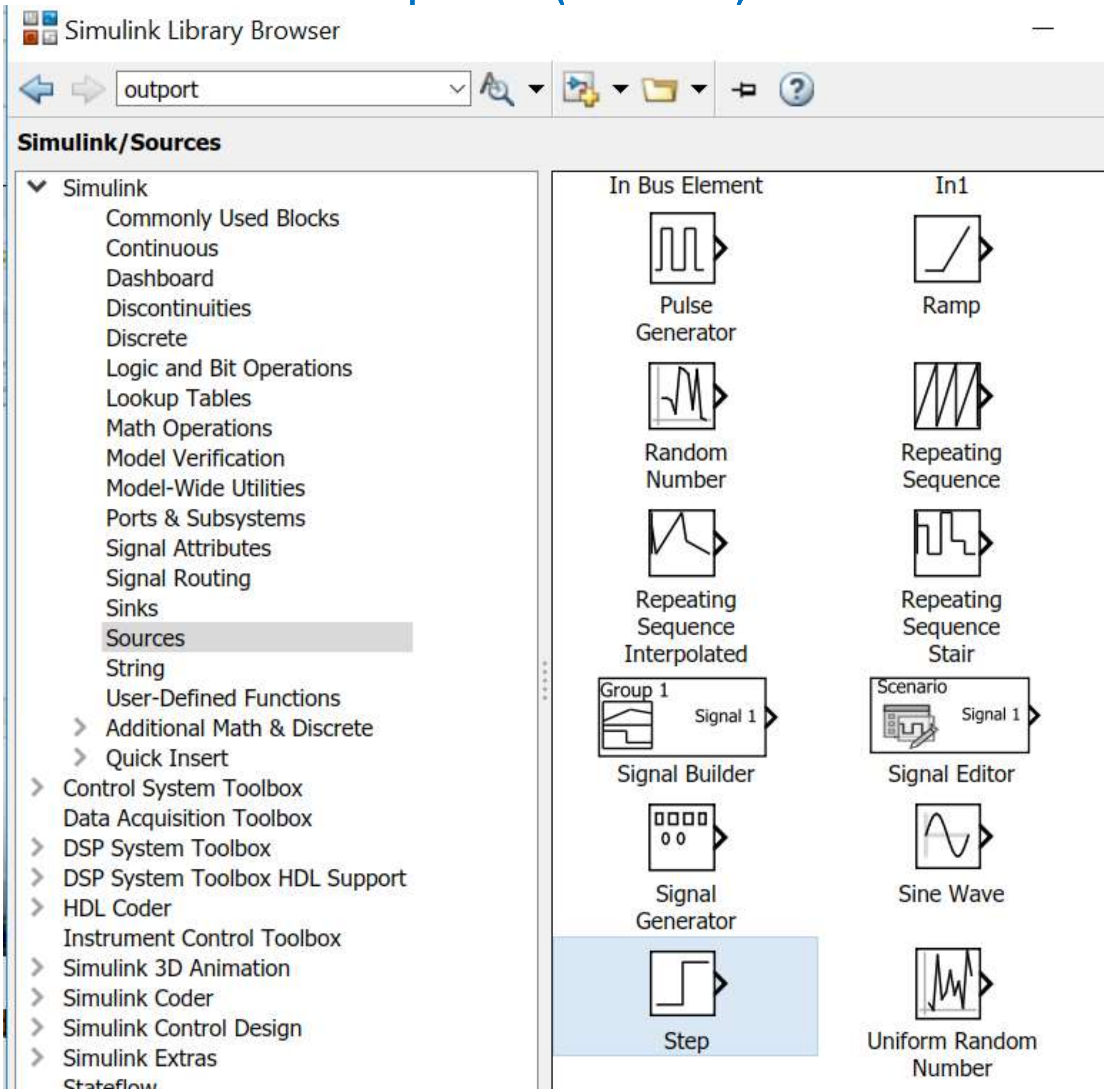
Double-click on the block to enter new numerator and denominator.



Example 2 (cont'd)



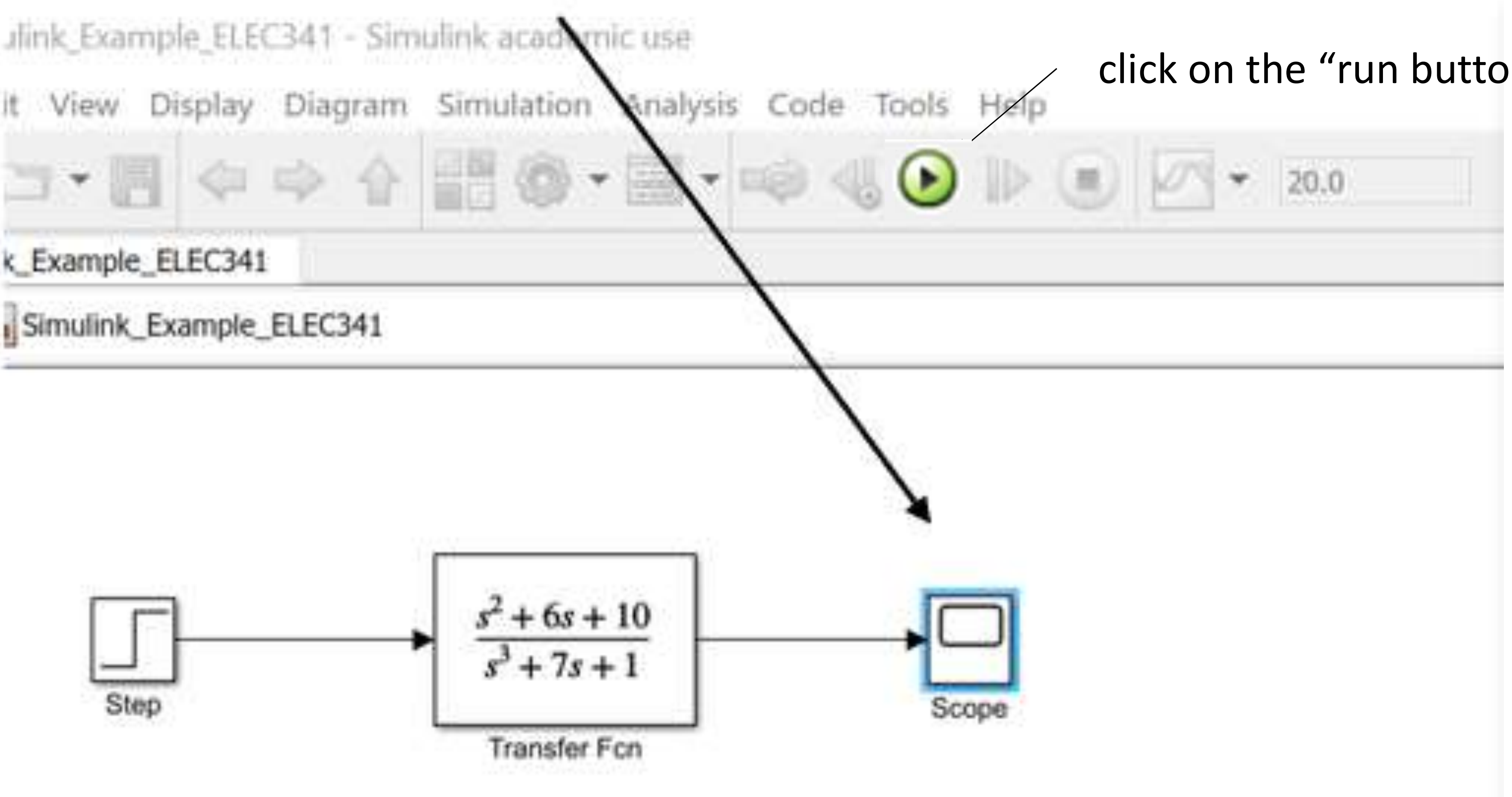
Example 2 (cont'd)



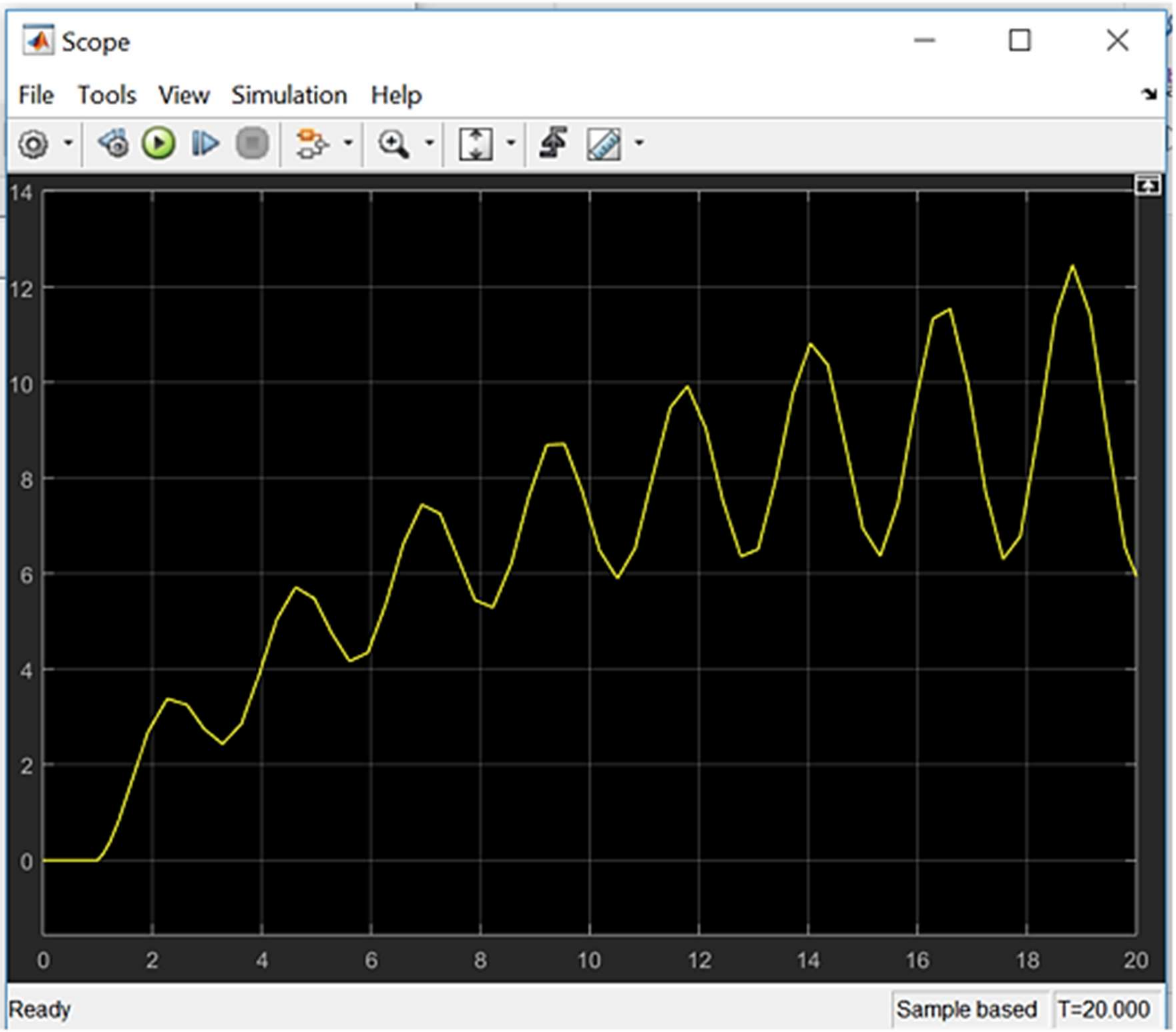
Example 2 (cont'd)

Double-click on scope.

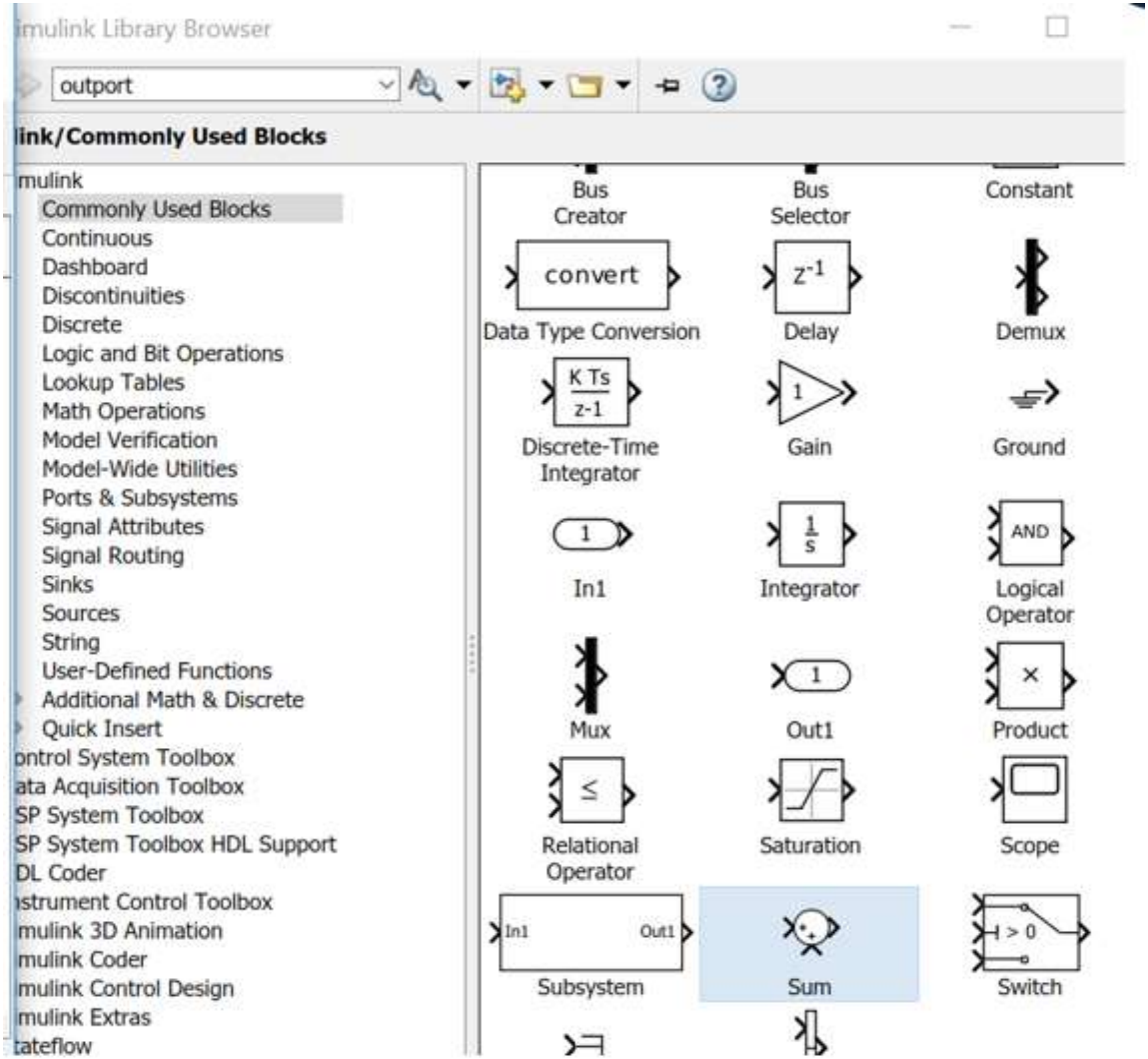
click on the "run button"



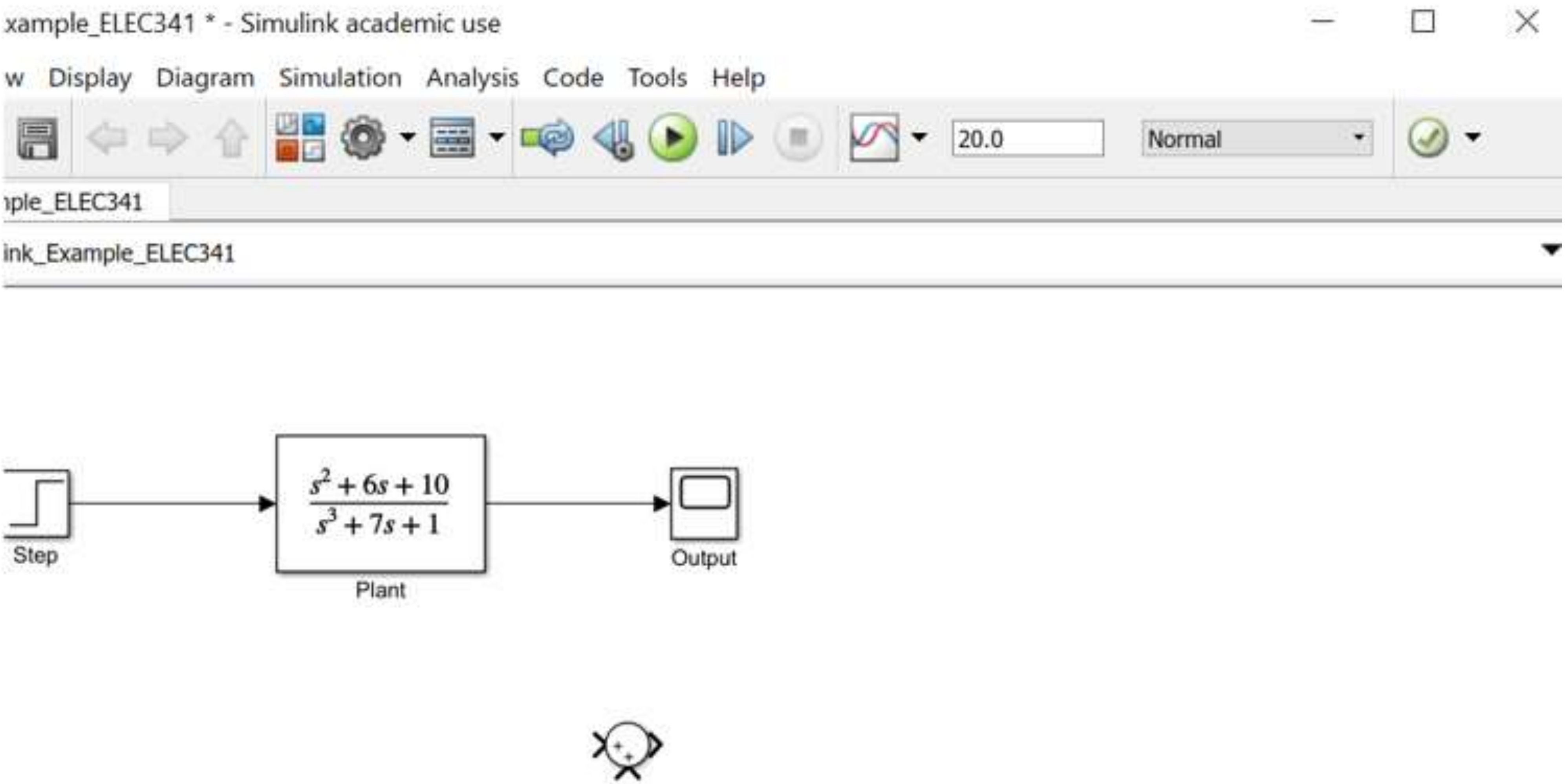
Example 2 (cont'd)



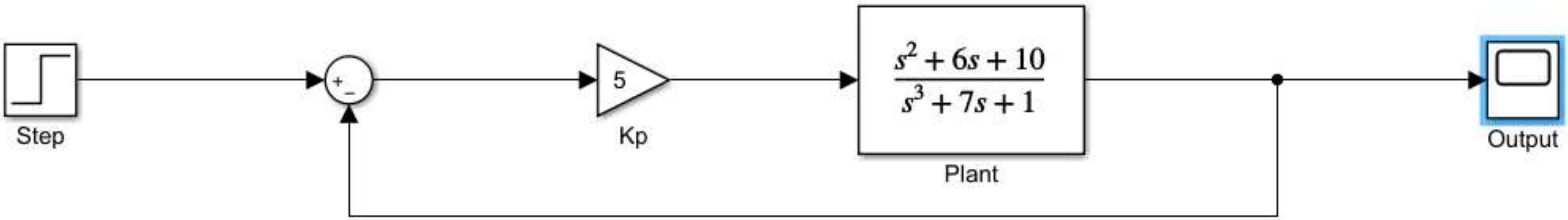
Example 2 (cont'd)

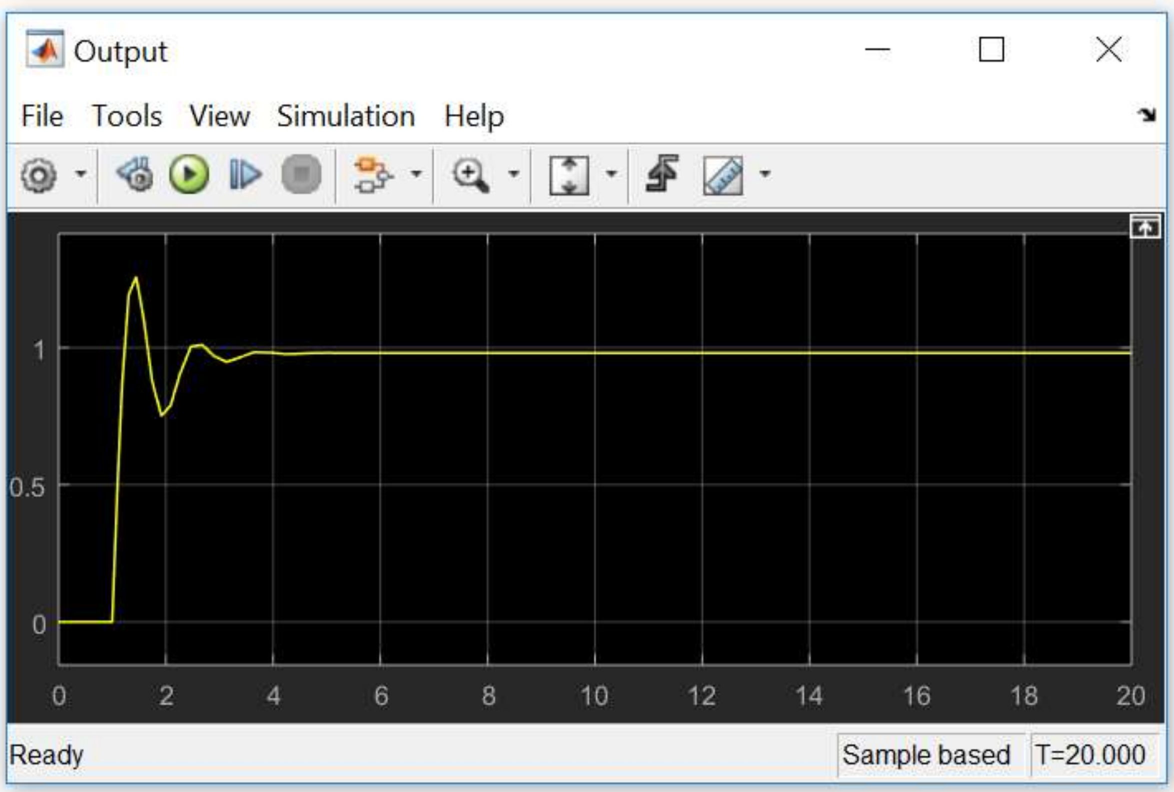


Example 2 (cont'd)



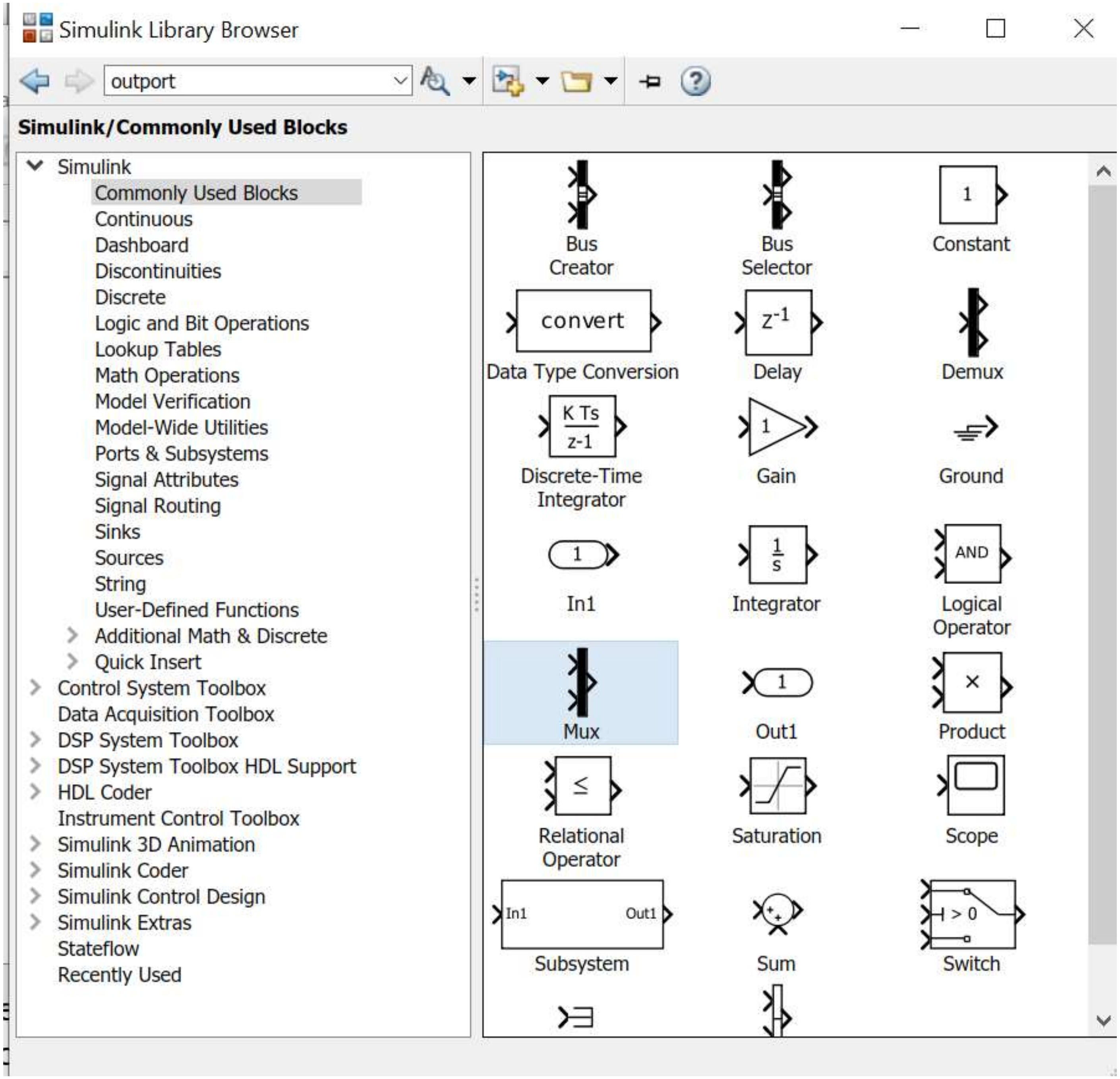
Example 2 (cont'd)



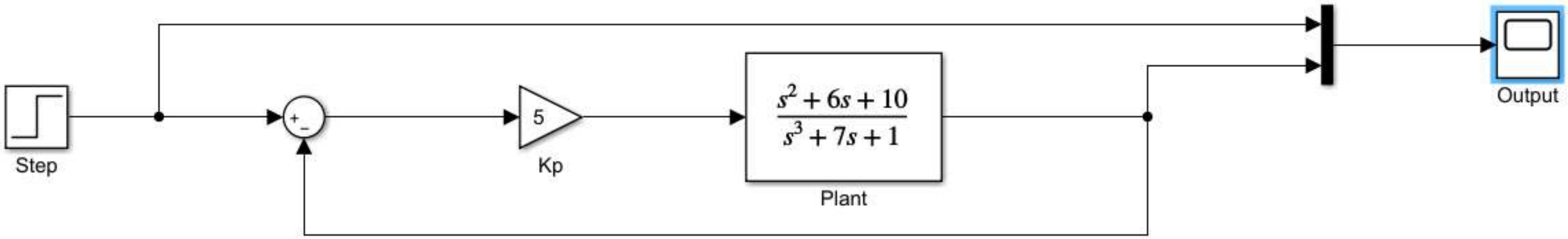


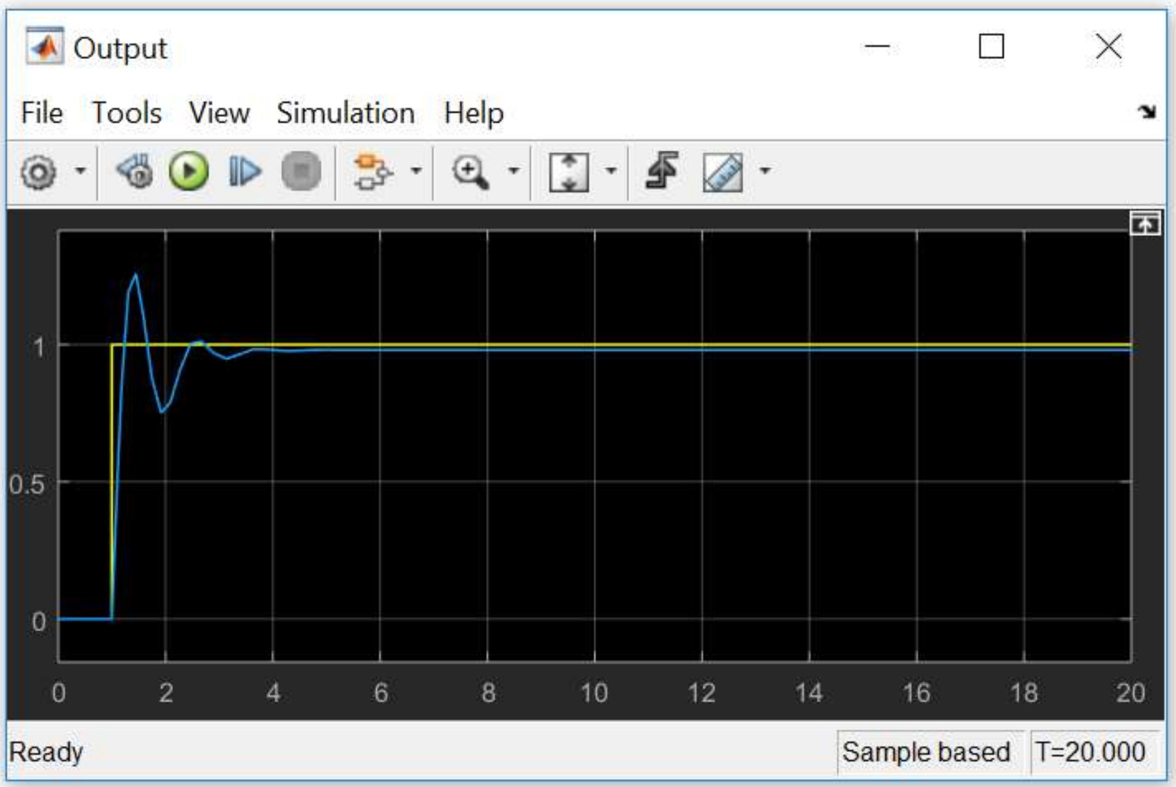
a place of mind

Example 2 (cont'd)

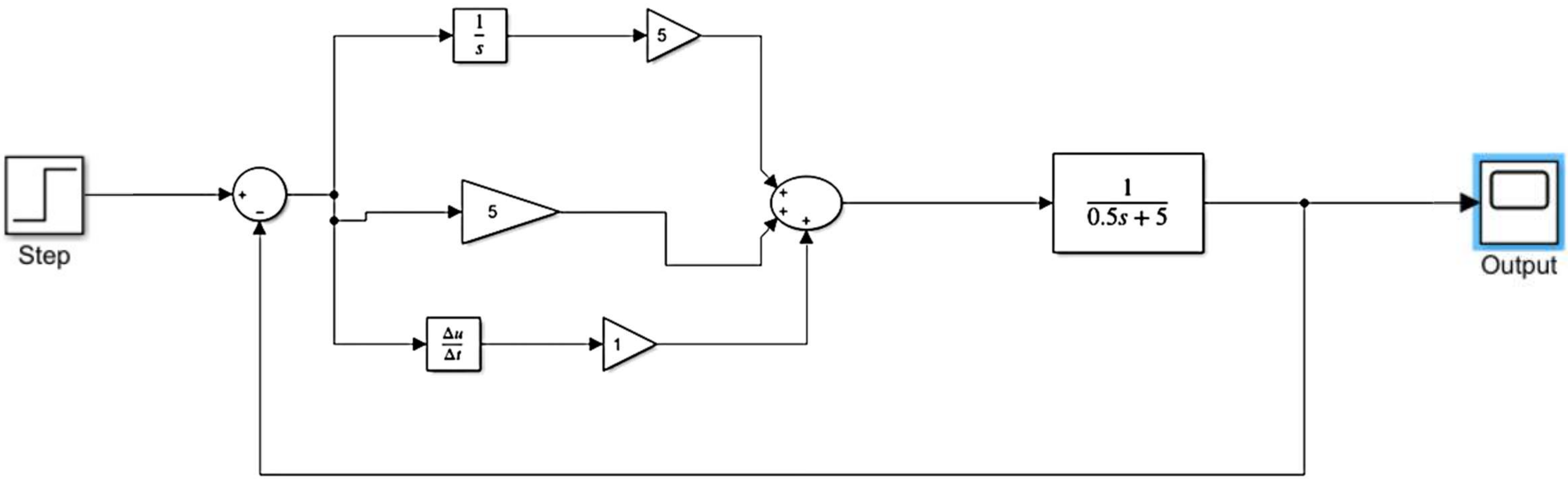


Example 2 (cont'd)

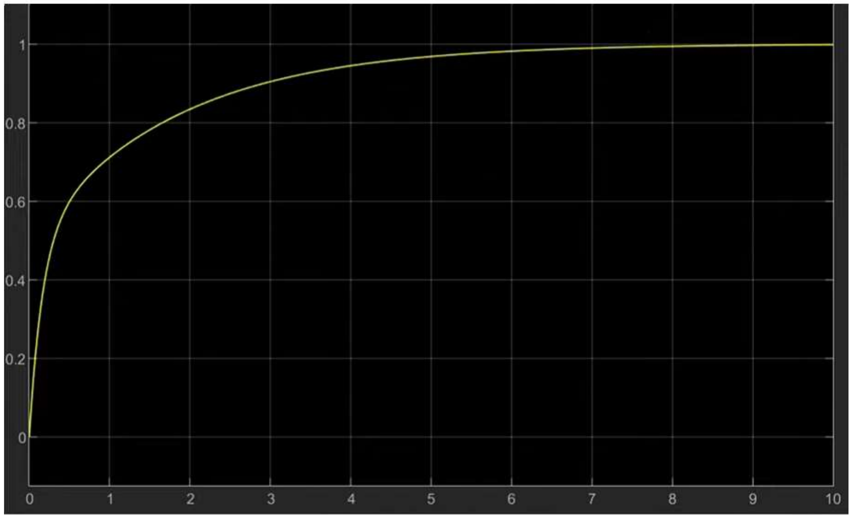




Example 3 (PID Controller Design)



Note: We can also use a block with "***s***" inside it as the derivative component of the PID controller design (instead of $\frac{\Delta u}{\Delta t}$).



Course roadmap

a place of mind UBC

Modeling

- ✓ Laplace transform
- ✓ Transfer function
- Models for systems
  - ✓ Electrical
  - ✓ Electromechanical
  - ✓ Mechanical
- ✓ Linearization, delay

Analysis

- ✓ Stability
  - ✓ Routh-Hurwitz
  - ✓ Nyquist
- ✓ Time response
  - ✓ Transient
  - ✓ Steady state
- ✓ Frequency response
  - ✓ Bode plot

Design

- ✓ Design specs
- ✓ Root locus
- ✓ Frequency domain
- ✓ PID & Lead-lag
- ✓ Design examples

✓ *Matlab simulations*

Thank You!

a place of mind

The End