ECE-420 Project 0: Introduction to the Embedded System Toolbox

Due: Friday September 11 at the beginning of class (e-mail me a memo)

In this project you will simulate a few discrete-time systems in both Matlab and Simulink to see how the embedded systems toolbox works. You will need to download files from the class website for this.

Mathematical Background: Consider a simple discrete-time transfer function with input $U(z)$ and output $Y(z)$,

$$G_p(z) = \frac{Y(z)}{\mathrm{U}(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

Cross multiplying we get

$$Y(z) + a_1\, z^{-1}\, Y(z) + a_2\, z^{-2}\, Y(z) = b_0\, U(z) + b_1\, z^{-1}\, \mathrm{U}(z) + b_2\, z^{-2}\, \mathrm{U}(z)$$

In the time-domain this becomes

$$y(n) = -a_1\, y(n-1) - a_2\, y(n-2) + b_0\, u(n) + b_1\, u(n-1) + b_2\, u(n-2)$$

In what follows it is important to continually think about this difference equation and how it is related to the transfer function.

<u>Part A</u>

Open the files **openloop_DE_A.slx** and **openloop_driver.m.** These files generate both a Matlab and a Simulink simulation of the unit step response for the discrete-time transfer function

$$G_p(z) = \frac{1}{z - 0.8} = \frac{z^{-1}}{1 - 0.8z^{-1}}$$

In the time domain this corresponds to the difference equation $y(n) = 0.8y(n-1) + u(n-1)$

Run **openloop_driver.m**, you should get the graph shown in Figure 1, however, it may take a while. This figure shows the results from the Matlab and Simulink simulations are identical, which is a good way to check your answers.

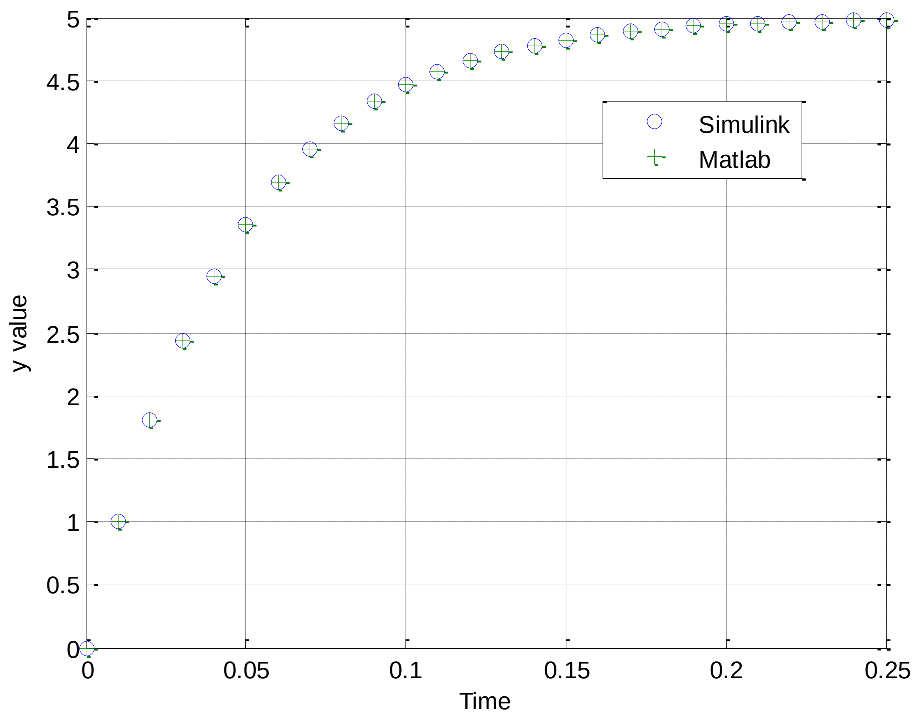


Figure 1. Open loop response of the system.

2) Now open up **openloop_DE_A.slx** (double click on it). It should look like Figure 2. We will be going through a number of the elements in this model so you can see how everything works.

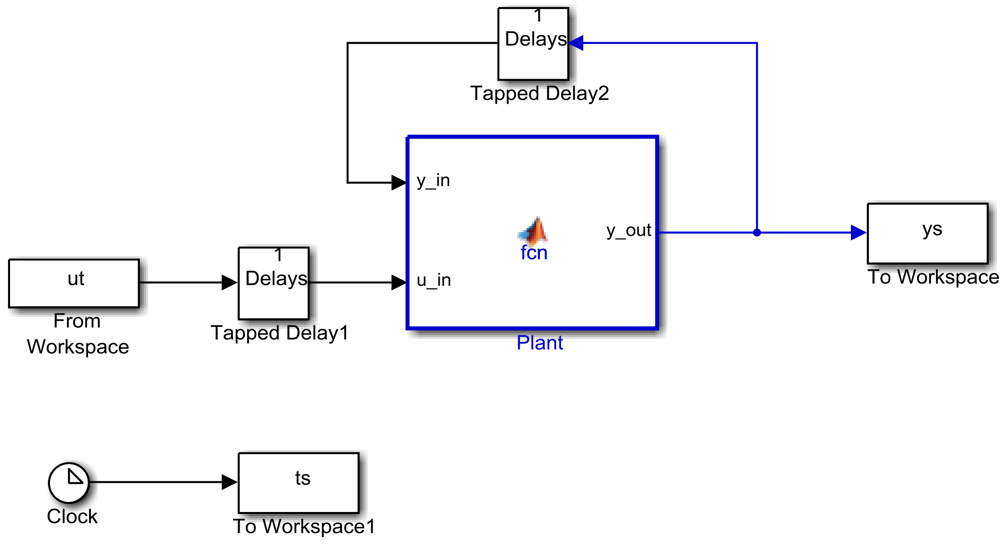
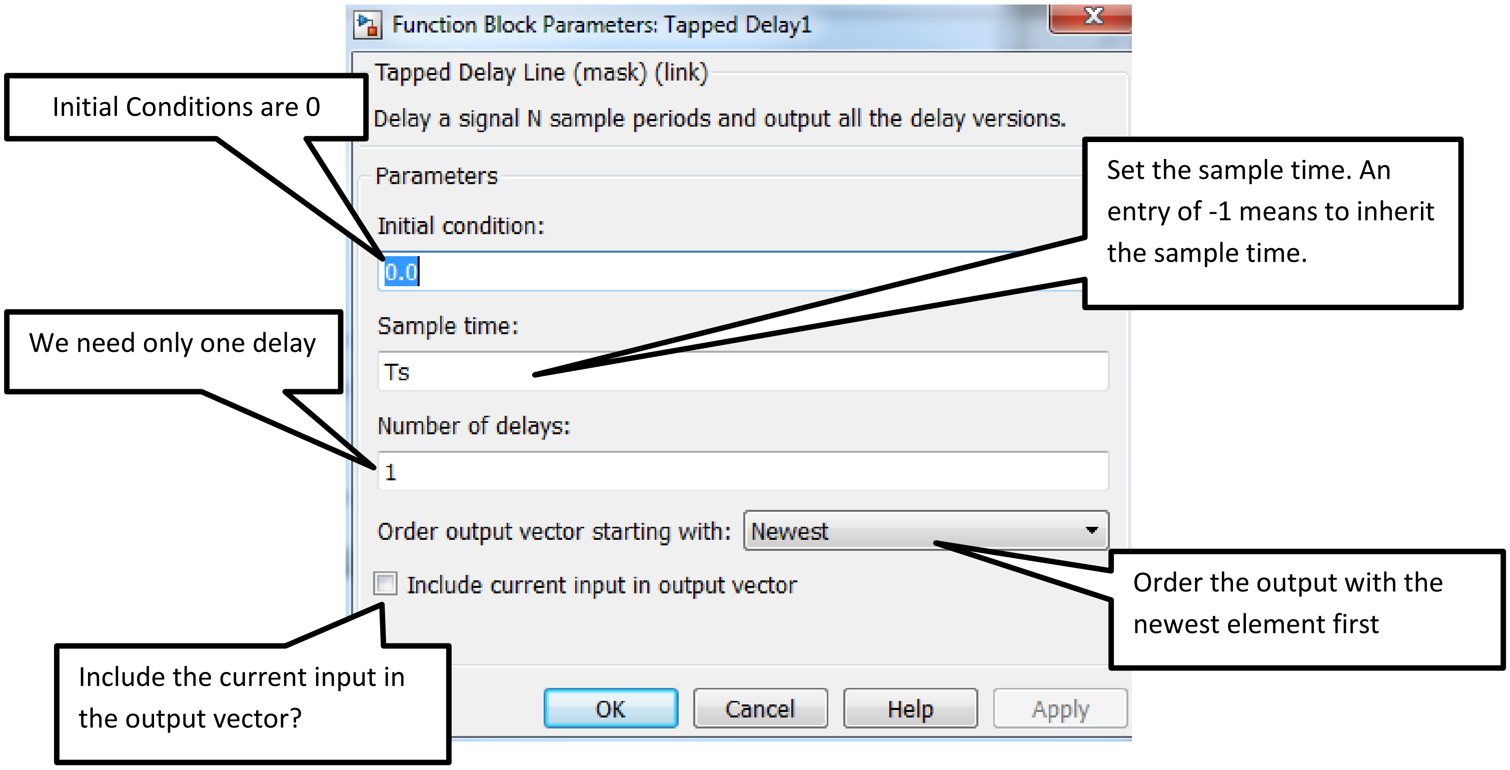


Figure 2. **openloop_DE_A.slx**

Starting at the left of the model (Figure 2), the input goes into a Delay Block (Tapped Delay 1). If you click on this delay block you will get the parameter block shown in Figure 3. This figure also shows what many of the parameters mean.



Function Block Parameters: Tapped Delay1

Tapped Delay Line (mask) (link)

Delay a signal N sample periods and output all the delay versions.

Parameters

Initial condition: 0.0

Sample time: Ts

Number of delays: 1

Order output vector starting with: Newest

☐ Include current input in output vector

Initial Conditions are 0

Set the sample time. An entry of -1 means to inherit the sample time.

We need only one delay

Order the output with the newest element first

Include the current input in the output vector?

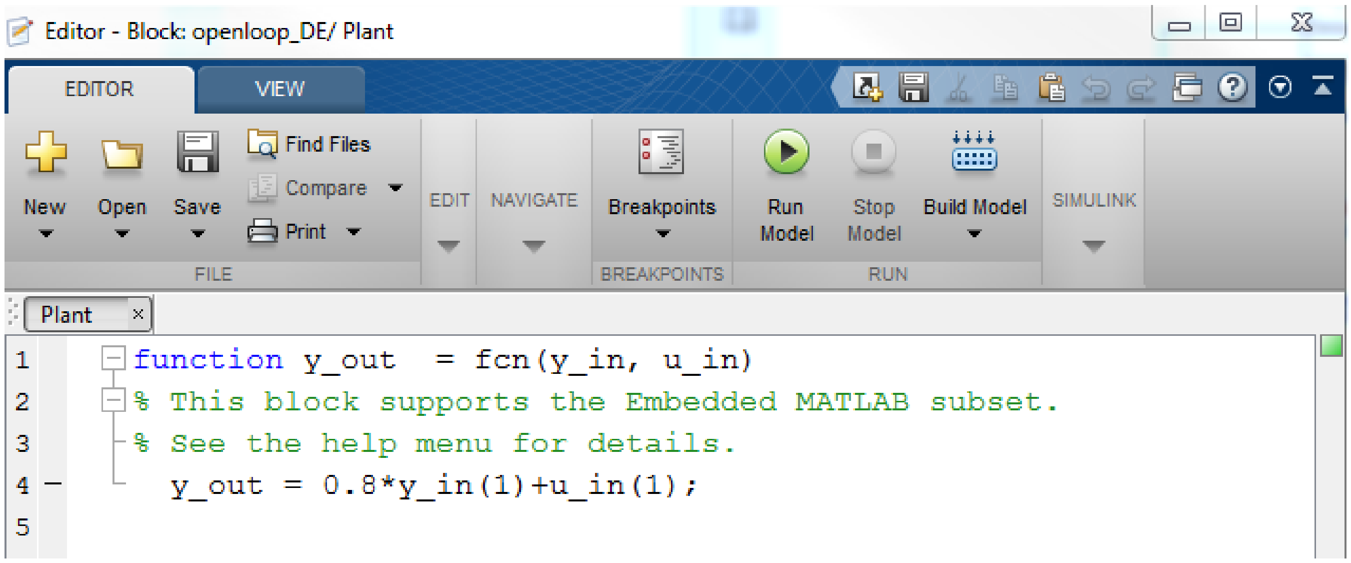
Figure 3. Parameter block for first delay, with explanations.

The output of this block is a scalar or a column vector. Figure 4 indicates the structure of this vector and how to access elements of the vector. *This is important to remember*!

$$
u\_in = \begin{cases}
\underbrace{\begin{bmatrix} u(n-1) \\ u(n-2) \\ u(n-3) \\ \vdots \\ u(n-N) \end{bmatrix}}_{\substack{\textit{If Include current input in} \\ \textit{output vector } \textbf{\textit{is not checked}}}} \leftrightarrow \underbrace{\begin{bmatrix} u(1) \\ u(2) \\ u(3) \\ \vdots \\ u(N) \end{bmatrix}}_{\substack{\textit{How to access the} \\ \textit{elments of } u\_in}} \leftrightarrow \underbrace{\begin{bmatrix} newest \\ \downarrow \\ \downarrow \\ oldest \end{bmatrix}}_{\substack{\textit{The order of the elements if} \\ \textit{Order output vector starting} \\ \textit{with } \textbf{\textit{Newest}} \textit{ is selected}}} \\
\underbrace{\begin{bmatrix} u(n) \\ u(n-1) \\ u(n-2) \\ \vdots \\ u(n-N) \end{bmatrix}}_{\substack{\textit{If Include current input in} \\ \textit{output vector } \textbf{\textit{is checked}}}} \leftrightarrow \underbrace{\begin{bmatrix} u(1) \\ u(2) \\ u(3) \\ \vdots \\ u(N+1) \end{bmatrix}}_{\substack{\textit{How to access the} \\ \textit{elments of } u\_in}} \leftrightarrow \underbrace{\begin{bmatrix} newest \\ \downarrow \\ \downarrow \\ oldest \end{bmatrix}}_{\substack{\textit{The order of the elements if} \\ \textit{Order output vector starting} \\ \textit{with } \textbf{\textit{Newest}} \textit{ is selected}}}
\end{cases}
$$

Figure 4. Output structure of a delay block. In the top row the **current input is not included** in the output. For both rows we assume there are N **delays** and the elements are in the **newest first** order. The middle vectors show how to access elements of the vector.

Now click on the plant (the large block in the center of Figure 2) and you will get the innocent looking piece of code for implementing a discrete-time transfer function shown in Figure 5. There are two things "passed" to this function: *y_in*, and *u_in.*

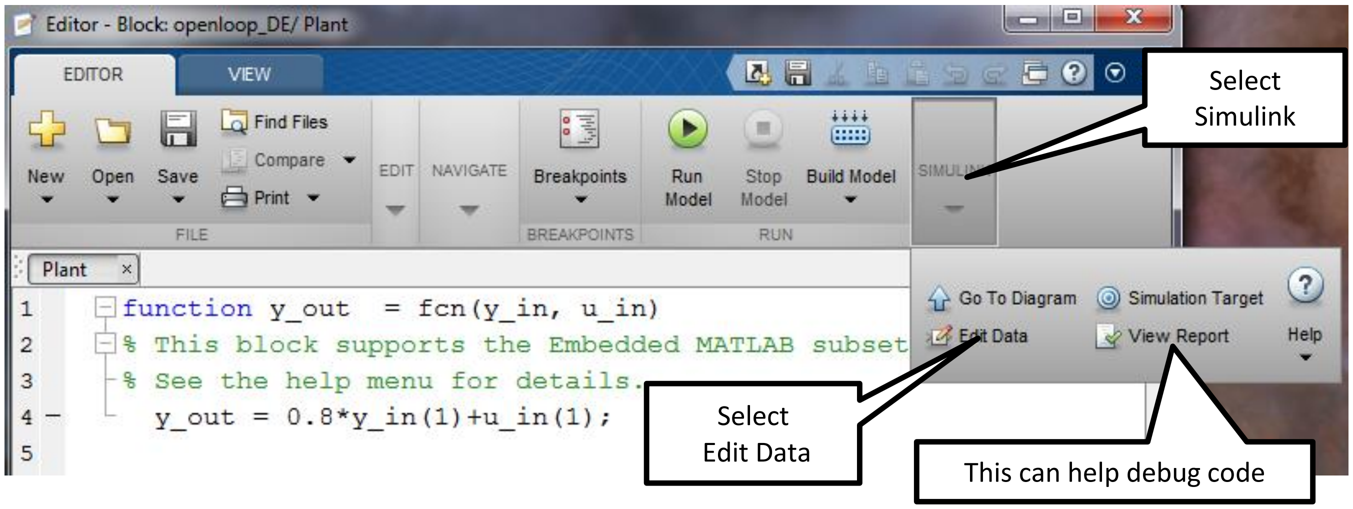


```
function y_out  = fcn(y_in, u_in)
% This block supports the Embedded MATLAB subset.
% See the help menu for details.
  y_out = 0.8*y_in(1)+u_in(1);

```

Figure 5. Code inside the plant block for implementing a discrete-time transfer function.

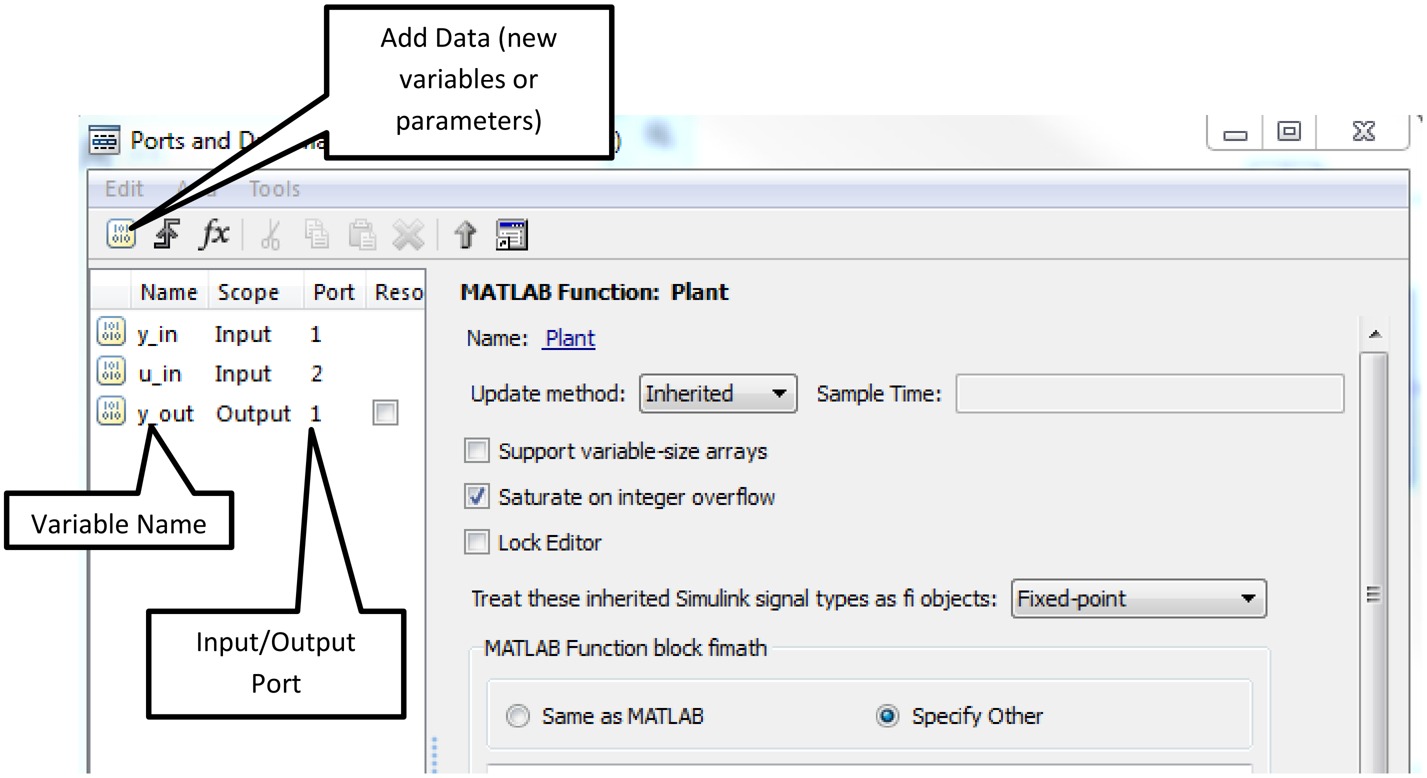
Click on the Simulink and then the Edit Data icon, as shown in Figure 6, to access the variables. (Note that you may not need to click on the Simulink tab.)



```
function y_out  = fcn(y_in, u_in)
% This block supports the Embedded MATLAB subset
% See the help menu for details.
   y_out = 0.8*y_in(1)+u_in(1);
```

Figure 6. Accessing the “passed” items to this block via the Edit Data icon.

Once you click on the Edit Data icon, you will get a window like that shown in Figure 7.



Name	Scope	Port	Reso
y_in	Input	1	
u_in	Input	2	
y_out	Output	1	☐

MATLAB Function: Plant

Name: Plant

Update method: Inherited Sample Time:

☐ Support variable-size arrays

☑ Saturate on integer overflow

☐ Lock Editor

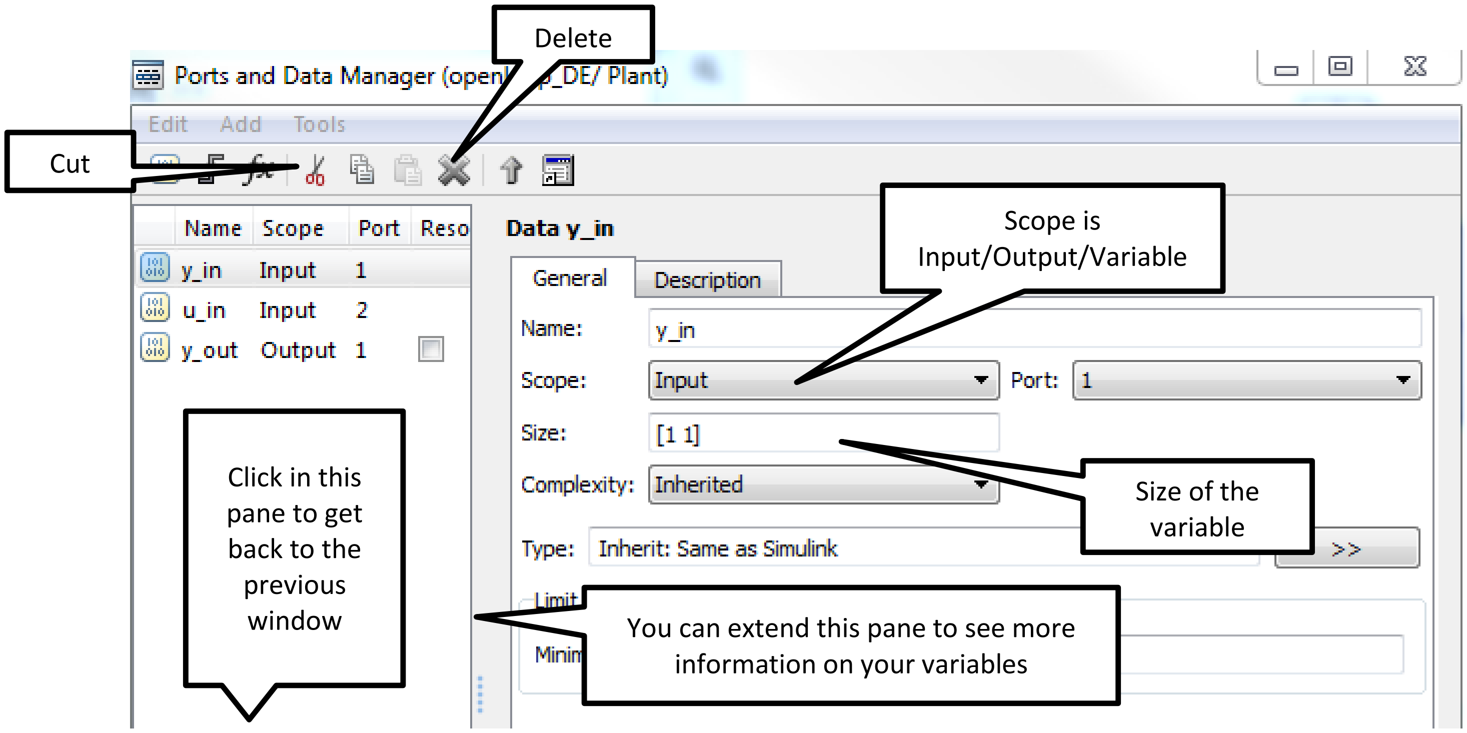
Treat these inherited Simulink signal types as fi objects: Fixed-point

MATLAB Function block fimath

○ Same as MATLAB ● Specify Other

Figure 7. Modifying variables and parameters

If you select one of the variables (such as **y_in**), you will get a figure like that in Figure 8. Click in the left-half plane to get back to the previous window.


Delete
Ports and Data Manager (open/_DE/ Plant)
Cut
Edit Add Tools
Scope is Input/Output/Variable
Click in this pane to get back to the previous window
Size of the variable
You can extend this pane to see more information on your variables
General
Description
Limit
Minim
>>

Name	Scope	Port	Reso
y_in	Input	1	
u_in	Input	2	
y_out	Output	1	☐

Data y_in

Name: y_in

Scope: Input

Port: 1

Size: [1 1]

Complexity: Inherited

Type: Inherit: Same as Simulink

Figure 8. Information on simulation variables.

Referring to Figures 7 and 8,

- If the **scope** is a **parameter**, then you should not expect it to change as the simulation is running. If the scope is an **input** or **output**, then you should expect it to change as the simulation is running, such as **y_in**, **r_in**, and **y_out**.
- The **port** indicates the order in which the variable will appear in the left (input) or right (output) side of the block. For this simulation, the **port** of **y_in** is **one** and the **port** of **u_in** is **two**, and this is reflected in how they are placed on the left of the **plant** block (See Figure 2).
- Most likely you will be changing the **Size** of the variables more than anything else.
- **Add Data** allows you to add new variables and parameters.

Part B

Copy **openloop_DE_A.slx** to **openloop_DE_B.slx (**save **Openloop_DE_A** as **openloop_DE_B)** and modify code as needed to determine the step response of the discrete-time plant

$$G_p(z) = \frac{1-0.2z^{-1}}{1-0.8z^{-1}}$$

Assume a sampling interval of 0.1 seconds and run the simulation for 3 seconds. You should start by writing out the appropriate difference equation. You will then have to

- modify the first delay bock,
- modify the plant block so the size of *u_in* is correct (make a *column* vector)
- modify the code inside the plan block
- change the transfer function in the Matlab driver file
- change the Simulink file the Matlab driver file invokes

If Matlab complains that there is an error in the Simulink model, it is often useful to click on the Simulink block in question and then select View Report (See Figure 6.). This will often help find the errors.

When you run the Matlab driver file you should get a plot like that shown in Figure 9. *Include your plot in your memo.*

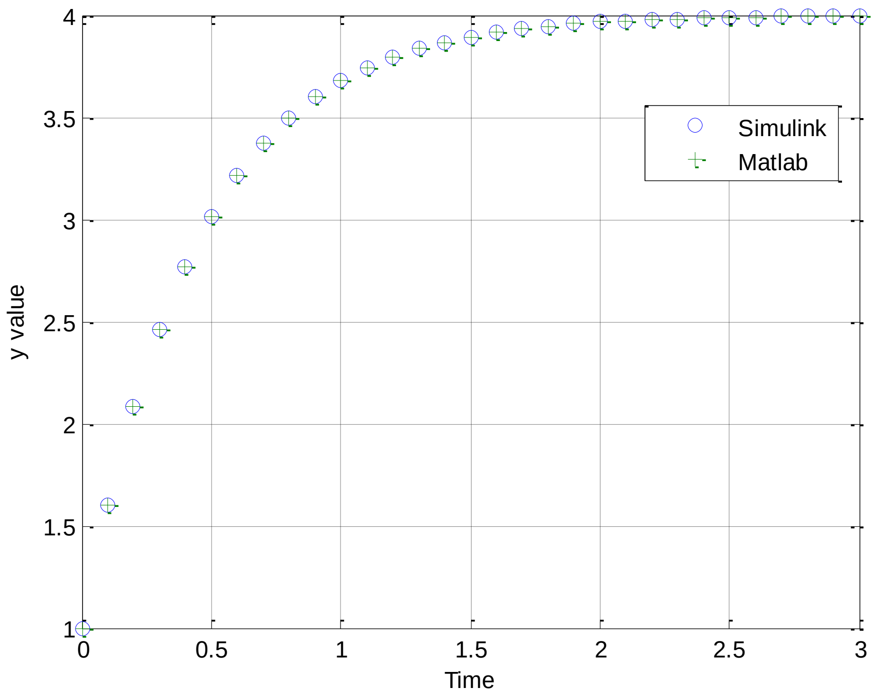


Figure 9. Step response for $G_p(z) = \frac{1-0.2z^{-1}}{1-0.8z^{-1}}$

Part C

Copy **openloop_DE_B.slx** to **openloop_DE_C.slx** and modify the code (both Matlab and Simulink) as needed to determine the step response of the discrete-time plant

$$G_p(z) = \frac{0.2z^{-1} + 0.1z^{-2}}{1 - 0.1z^{-1} + 0.6z^{-2}}$$

Assume a sampling interval of 0.1 seconds and run the simulation for 2 seconds. You should start by writing the difference equation in the time domain. You will need to change both of the delay blocks and the plant block (both the code and the dimensions of the signals.) You should get a plot like that shown in Figure 10. *Include your plot in your memo.*

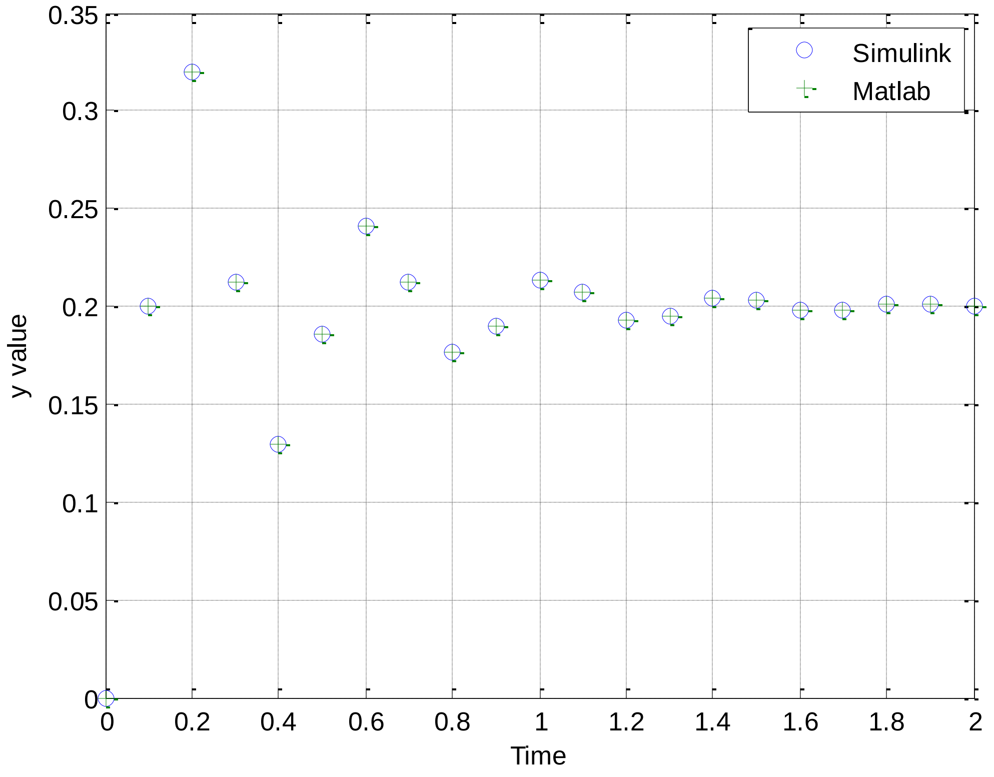


Figure 10. Step response for $G_p(z) = \frac{0.2z^{-1} + 0.1z^{-2}}{1 - 0.1z^{-1} + 0.6z^{-2}}$

Part D

Now we want to implement a controller in addition to our plant. As with our plants, it is easiest to get everything correct if you first write out a difference equation, and then figure out what you need in the delays and sizes of variables.

Copy your Simulink file **openloop_DE_A.slx** to **closedloop_DE_A.slx**. Your plant for part A should already be modelled correctly, and now we want to implement the integral controller $G_c(z) = \frac{0.04}{z-1}$. We also want a delay between the output and the input, so $H(z) = z^{-1}$ (this is a simple delay). To implement the controller, again write out the difference equation in the time domain, and then modify **closedloop_DE_A.slx** so it looks something like that in Figure 11. *Note that the input is now rt, not ut, and the input and output variables in the controller have different names than for the plant.* It is probably easiest to copy your plant and then make it a controller.

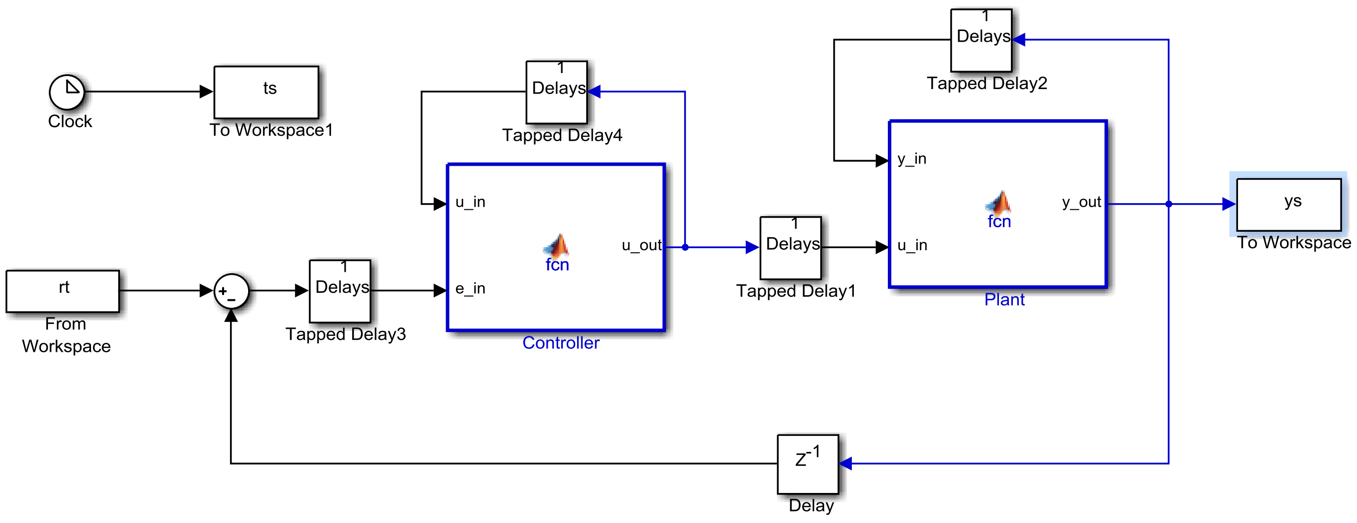


Figure 11. Closed loop system for plant A.

If you run the program **closedloop_driver.m** with a sampling interval 0.1 seconds and your Simulink is working properly you should get a figure like that in Figure 12. *Include your plot in your memo.*

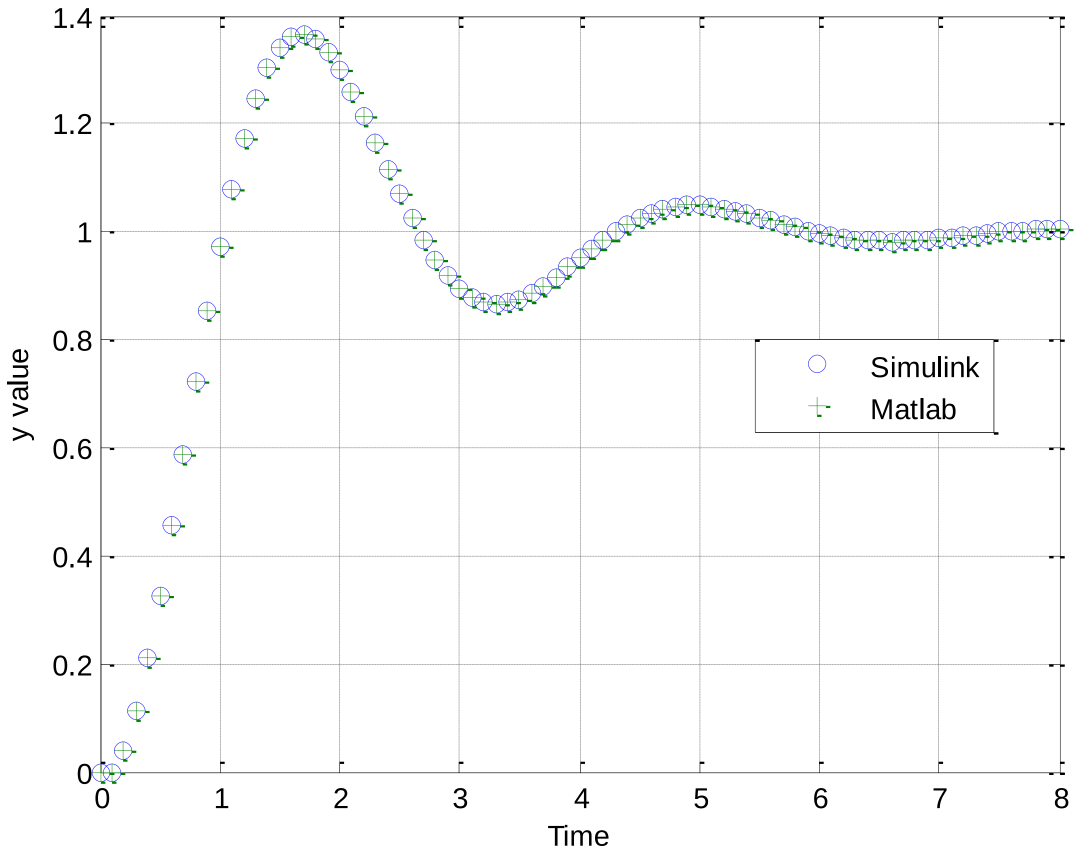


Figure 12. Result of Matlab and Simulink simulations for system A with an integral controller.

Part E

Copy your Simulink file **openloop_DE_B.slx** to **closedloop_DE_B.slx**. Your plant for part B should already be modelled correctly, and now we want to implement the proportional +integral controller $G_c(z) = \frac{0.115(\text{z}-0.547)}{z-1}$. We also want a delay between the output and the input, so $H(z) = z^{-1}$ (this is a simple delay). You may also want to copy parts of your model from **closedloop_DE_A.slx**. To implement the controller, again write out the difference equation in the time domain, and then modify **closedloop_DE_B.slx** so it again looks something like that in Figure 11 (though the delays may be different). After you have modified this file and the Matlab driver file (with a sampling interval of 0.1 sec), your results should look like those in Figure 13. *Include your plot in your memo.*

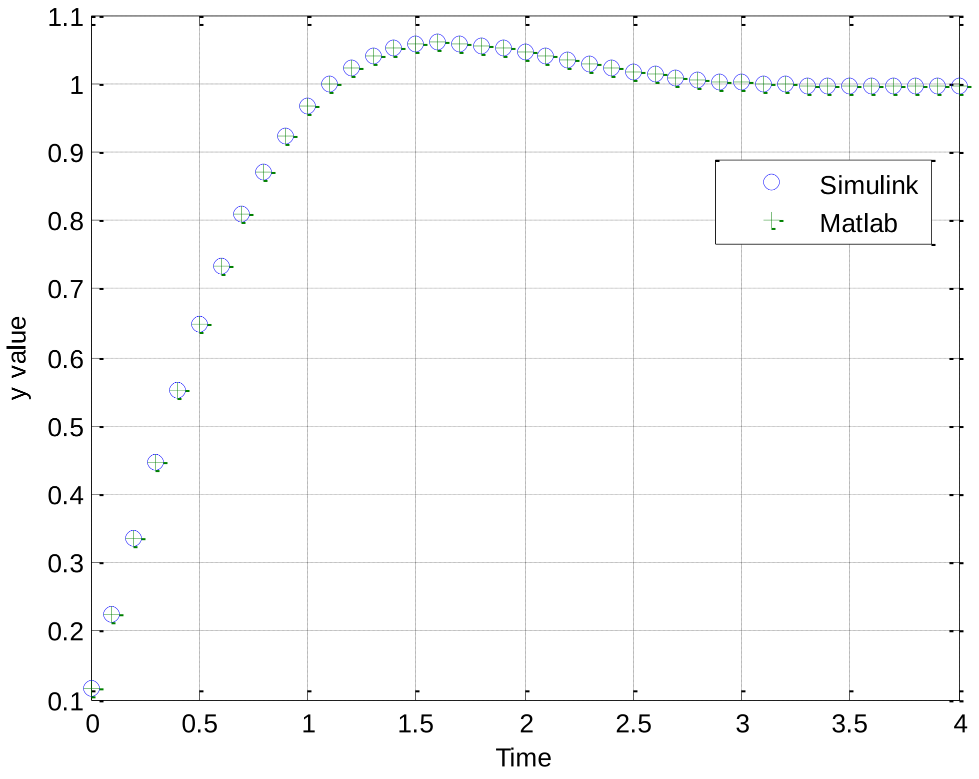


Figure 13. Result of Matlab and Simulink simulations for system B with a proportional plus integral controller.

Part F

Copy your Simulink file **openloop_DE_C.slx** to **closedloop_DE_C.slx**. Your plant for part C should already be modelled correctly, and now we want to implement the proportional +integral+derivative controller $G_c(z) = \frac{1.0631(z^2 - 0.316z + 0.199)}{z(z-1)}$. We also want a delay between the output and the input, so $H(z) = z^{-1}$ (this is a simple delay). To implement the controller, again write out the difference equation in the time domain, and then modify **closedloop_DE_C.slx** so it again looks something like that in Figure 11 (though the delays may be different). After you have modified this file and the Matlab driver file (with a sampling interval of 0.1 sec), your results should look like those in Figure 14. *Include your plot in your memo.*

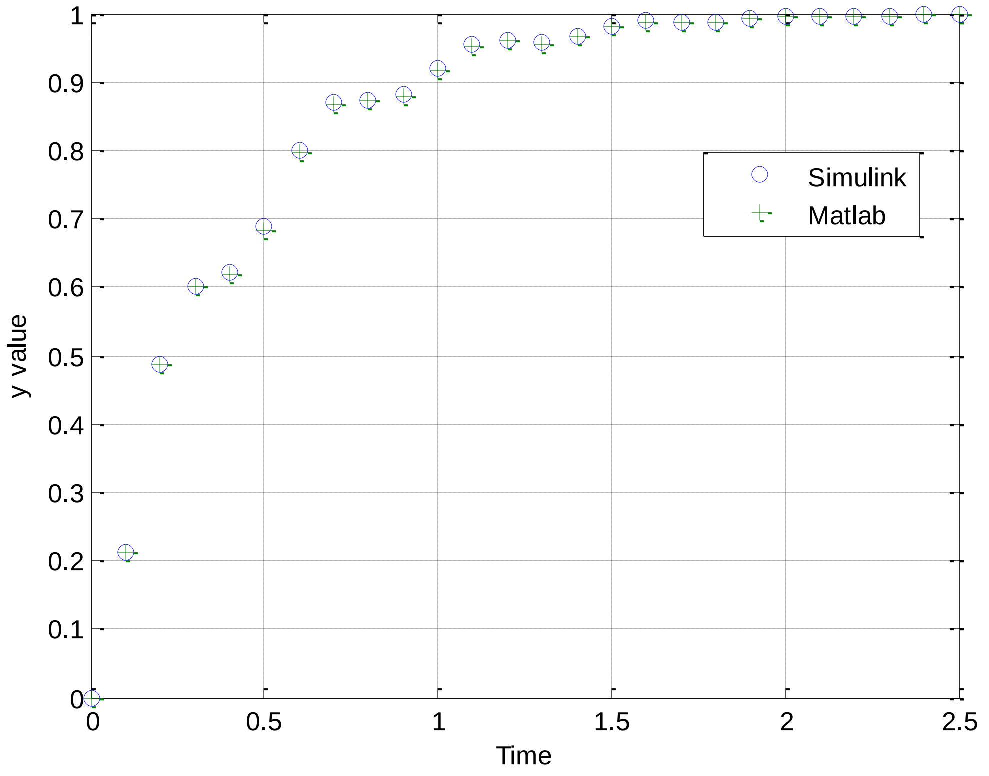


Figure 14. Result of Matlab and Simulink simulations for system C with a proportional plus integral plus derivative controller.